Crime Linkage and Unsolved Crimes

Dr. Jeffrey Strickland

8/28/2018

Introduction to Crime Series Linkage

Statistical linkage and clustering of criminal events can be used by crime analysts to create of lists of potential suspects for an unsolved crime, identify groups of crimes that may have been committed by the same individuals or group of individuals, for offender profiling, and for predicting future events. Pairwise case linkage attempts to establish if a \emph{pair} of crimes share a common offender. In practice, there is more interest in **crime series linkage**, which attempts to identify the *set* of crimes committed by a common offender. For example, a crime analyst may need to identify the additional crimes that are part of a crime series (crime series identification) or discover all the crime series in a criminal incident database (crime series clustering).

Crime series clustering is fundamentally a clustering problem where we seek to group the crimes into clusters that correspond to a serial offender (or group of offenders). Instead of simultaneously clustering all crimes in a criminal database, **Crime series identification** is focused on identifying the additional unsolved crimes that are part of an existing crime series. Crime series identification can be useful, for example, in interrogations [@Adderly-Musgrove-2003; @Adderly-2004] by providing investigators a list of additional crimes (like the known crimes) that a suspect may be responsible for. A crime analyst could also use crime series identification when investigating a crime series or suspected crime series to generate a list of additional crimes to jointly investigate.

The `crimelinkage` package provides several tools for crime series identification and clustering based on methods from hierarchical and model-based clustering.

Trial Run with crimeLinkage

It is a good idea to open the help file for fitting with Bayesian model-based partially-supervised clustering for reference, so we do it in the code, lest we forget. help("crimeClust_bayes") Now, we will run the crimeLinkage documentation example for fitting with Bayesian model-based partially-supervised clustering. This will familiarize you with the function and its parameters.

Make IDs: Criminal 1 committed crimes 1-4, etc.

```
id <- c(1,1,1,1,
        2,2,2,2,
        3,3,3,3)
```

spatial locations of the crimes:

```
s <- c(0.8,0.9,1.1,1.2,
       1.8,1.9,2.1,2.2,
       2.8,2.9,3.1,3.2)
s <- cbind(0,s)
```

We can categorical crime features, say mode of entry (1=door, 2=other) and type of residence (1=apartment, 2=other). # Different distribution by criminal

```
Mode <- c(1,1,1,1,
          1,2,1,2,
          2,2,2,2)
```

Same distribution for all criminals

```
Type <- c(1,2,1,2,
          1,2,1,2,
          1,2,1,2)
Xcat <- cbind(Mode,Type)
```

Times of the crimes

```
t <- c(1,2,3,4,
       2,3,4,5,
       3,4,5,6)
```

Now let's pretend we don't know the criminal for crimes 1, 4, 6, 8, and 12.

```
id <- c(NA,1,1,NA,2,NA,2,NA,3,3,3,NA)
```

Then, we fit the model (naïve Bayes, NB, use much larger iterations and burn on a real problem)

```
fit1 <- crimeClust_bayes(crimeID=id, spatial=s, t1=t,t2=t,
Xcat=Xcat,maxcriminals=12,iters=500,burn=100,update=100)

# Plots are omitted
```

```
summary(fit1)

##              Length Class  Mode
## p.equal       144   -none- numeric
## D             500   -none- numeric
## df           1000   -none- numeric
## sd1           500   -none- numeric
## sd2           500   -none- numeric
## sds          1000   -none- numeric
## theta         500   -none- numeric
## s.miss          0   -none- NULL
```

```
## t.censored     0   -none- NULL
## missing_s      0   -none- numeric
## missing_t      0   -none- numeric
## crimeID       12   -none- numeric
```

Now we plot the posterior probability matrix in which each pair of crimes was contained, and view the model summary.

committed by the same criminal:

```
if(require(fields,quietly=TRUE)) {
     fields::image.plot(1:12,1:12,fit1$p.equal,
      xlab="Crime", ylab="Crime",
main="Probability crimes are from the same criminal")
}
```

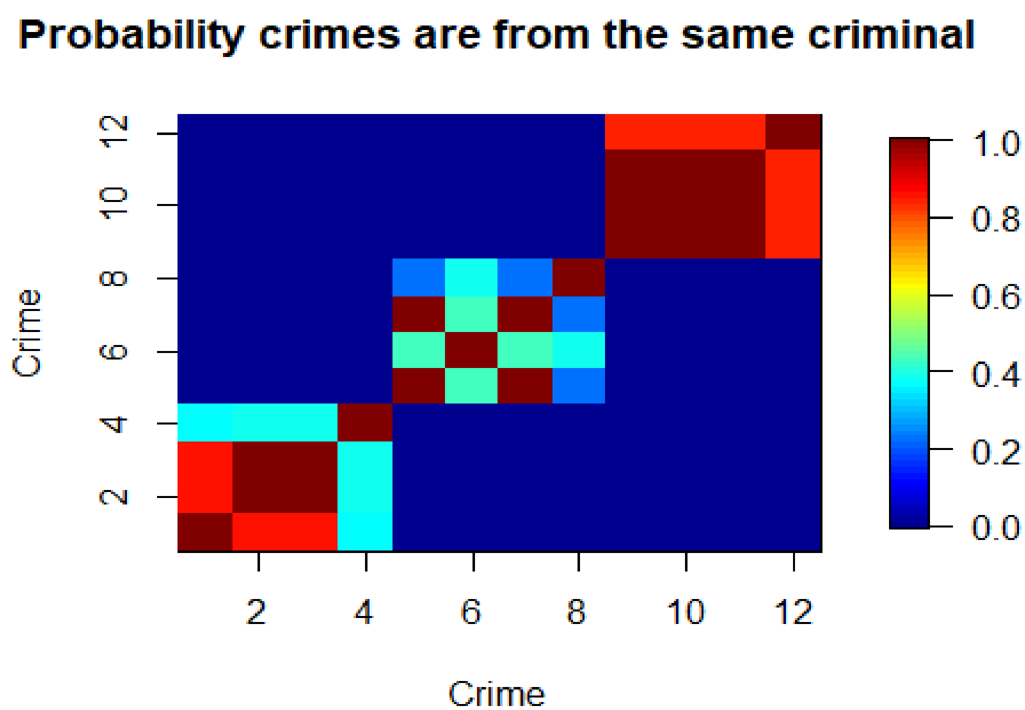

Figure 1: This “heatmap” plot show the probability that the is a linkage between unsolved and solved crime. The “hot” spots (shades of red) have a very high chance of linkage and can likely be solved.

Preliminaries

Using crimeLinkage for Analyzing Unsolved Crimes

We should first check to see if the package is installed using:

```
installed.packages("crimeLinkage")
```

```
Package LibPath Version Priority Depends Imports LinkingTo Suggests Enhances
License
License_is_FOSS License_restricts_use OS_type Archs MD5sum NeedsCompilation Built
```

The output above shows use that `crimeLinkage` is installed. If it were not, we would install the package using:

```r
install.packages("crimeLinkage")
```

Next, we need to load the package and get the example crime data.

Load Libraries

```r
library(crimelinkage)
library(fields)
```

Load the Data

```r
data(crimes)
data(offenders)
head(crimes)
```

```
##   crimeID       X        Y MO1  MO2  MO3             DT.FROM
## 1     C:1 13661.1  -4659.3   6    a    J 1993-01-06 23:55:00
## 2     C:2  6758.8 -10092.4  25    a    E 1993-01-06 00:00:00
## 3     C:3 10077.6  -8810.9  25    a    E 1993-01-08 07:00:00
## 4     C:4 12080.3  -4011.1  25    a    E 1993-01-10 20:27:00
## 5     C:5 10862.5  -8091.0  19    b    C 1993-01-11 07:45:00
## 6     C:6 14032.9  -1945.4   7 <NA> <NA> 1993-01-04 15:30:00
##                 DT.TO
## 1 1993-01-07 04:39:00
## 2 1993-01-06 00:00:00
## 3 1993-01-08 17:30:00
## 4 1993-01-10 20:27:00
## 5 1993-01-11 07:45:00
## 6 1993-01-04 15:30:00
```

```r
head(offenders)
```

```
##   offenderID crimeID
## 1        O:1     C:6
## 2        O:2     C:2
## 3        O:3    C:29
## 4        O:4    C:19
## 5        O:5    C:18
## 6        O:6    C:18
```

Data Description

- crimeID The crime ID number
- X,Y Spatial
- coordinates MO1 A categorical
- MO variable that takes values 1,...,31

- MO2 A categorical MO variable that takes values a,…,h
- MO3 A categorical MO variable that takes values A,…,O
- DT.FROM The earliest possible Date-time of the crime.
- DT.TO The latest possible Date-time of the crime

Setup the Data for Analysis

Make a Crime Series

```
seriesData = makeSeriesData(crimedata=crimes,offenderTable=offenders)
head(seriesData)
```

```
##   crimeID Index CS offenderID                TIME
## 1     C:6     6  1        O:1 1993-01-04 15:30:00
## 2     C:9     9  2       O:10 1993-01-11 13:00:00
## 3   C:121   121  3      O:100 1993-06-02 03:14:00
## 4   C:127   127  4      O:101 1993-06-04 16:32:00
## 5   C:100   100  5      O:102 1993-05-17 00:00:00
## 6    C:94    94  6      O:103 1993-05-10 13:50:00
```

