Development and Optimization of Machine Learning Algorithms and Models of Relevance to START Databases

Report to the Office of University Programs, Science and Technology Directorate, U.S. Department of Homeland Security

April 2016

National Consortium for the Study of Terrorism and Responses to Terrorism
A Department of Homeland Security Science and Technology Center of Excellence
Led by the University of Maryland

8400 Baltimore Ave., Suite 250 • College Park, MD 20742 • 301.405.6600
www.start.umd.edu

About This Report

The authors of this report are Prabhakar Misra, Raul Garcia-Sanchez and Daniel Casimir, Department of Physics & Astronomy, Howard University. Questions about this report should be directed to Prabhakar Misra at pmisra@howard.edu.

This research was supported by the Department of Homeland Security Science and Technology Directorate's Office of University Programs through Award Number DHS50000, through the DHS Research Team Follow-On Funding Program for Minority Serving Institutions. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the U.S. Department of Homeland Security or START.

About START

Citations

To cite this report, please use this format:

Prabhakar Misra, Raul Garcia-Sanchez and Daniel Casimir. "Development and Optimization of Machine Learning Algorithms and Models of Relevance to START Databases," Report to the Office of University Programs, Science and Technology Directorate, U.S. Department of Homeland Security. College Park, MD: START, 2016.

CONTENTS

GENERAL OVERVIEW

The goal of this project has been to develop an algorithm for finding patterns in terrorism-related databases developed by the National Consortium for the Study of Terrorism and Responses to Terrorism (START), the DHS Center of Excellence at the University of Maryland. The present work expands on the research carried out during the Summer of 2014 [1], with the goal of further developing means to address the missing data problems in the START databases using pattern recognition. In addition, we are working on determining ways to significantly speed up the neural network training process, which can take a fairly long time under our original implementation. While we focused our efforts on specific datasets, this time around we generalized our code so that it could be implemented using any START database.

In addition to the work on improving our neural network approach, we also focused on developing a logistic regression algorithm - based on Dr. Ronald Breiger's previous R-script based work [2] – but now employing Matlab, which served to complement our neural network methodology, as we explored innovative ways to decrease the neural network's misclassification errors.

We worked closely with the START researchers and management team every step of the way, making four presentations relating to our research progress at regular intervals that facilitated feedback and allowed us to be on track. Success will be measured based on accomplishing the goals the START team would like to be completed and meeting satisfactorily the scientific goals laid out in the follow-on funding proposal.

KEY PARTNERSHIPS

- Our research collaborator at START is Dr. Gary Ackerman, Director of the Unconventional Weapons and Technology Division, along with Mr. Markus Binder, Project Manager/Researcher, who provided guidance and feedback on an ongoing basis. Mr. Eamon Diab, IT Program Manager, at START was our point person who facilitated the utilization of computational resources at the University of Maryland for the collaborative research effort. We also interacted with additional START staff, in particular Mr. Patrick James, Project Manager/Researcher, in order to look into the missing data problem in another important START Database, Profiles of Individual Radicalization in the United States (PIRUS), and Mr. John Sawyer, Commercialization Director, who was a sounding board for regular discussions about our neural network methodology as applied to START databases. Besides the START personnel mentioned above, the support from senior management, Mr. William Braniff, Executive Director, and Dr. Gary LaFree, Director of START, was exceptional in providing us with administrative and IT resources.

- We also collaborated with Dr. Ronald Breiger of the University of Arizona on implementing R-scripts for the Profiles of Incidents Involving CBRN by Non-state Actors (POICN) database and adapted it to Matlab.

- We have had in-house discussions with Mr. Brandon Behlendorf at START regarding external proposal development and submissions to the National Science Foundation and the Department of Defense. We gave a presentation of our research on the current project to Dr. Paul Tandy of the Defense Threat Reduction Agency (DTRA) on 10/13/15 in an effort to explore potential funding opportunities that may arise in the near future. We have also been in communication with Professor Claudio Cioffi-Revilla of George Mason University regarding participation in future conferences that he organizes, as well as other symposia related to large databases and smart algorithm development that we can participate in.

ACTIVITIES

The primary focus for the year-long machine learning research project centered on the following research activities:

1. Expansion to other START databases (POICN and PIRUS), in addition to the Global Terrorism Database (GTD).
2. Dealing with missing or incomplete data
3. Improving pattern recognition in START databases
4. Logistic regression and neural networks comparison study
5. Data compression

PATTERN RECOGNITION AND MISSING DATA METHODOLOGY

EXPANSION TO OTHER START DATABASES

Script Structure

The flow diagram for the typical neural network dataset breakdown is shown in Figure 1. This shows how the data that we use for our neural network is split into a training test, a test set, and a prediction set.

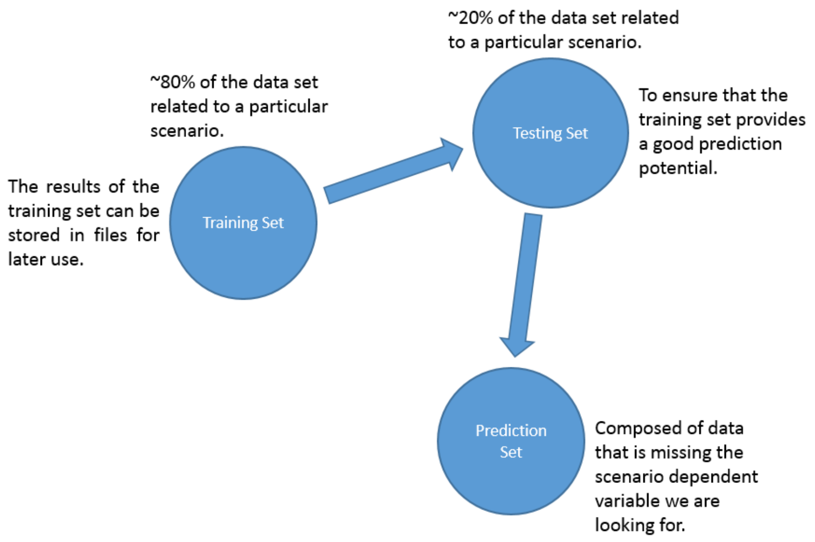

~20% of the data set related to a particular scenario.
~80% of the data set related to a particular scenario.
Testing Set
To ensure that the training set provides a good prediction potential.
The results of the training set can be stored in files for later use.
Training Set
Prediction Set
Composed of data that is missing the scenario dependent variable we are looking for.

Figure 1. The Neural Network Dataset Structural Flow Diagram.

The current script structure is shown in Figure 2.