Make Crime Pairs for Case Linkage

```
set.seed(1)         # set random seed for replication
allPairs = makePairs(seriesData,thres=365,m=40)
```

Make Evidence Variables for Case Linkage

```
set.seed(1)         # set random seed for replication
allPairs = makePairs(seriesData,thres=365,m=40)
varnames = list(spatial = c("X", "Y"), temporal = c("DT.FROM","DT.TO"),
categorical = c("MO1",  "MO2", "MO3"))
X = compareCrimes(allPairs, crimes, varnames, binary=TRUE)  # Evidence data
Y = ifelse(X$type=='linked',1,0)      # Linkage indicator. 1=linkage,
0=unlinked
```

Build Training and Testing Data

```
set.seed(3) # set random seed for replication
train = sample(c(TRUE,FALSE),nrow(X),replace=TRUE,prob=c(.7,.3)) # assign
pairs to training set
test = !train
D.train = data.frame(X[train,],Y=Y[train]) # training data
D.test = data.frame(X[test,],Y=Y[test]) # training data
```

Fit Logistic Regression model and make estimateBF() function

```
spatial = c("X", "Y")
temporal = c("DT.FROM","DT.TO")
categorical = c("MO1",  "MO2", "MO3")
vars = c("spatial","temporal","tod","dow","MO1","MO2","MO3")
fmla.all = as.formula(paste("Y ~ ", paste(vars, collapse= "+")))
fmla.all
```

```
## Y ~ spatial + temporal + tod + dow + MO1 + MO2 + MO3
```

```
fit.logistic = glm(fmla.all,data=D.train,family=binomial,weights=weight)
```

```
## Warning in eval(family$initialize): non-integer #successes in a binomial
## glm!
```

```
summary(fit.logistic)
```

```
##
## Call:
## glm(formula = fmla.all, family = binomial, data = D.train, weights =
weight)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.7063  -0.0676  -0.0300  -0.0117   5.0136
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -2.013335   0.605822  -3.323  0.00089 ***
## spatial     -0.450665   0.086470  -5.212 1.87e-07 ***
## temporal    -0.016284   0.003922  -4.152 3.29e-05 ***
## tod         -0.128988   0.065646  -1.965  0.04943 *
## dow         -0.149256   0.183872  -0.812  0.41694
## MO11         1.507261   0.370704   4.066 4.78e-05 ***
## MO21         0.278068   0.364385   0.763  0.44539
## MO31         0.081398   0.412758   0.197  0.84367
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 424.84  on 7900  degrees of freedom
## Residual deviance: 308.86  on 7893  degrees of freedom
## AIC: 242.59
##
## Number of Fisher Scoring iterations: 10
```

```
estimateBF <- function(X){   # estimateBF() returns the estimated log Bayes
factor
  predict(fit.logistic,X)
}
est <- estimateBF(D.test)
```

Plot the Results

```
plot(fit.logistic)
```

```
plot(est)
```

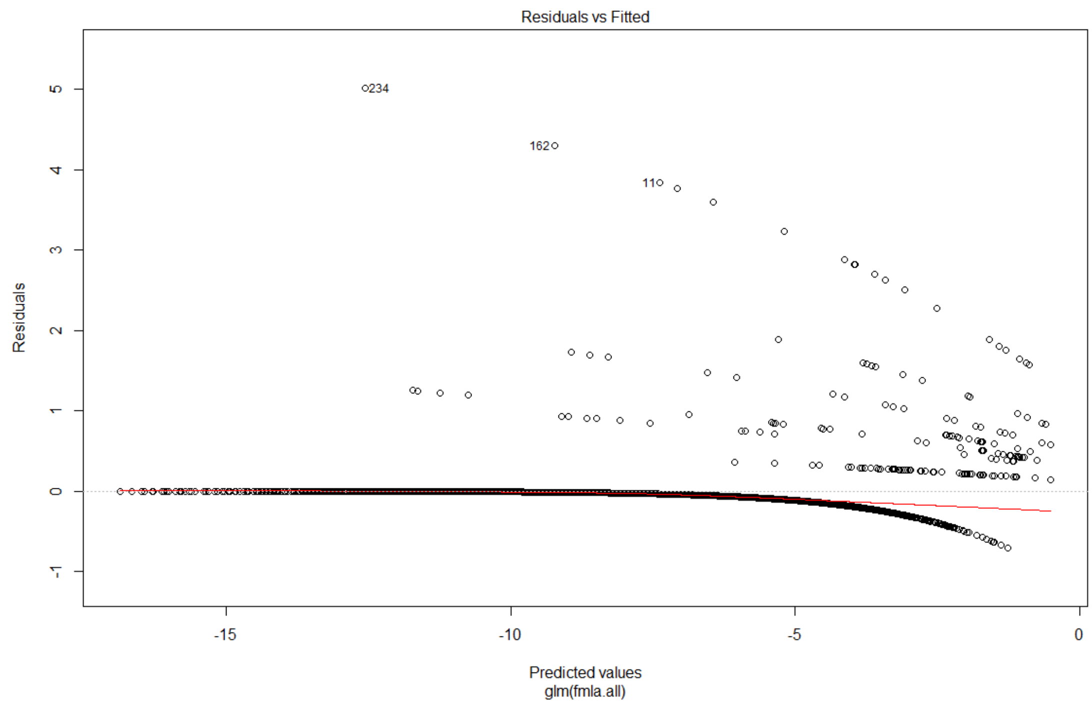

Figure 2a: This is the plot of residual versus fitted predicted values for the logistic regression model

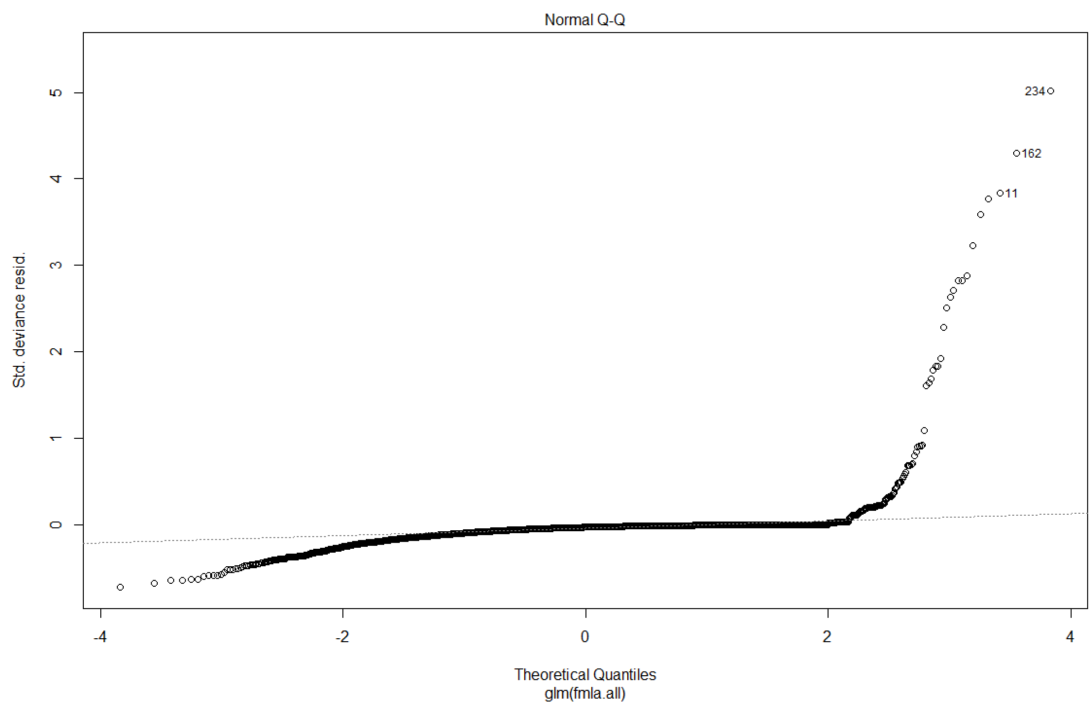

Figure 2a: This is the Normal QQ plot of theoretical quantiles for the logistic regression model

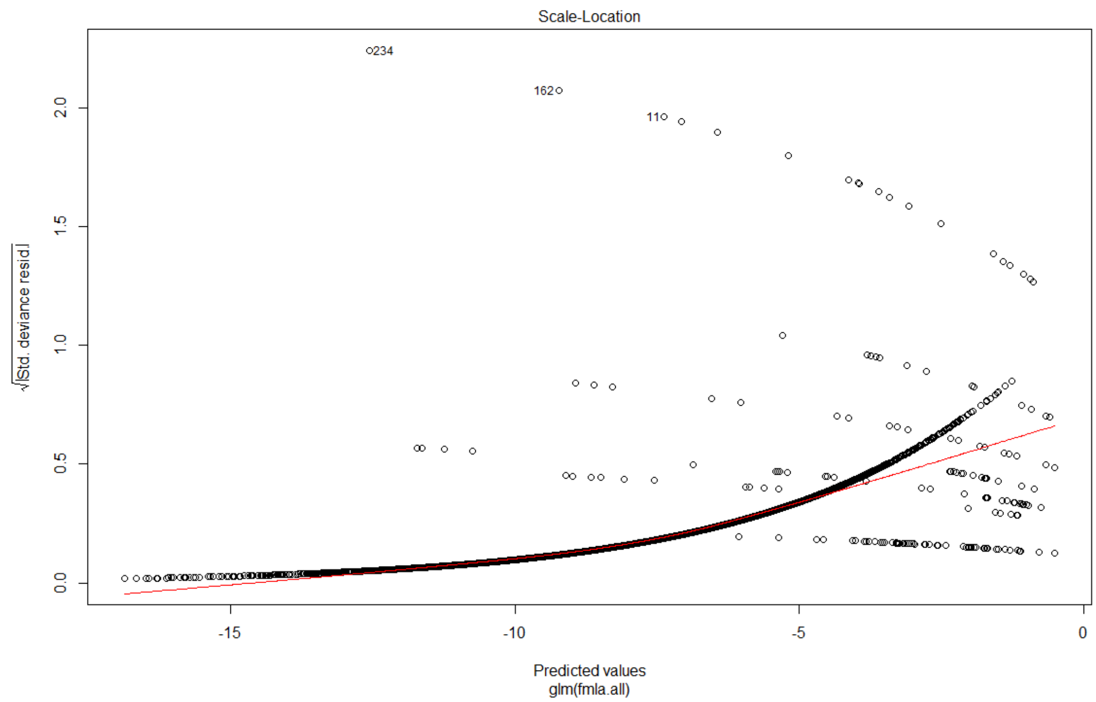

Figure 2c: This is the Scale-location plot of predicted values for the logistic regression model

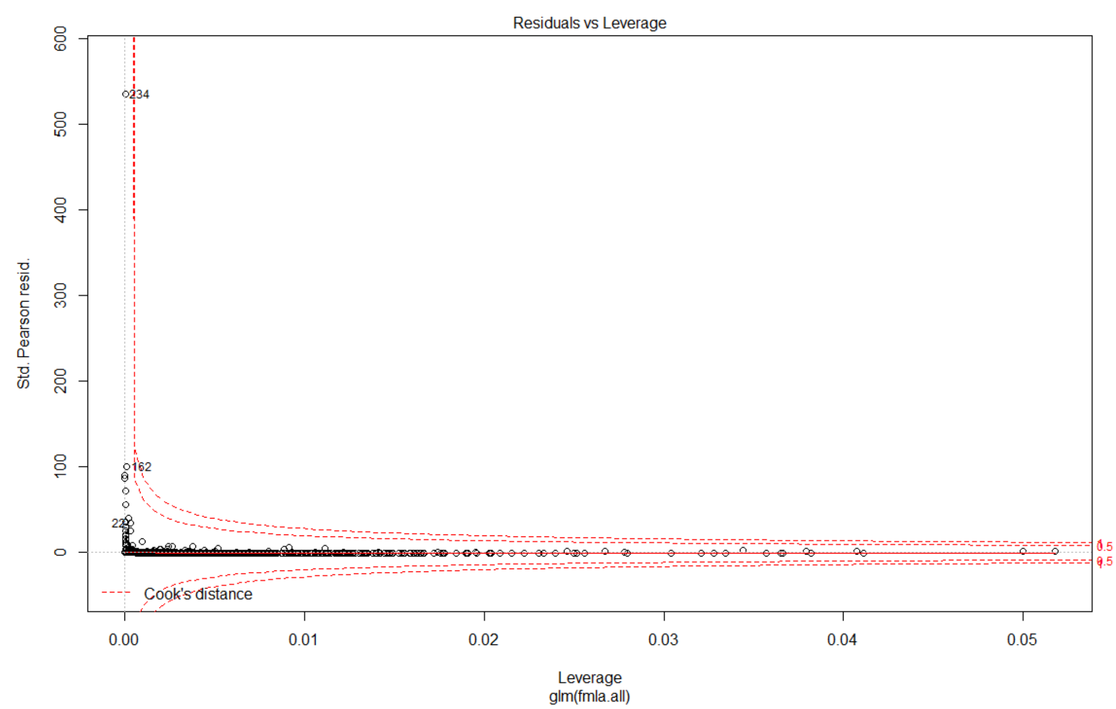

Figure 2d: This is the Residuals versus leverage plot for the logistic regression model

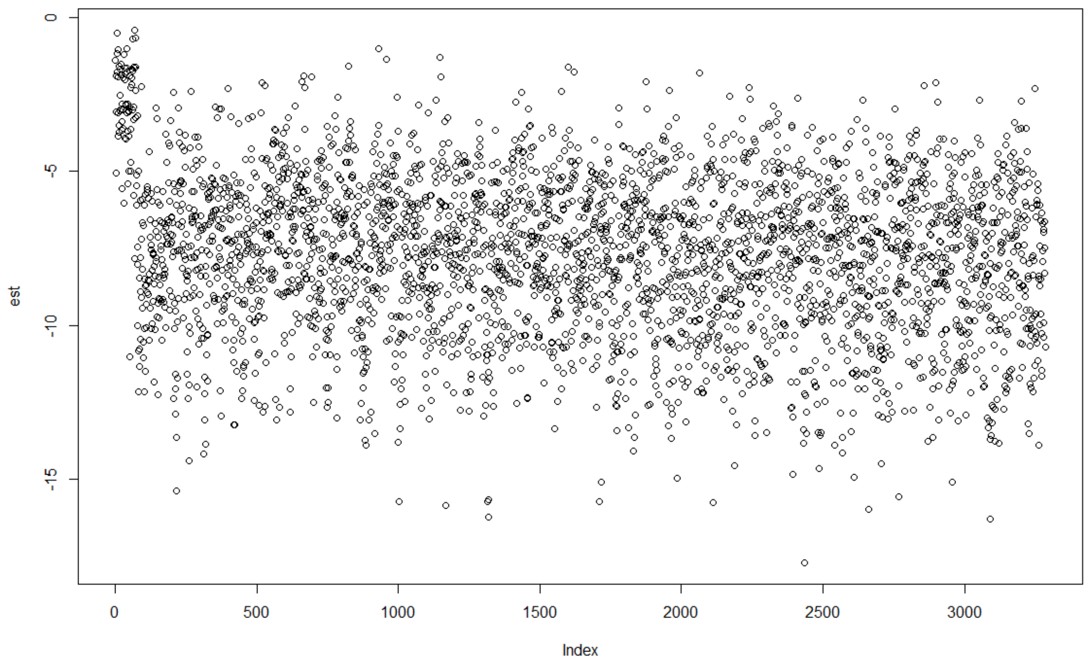

Figure 2e: This is the plot of estimated values for the logistic regression model

Examine the Factors

```
fit.logistic$R

##             (Intercept)   spatial   temporal          tod         dow
## (Intercept)    -5.55762 -14.04891 -256.02661 -26.226672781 -8.97039066
## spatial         0.00000  11.70624   11.51314   0.609282777  0.21928296
## temporal        0.00000   0.00000  255.52427  -0.004565678  0.08918358
## tod             0.00000   0.00000    0.00000  15.310088738  0.17031564
## dow             0.00000   0.00000    0.00000   0.000000000 -5.44614620
## MO11            0.00000   0.00000    0.00000   0.000000000  0.00000000
## MO21            0.00000   0.00000    0.00000   0.000000000  0.00000000
## MO31            0.00000   0.00000    0.00000   0.000000000  0.00000000
##                    MO11         MO21        MO31
## (Intercept) -3.00844646 -2.859329604 -1.46245700
## spatial     -0.39530917  0.008579316 -0.02068826
## temporal    -0.15428202  0.040069342 -0.04939764
## tod         -0.20798744  0.086261012  0.09196271
## dow          0.04021707 -0.110916385  0.06374548
## MO11        -2.72838851 -0.329903354 -0.24952803
## MO21         0.00000000 -2.754095684 -0.20432278
## MO31         0.00000000  0.000000000  2.42272556
```

Prediction using the Testing Data

```
pred <- predict.glm(fit.logistic, D.test)
summary(pred)
```

```
##      Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
##  -17.7122  -9.3407  -7.5364  -7.6950  -5.9541  -0.4169
```

Fit naive Bayes model and make estimateBF() function

```
vars = c("spatial","temporal","tod","dow","MO1","MO2","MO3")
fmla.all = as.formula(paste("Y ~ ", paste(vars, collapse= "+")))
fmla.all

## Y ~ spatial + temporal + tod + dow + MO1 + MO2 + MO3

NB =
naiveBayes(fmla.all,data=D.train,weights=weight,df=10,nbins=15,partition='qua
ntile')

estimateBF <- function(X){# estimateBF() returns the estimated log Bayes
factor
  predict(NB,newdata=X)
}
```

Plot Results

```
plot(NB)

plot(est)
```

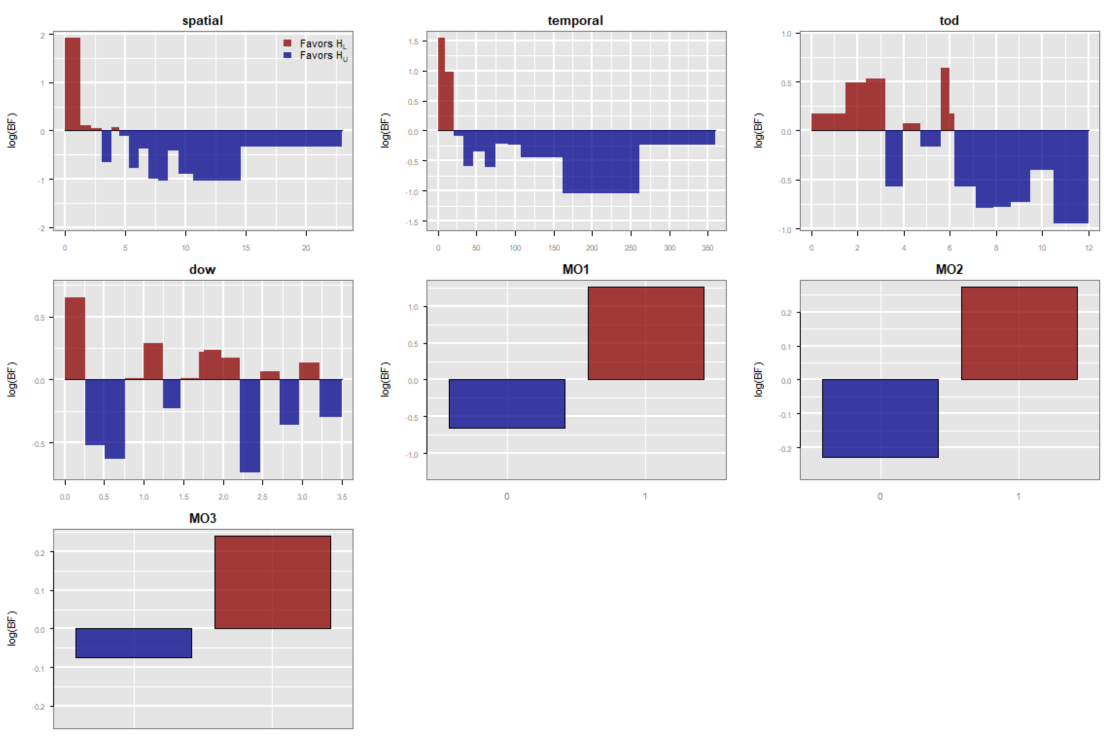

Figure 3a: The log-scale plots of factors for the naïve Bayes model

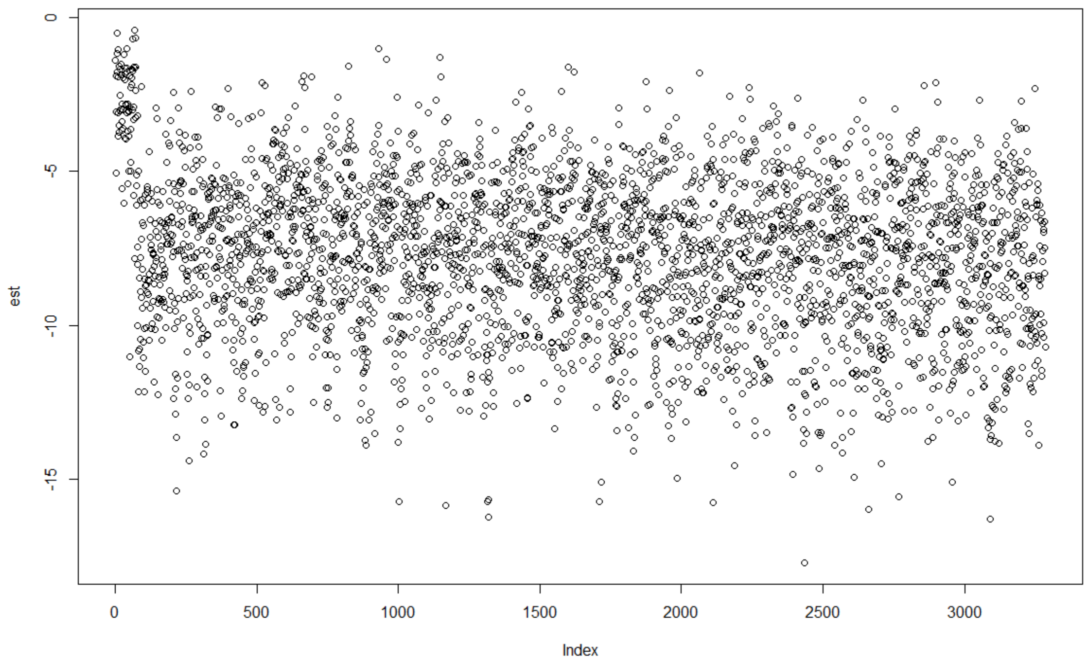

Figure 3b: The plots of estimated values for the naïve Bayes model

Print Factors

```
NB$spatial

##         from        to       value    N.linked N.unlinked    p.linked
## 1   0.000000  1.330002    [0,1.33] 18.92490842        408 0.39386082
## 2   1.330002  2.236050 (1.33,2.24]  2.58791209        515 0.07441269
## 3   2.236050  3.111460 (2.24,3.11]  2.35531136        519 0.06986449
## 4   3.111460  3.854694 (3.11,3.85]  0.57765568        522 0.03510481
## 5   3.854694  4.568613 (3.85,4.57]  2.47692308        516 0.07224245
## 6   4.568613  5.365327 (4.57,5.37]  1.87765568        520 0.06052457
## 7   5.365327  6.126928 (5.37,6.13]  0.35714286        521 0.03079298
## 8   6.126928  6.918310 (6.13,6.92]  1.16666667        525 0.04662213
## 9   6.918310  7.728588 (6.92,7.73]  0.04761905        526 0.02474065
## 10  7.728588  8.575494 (7.73,8.58]  0.00000000        526 0.02380952
## 11  8.575494  9.461998 (8.58,9.46]  1.04761905        525 0.04429431
## 12  9.461998 10.631034 (9.46,10.6]  0.19047619        523 0.02753403
## 13 10.631034 12.286147 (10.6,12.3]  0.00000000        526 0.02380952
## 14 12.286147 14.616999 (12.3,14.6]  0.00000000        527 0.02380952
## 15 14.616999 22.931787 (14.6,22.9]  1.26666667        522 0.04857750
##    p.unlinked        BF
## 1  0.05777996 6.8165648
## 2  0.06668887 1.1158187
## 3  0.06702191 1.0424127
```

```
## 4  0.06727170 0.5218362
## 5  0.06677213 1.0819251
## 6  0.06710517 0.9019360
## 7  0.06718843 0.4583077
## 8  0.06752148 0.6904785
## 9  0.06760474 0.3659603
## 10 0.06760474 0.3521872
## 11 0.06752148 0.6560033
## 12 0.06735496 0.4087900
## 13 0.06760474 0.3521872
## 14 0.06768800 0.3517540
## 15 0.06727170 0.7221090
```

```
NB$temporal
```

```
##            from         to         value   N.linked N.unlinked   p.linked
## 1    0.01052083   9.936227 [0.0105,9.94] 13.3608059        451 0.28506223
## 2    9.93622685  21.243403   (9.94,21.2]  7.5739927        486 0.17190883
## 3   21.24340278  33.065972   (21.2,33.1]  1.8457875        504 0.05990143
## 4   33.06597222  46.466667   (33.1,46.5]  0.6692308        515 0.03689544
## 5   46.46666667  60.875926   (46.5,60.9]  1.2172161        521 0.04761056
## 6   60.87592593  75.062500   (60.9,75.1]  0.6428571        518 0.03637974
## 7   75.06250000  90.860880   (75.1,90.9]  1.5666667        522 0.05444360
## 8   90.86087963 107.862384    (90.9,108]  1.5000000        524 0.05314002
## 9  107.86238426 124.468750     (108,124]  1.0000000        526 0.04336319
## 10 124.46875000 141.972454     (124,142]  1.0000000        525 0.04336319
## 11 141.97245370 162.146065     (142,162]  1.0000000        526 0.04336319
## 12 162.14606481 186.607639     (162,187]  0.0000000        527 0.02380952
## 13 186.60763889 218.381481     (187,218]  0.0000000        526 0.02380952
## 14 218.38148148 261.519676     (218,262]  0.0000000        527 0.02380952
## 15 261.51967593 359.565278     (262,360]  1.5000000        523 0.05314002
##    p.unlinked        BF
## 1  0.06136017 4.6457207
## 2  0.06427430 2.6746121
## 3  0.06577300 0.9107298
## 4  0.06668887 0.5532473
## 5  0.06718843 0.7086124
## 6  0.06693865 0.5434788
## 7  0.06727170 0.8093091
## 8  0.06743822 0.7879808
## 9  0.06760474 0.6414223
## 10 0.06752148 0.6422132
## 11 0.06760474 0.6414223
## 12 0.06768800 0.3517540
## 13 0.06760474 0.3521872
## 14 0.06768800 0.3517540
## 15 0.06735496 0.7889549
```

```
NB$MO1
```

```
##   value N.linked N.unlinked  p.linked p.unlinked        BF
## 1     0 14.37253       6482 0.4371664  0.8395286 0.5207285
## 2     1 18.50403       1239 0.5628336  0.1604714 3.5073751
```

```
NB$MO2$BF
```

```
## [1] 0.797155 1.312511
```

Agglomerative Hierarchical Crime Series Clustering

Hierarchical clustering is an algorithmic approach to crime series cluster analysis that sequentially forms a hierarchy of cluster solutions. The agglomerative approach starts with every observation (e.g., crime incident) in its own cluster. Then it sequentially merges the two closest clusters to form a new larger cluster. This process is repeated until all observations are in the same cluster or a stopping criterion is met.

This algorithm requires two similarity measures to be specified: the pairwise similarity between two observations and the similarity between two groups of observations. There are three primary approaches to measuring the similarity between groups of observations. \emph{Single linkage}, or nearest neighbor, uses the most similar pair between the two groups as the group similarity measure. In contrast, \emph{complete linkage} uses the least similar pair between two groups as the measure of group similarity. \emph{Average linkage} uses the average similarity between all pairs in the two groups.

The `crimelinkage` package provides the function `crimeClust_hier()` for agglomerative hierarchical crime clustering. It uses the log Bayes factor as the pairwise similarity measure. That is, the similarity between crimes i and j is $S(i,j) = \log BF(i,j)$, where $BF(i,j)$ is the estimated Bayes factor for linkage. Then one of: *average*, *single*, or *complete* linkage is used for the similarity between groups.

In this example, we will cluster all of the unsolved crimes from `crimes` data and display dendrogram of results with `plot_hcc()` function.

Get unsolved crimes

```
unsolved = subset(crimes, !crimeID %in% seriesData$crimeID)
```

Run agglomerative hierarchical crime clustering

```
tree = crimeClust_hier(unsolved,varlist,estimateBF,linkage='average',
binary=TRUE)
```

Plot results in dendrogram using plot_hcc()

```
plot_hcc(tree,yticks=seq(-2,6,by=2),type="triangle",hang=.05,main="Average
Linkage")
```

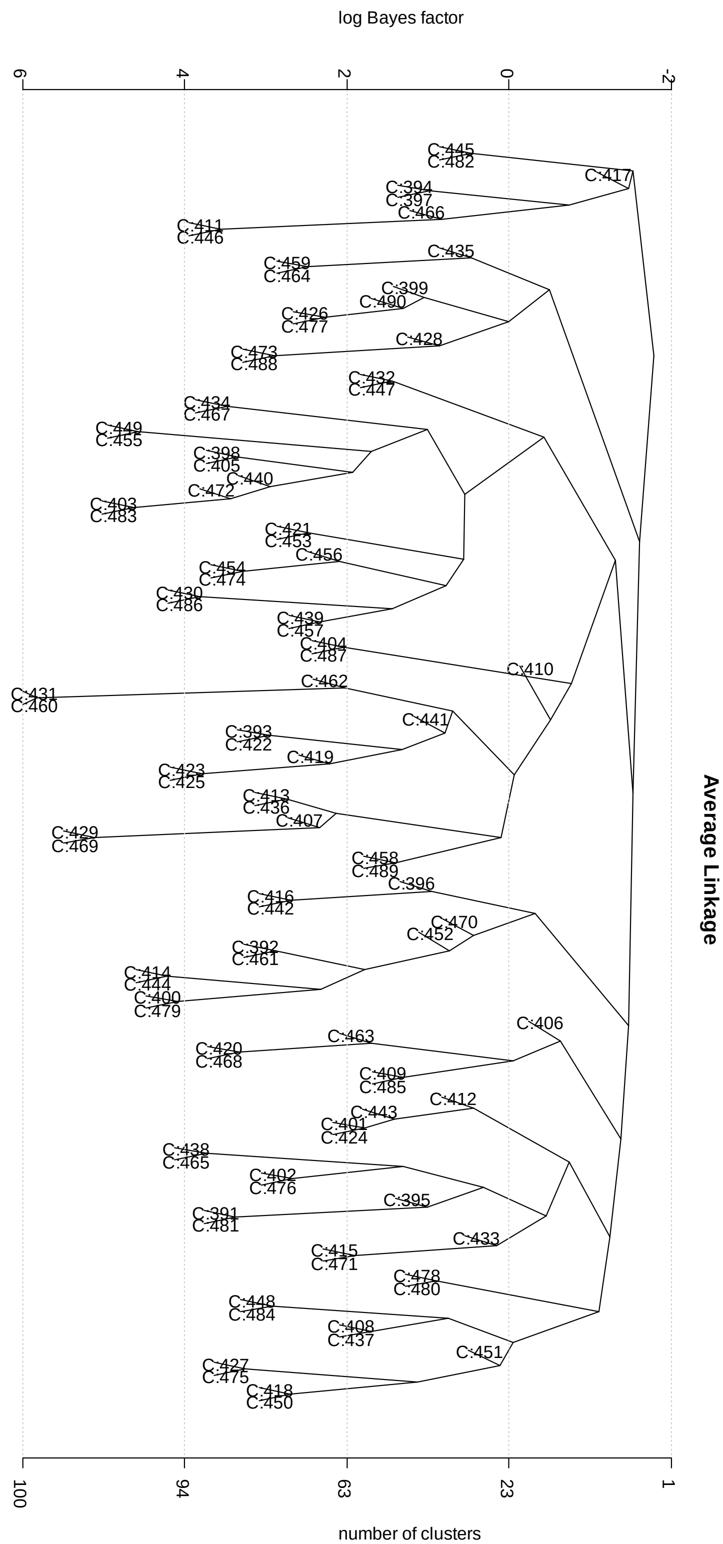

Figure 4: The dendrogram created using `plot_hcc()`. The scale of the bottom (or right if in landscape view) represents the number of clusters as you move left (or down). The top (or left scale) represents the log Bayes factors for the different clusters.

Examine crimes C:431 and C:460

```
subset(crimes,crimeID %in% c('C:431','C:460'))
```

```
##     crimeID      X        Y MO1 MO2 MO3             DT.FROM
## 431   C:431 7097.0 -10256.8  26   a   D 1993-10-23 22:00:00
## 460   C:460 7195.3 -10624.8  26   a   D 1993-11-07 03:10:00
##                  DT.TO
## 431 1993-10-24 06:00:00
## 460 1993-11-07 03:10:00
```

Find path info for crime C:429

```
cp = clusterPath('C:429',tree)
cp[cp$logBF>0,] # only return path for scores > 0
```

```
##       logBF       crimes
## 1 5.1074735        C:469
## 2 2.3400814        C:407
## 3 2.1364088 C:413, C:436
## 4 0.1009954 C:458, C:489
```

Hierarchical Based Crime Series Linkage

This approach to crime series identification compares an unsolved crime to every crime in criminal incident database and calculates its similarity as the log Bayes factor (according to the model developed for case linkage). Then it aggregates the similarity scores over the crime groups using single, complete, or average linkage. Single linkage uses the largest score (most similar crime) from each group, complete linkage uses the smallest score (least similar crime) from each group, and average linkage uses the average score as the group score.

Example

To give an example, extract the solved and unsolved crimes from the crimes data.

```
solved = subset(crimes, crimeID %in% seriesData$crimeID)
unsolved = subset(crimes, !crimeID %in% seriesData$crimeID)
```

The function seriesID() can be used to find the most similar crime series to the unsolved crime.

```
crime = unsolved[2,]  # use the 2nd unsolved crime C:392
crime
```

```
##     crimeID       X       Y MO1 MO2 MO3             DT.FROM
## 392   C:392 12793.2 -3386.5  25   a   E 1993-06-19 07:00:00
##                  DT.TO
## 392 1993-06-19 07:00:00
```

```
results = seriesID(crime,solved,seriesData,varlist,estimateBF)
head(results$score)
```

```
##   group  average   single complete
## 1    12 3.739680 3.739680 3.739680
## 2   154 3.637851 3.637851 3.637851
## 3   160 3.617055 3.617055 3.617055
## 4     8 3.494182 3.494182 3.494182
## 5     9 3.224453 3.924757 2.760956
## 6    10 3.224453 3.924757 2.760956
```

This shows that the unsolved crime is most similar to the crime(s) in crime group 12 with an average linkage log Bayes factor of 4.06. To get the crimes and offenders associated with these groups, just use the subset() function with the groups object:

```
subset(results$groups,group=='12')      # most similar crime series

##    crimeID Index CS offenderID                TIME group
## 27   C:148   148 12      O:109 1993-06-25 01:20:00    12

subset(results$groups,group=='154')     # 2nd most similar series

##     crimeID Index  CS offenderID                TIME group
## 205   C:304   304 154      O:237 1993-10-25 07:00:00   154

subset(results$groups,group=='9')       # a series with multiple crimes

##    crimeID Index CS offenderID                TIME group
## 9    C:144   144  9      O:106 1993-06-20 03:27:00     9
## 10   C:163   163  9      O:106 1993-06-20 13:15:00     9
## 11   C:145   145  9      O:106 1993-06-20 01:30:00     9
## 12   C:164   164  9      O:106 1993-06-20 13:15:00     9
## 13   C:165   165  9      O:106 1993-06-20 12:45:00     9
## 14   C:166   166  9      O:106 1993-06-20 12:45:00     9
```

We can do this for another unsolved crime

```
crime4 = unsolved[4,] # use the 4th unsolved crime
results4 = seriesID(crime4,solved,seriesData,varlist,estimateBF)
head(results4$score)

##   group   average    single  complete
## 1   136 1.5283543 1.5283543 1.5283543
## 2   316 1.1363833 1.1363833 1.1363833
## 3    37 0.8748010 0.8748010 0.8748010
## 4    48 0.8748010 0.8748010 0.8748010
## 5   206 0.5522861 0.5522861 0.5522861
## 6   219 0.4631613 0.4631613 0.4631613
```

Because the scores are so low (log Bayes factors around 1), this unsolved crime is not very similar to any other solved crimes in the crime database. Perhaps this is the start of a new crime series? It is also possible to compare a crime to all unsolved crimes to detect potential unsolved crime series.

Using Crime C:394 (the 4th unsolved crime)

```
pairs = data.frame(i1=unsolved$crimeID[4],i2=unique(unsolved$crimeID[-4]))
X = compareCrimes(pairs,unsolved,varlist,binary=TRUE)     # Evidence data
score = data.frame(pairs,logBF=estimateBF(X))
head(score[order(-score$logBF),])
```

```
##       i1    i2         logBF
## 6  C:394 C:397  0.980062532
## 79 C:394 C:470  0.752385770
## 5  C:394 C:396  0.339529086
## 60 C:394 C:451 -0.007776621
## 69 C:394 C:460 -0.186420669
## 47 C:394 C:438 -0.206162572
```

There are no unsolved crimes that are very similar to this one - probably not enough evidence to link this crime to any others. This approach also gives similar results to what was obtained from the hierarchical clustering path approach:

```
C429 = which(unsolved$crimeID %in% 'C:429')       # now use crime C:429
pairs = data.frame(i1=unsolved$crimeID[C429],i2=unique(unsolved$crimeID[-
C429]))
X = compareCrimes(pairs,unsolved,varlist,binary=TRUE)     # Evidence data
score = data.frame(pairs,logBF=estimateBF(X))
head(score[order(-score$logBF),])
```

```
##       i1    i2    logBF
## 78 C:429 C:469 5.107474
## 17 C:429 C:407 2.735772
## 23 C:429 C:413 2.371051
## 45 C:429 C:436 2.287908
## 13 C:429 C:403 1.305843
## 3  C:429 C:393 1.261425
```

Results from hierarchical clustering

```
cp = clusterPath('C:429',tree)
cp[cp$logBF>0,]
```

```
##       logBF       crimes
## 1 5.1074735        C:469
## 2 2.3400814        C:407
## 3 2.1364088 C:413, C:436
## 4 0.1009954 C:458, C:489
```

The function crimeClust_bayes() is used for the Bayesian model-based clustering approach. Because it uses both solved and unsolved crimes, the labels (crime group) for the solved crimes is also passed into the function. (Note: this function will take 20+ minutes to run.)

Bayesian Model-Based Approaches

This section illustrates the partially-supervised Bayesian model-based clustering approach to crime series linkage of [@CrimeClust]. This approach is partially-supervised because the offender is known for a subset of the events and utilizes spatiotemporal crime locations as well as crime features describing the offender\'s modus operandi.

The hierarchical model naturally handles complex features often seen in crime data, including missing data, interval censored event times, and a mix of discrete and continuous variables. It can also provide uncertainty assessments for all model parameters, including the relative influence of each feature (space, time, method of entry, etc.) in the model. In addition, the model produces posterior clustering probabilities which allow analysts to act on model output only as warranted.

The function `crimeClust_bayes()` is used for the Bayesian model-based clustering approach. Because it uses both solved and unsolved crimes, the labels (crime group) for the solved crimes is also passed into the function. (Note: this function will take 20+ minutes to run.)

Make the crime group labels for each crime (NA for unsolved crimes)

```
seriesData$CG = makeGroups(seriesData,method=2)      # method=2 uses unique
co-offenders
group_info = unique(seriesData[,c('crimeID','CG')])  # extract the group info
A = merge(crimes,group_info,by="crimeID",all.x=TRUE) # add group info to
crimes
A = A[order(A$CG),]                                  # order by crime group
```

Run MCMC

```
fit = crimeClust_bayes(A$CG, spatial=A[,c('X','Y')],t1=A$DT.FROM,t2=A$DT.TO,
                       Xcat=A[,c("MO1","MO2","MO3")],maxcriminals=1000,
                       iters=2000,burn=500,update=100,seed=5)
```

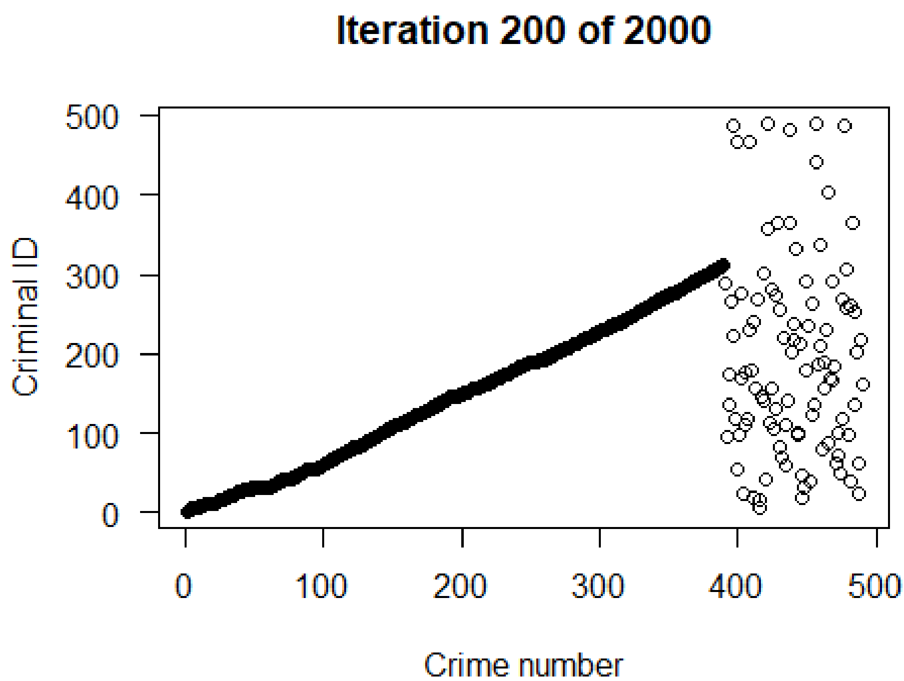

Figure 5a: This is one of second solution plots from 2000 iterations of the `crimeClust_bayes()` model, recalling that we update every 100 iterations.

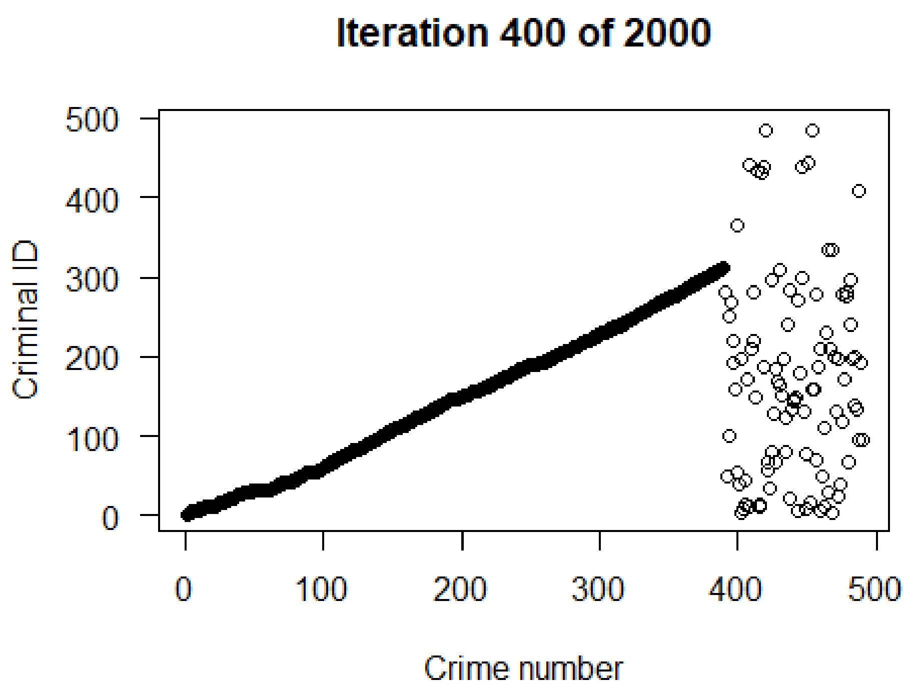

Figure 5b: This is one of fourth solution plots from 2000 iterations of the `crimeClust_bayes()` model, recalling that we update every 100 iterations.

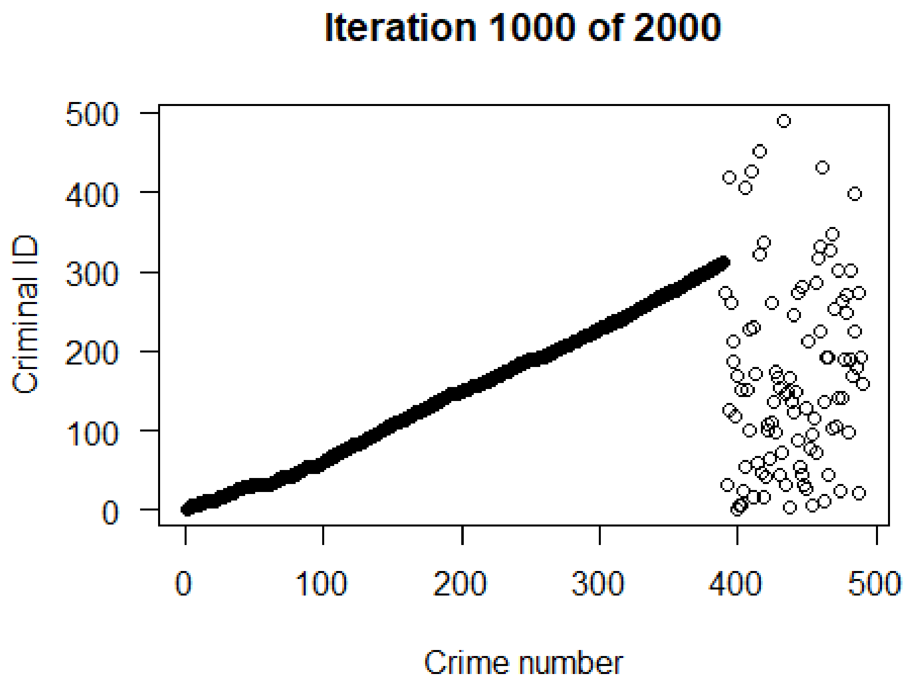

Figure 5c: This is one of tenth solution plots from 2000 iterations of the `crimeClust_bayes()` model, recalling that we update every 100 iterations.

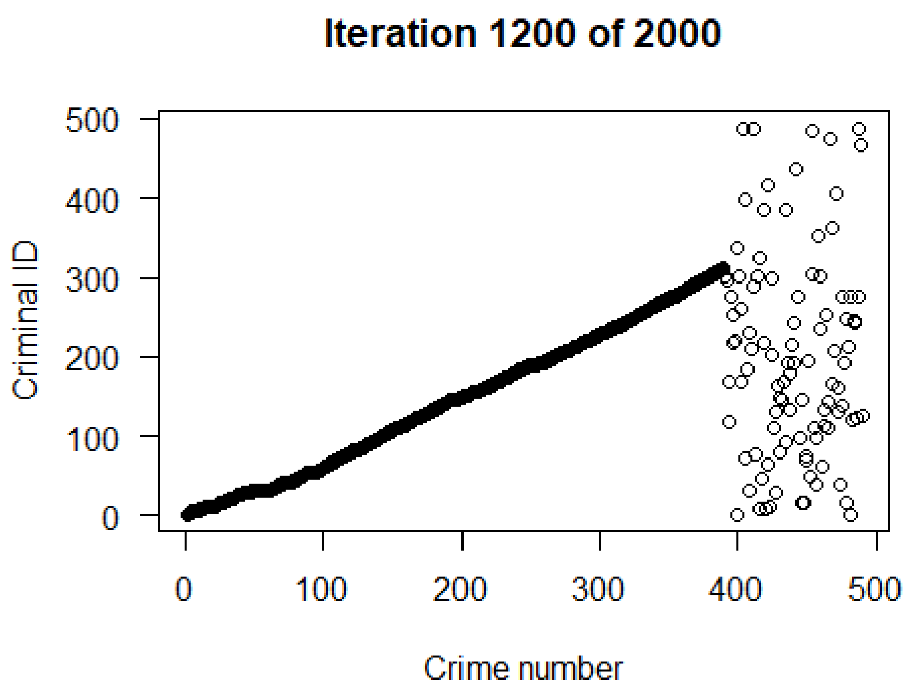

Figure 5d: This is one of twelfth solution plots from 2000 iterations of the `crimeClust_bayes()` model, recalling that we update every 100 iterations.

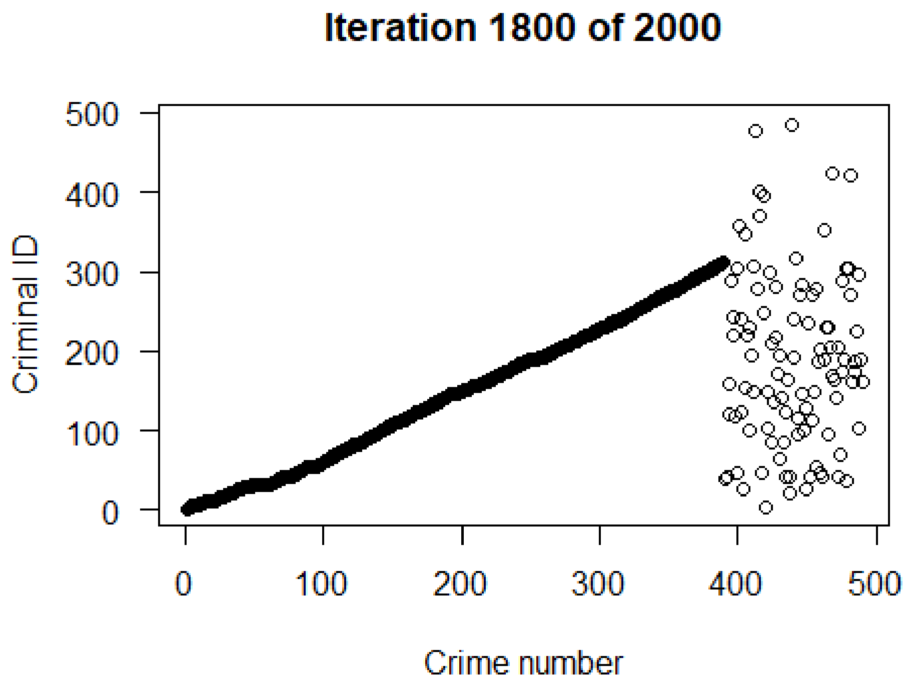

Figure 5e: This is one of eighteenth solution plots from 2000 iterations of the `crimeClust_bayes()` model, recalling that we update every 100 iterations.

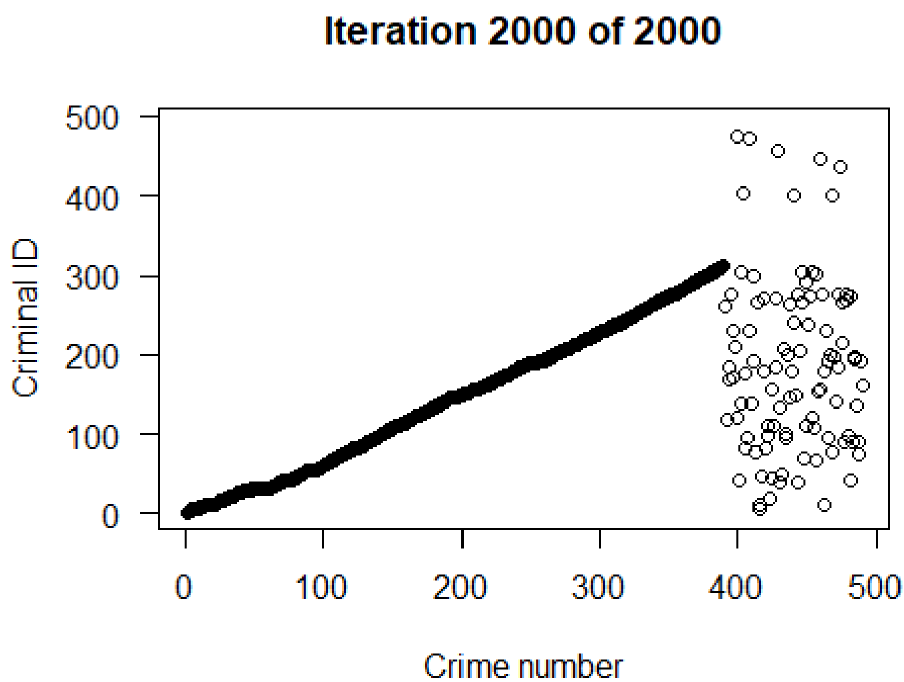

Figure 5f: This is one of twentieth solution plots from 2000 iterations of the `crimeClust_bayes()` model, recalling that we update every 100 iterations.

```
summary(fit)
```

```
##             Length Class  Mode
## p.equal    240100 -none- numeric
## D            2000 -none- numeric
## df           6000 -none- numeric
## sd1          2000 -none- numeric
## sd2          2000 -none- numeric
## sds          4000 -none- numeric
## theta        2000 -none- numeric
## s.miss          0 -none- NULL
## t.censored      0 -none- NULL
## missing_s       0 -none- numeric
## missing_t     319 -none- numeric
## crimeID       490 -none- numeric
```

Extract pairwise probabilities

```
pp = fit$p.equal  # probability that crime i is linked to crime j
diag(pp) = NA
summary(pp)
```

```
##        V1                   V2                   V3
##  Min.   :0.0000000   Min.   :0.0000000   Min.   :0.0000000
##  1st Qu.:0.0000000   1st Qu.:0.0000000   1st Qu.:0.0000000
##  Median :0.0000000   Median :0.0000000   Median :0.0000000
##  Mean   :0.0003476   Mean   :0.0003476   Mean   :0.0003545
##  3rd Qu.:0.0000000   3rd Qu.:0.0000000   3rd Qu.:0.0000000
##  Max.   :0.0766667   Max.   :0.0926667   Max.   :0.0340000
##  NA's   :1           NA's   :1           NA's   :1
##        V4                   V5                  V6
##  Min.   :0.0000000   Min.   :0.000000   Min.   :0.000000
##  1st Qu.:0.0000000   1st Qu.:0.000000   1st Qu.:0.000000
##  Median :0.0000000   Median :0.000000   Median :0.000000
##  Mean   :0.0006885   Mean   :0.001114   Mean   :0.006849
##  3rd Qu.:0.0000000   3rd Qu.:0.000000   3rd Qu.:0.000000
##  Max.   :0.0640000   Max.   :0.052000   Max.   :1.000000
##  NA's   :1           NA's   :1          NA's   :1
##  .
##  .
##  .
##       V488                V489                V490
##  Min.   :0.000000   Min.   :0.000000   Min.   :0.000000
##  1st Qu.:0.000000   1st Qu.:0.000000   1st Qu.:0.000000
##  Median :0.000000   Median :0.000000   Median :0.000000
##  Mean   :0.003264   Mean   :0.003847   Mean   :0.002363
##  3rd Qu.:0.002000   3rd Qu.:0.000000   3rd Qu.:0.000000
##  Max.   :0.034000   Max.   :0.101333   Max.   :0.131333
##  NA's   :1          NA's   :1          NA's   :1
```

The matrix pp contains the pairwise estimated probability that two crime are linked (share a common offender). We can use this information for crime series identification. Using the image.plot() from the fields package, we can see how strongly the unsolved crimes are linked to the existing (solved) crime series.

```
library(fields) # if not installed, type: install.packages("fields")
```

Get index of unsolved crimes

```
ind.unsolved = which(is.na(A$CG)) # index of unsolved crimes
n = nrow(A) # number of crimes
```

Image plot of linkage probabilities

```
fields::image.plot(1:n,ind.unsolved,pp[1:n,ind.unsolved],
                   xlab="Crime",ylab="Unsolved Crime",
                   main="Probability crimes are linked")
```

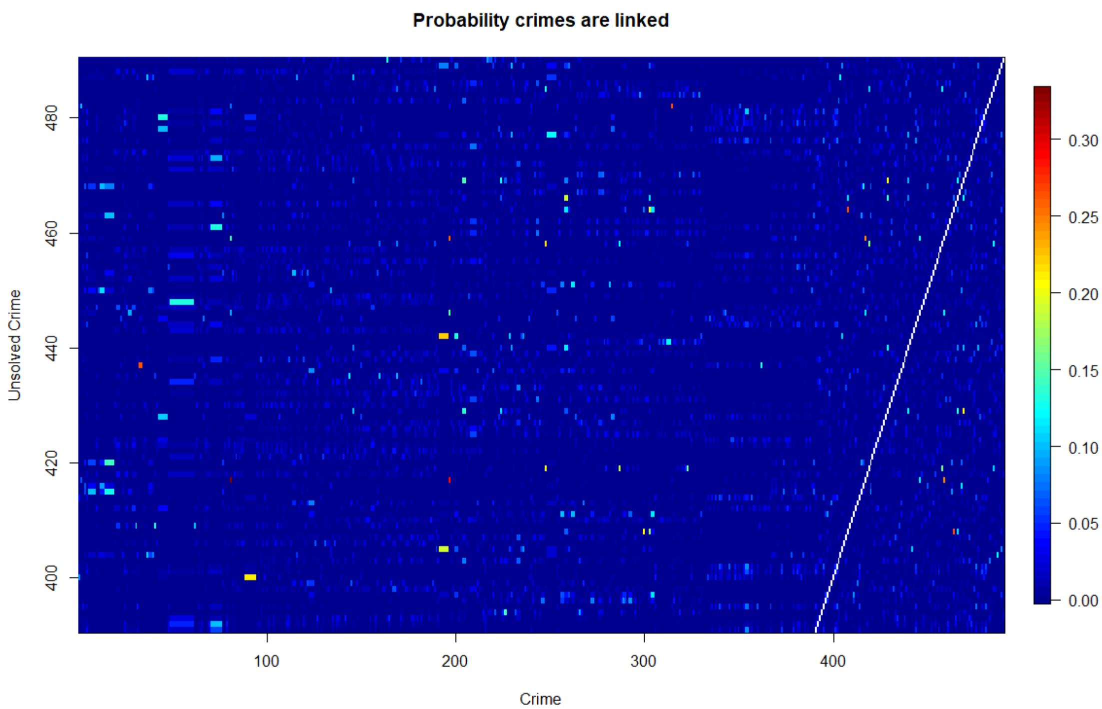

Figure 6: This plot shows the probability that unsolved crimes are related to crimes that are solved and clustered with. For most of the crimes shown, there is very little chance they are linked.

We see that some unsolved crimes are linked to solved crimes with a posterior probability above 0.25. These crimes may be worth further investigations. Here we plot the maximum posterior probability that an unsolved crime is linked to another crime (solved or unsolved).

Find strongest linkages

```
unsolved.probs = apply(pp[ind.unsolved,],1,max,na.rm=TRUE)  # maximum
probability
plot(ind.unsolved,unsolved.probs,xlab="unsolved crime",ylab='maximum
```

```
probability of linkage')
abline(h=0.25)
```

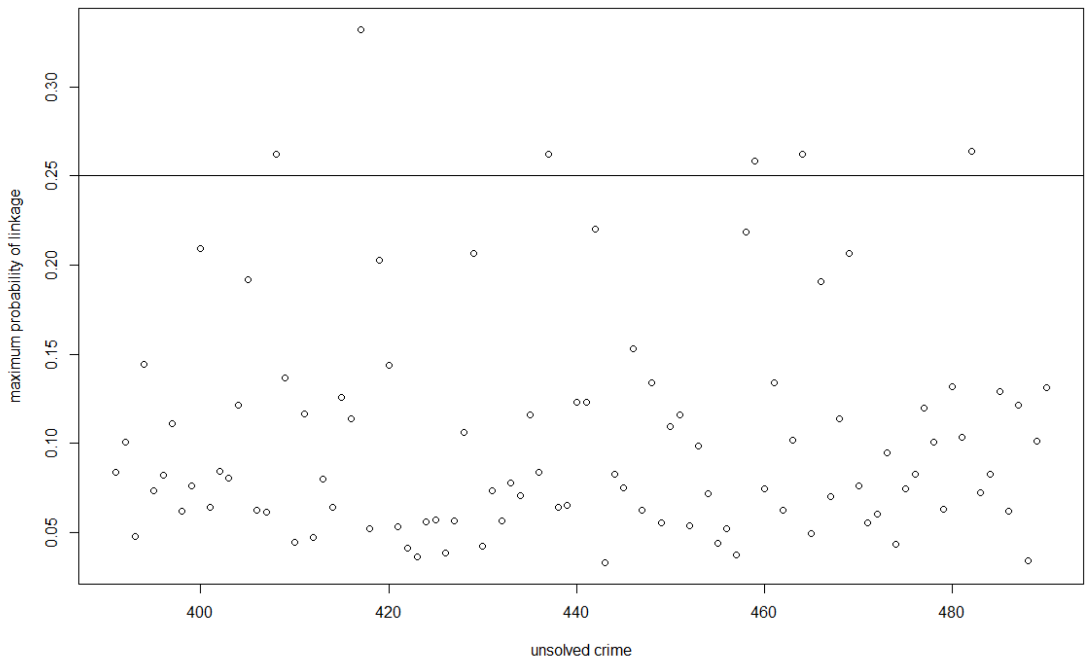

Figure 7: The plot also shows the probability that unsolved crimes are related to crimes that are solved and clustered with. There are only five unsolved crime that have a better than 25 percent chance of linkage.

```
ind = ind.unsolved[unsolved.probs > 0.25]
investigate = as.character(A$crimeID[ind])  # crimeIDs for crimes with
strongest linkage
investigate

## [1] "C:408" "C:417" "C:437" "C:459" "C:464" "C:482"
```

This shows that C:408, C:417, C:437, C:459, C:464, and C:482 are the crimes with the strongest linkages (posterior probabilities greater that 0.25). A particular crime can be investigated in more detail with the function bayesProb():

```
bp = bayesProb(pp[A$crimeID %in% "C:417"])
bp$crimeID = A$crimeID[bp$index]
bp$CG = A$CG[bp$index]
head(bp)

##   index       prob crimeID  CG
## 1    81 0.33200000    C:15  46
## 2   197 0.28800000    C:26 146
## 3   459 0.24400000   C:459  NA
## 4   446 0.11400000   C:446  NA
## 5     2 0.06066667    C:10   2
## 6   482 0.04066667   C:482  NA
```

For this example, our model provides a list of the most likely crimes associated with the unsolved crime C:417. The first two crimes (C:15 and C:26) are solved crimes indicating that the offender(s) responsible for these crimes may also be responsible for C:417. The next two crimes, C:459 and C:446 do not have a group ID. This means that they are unsolved crimes. By providing the posterior probabilities, crime analysts may choose to investigate further only if the linkage is strong enough.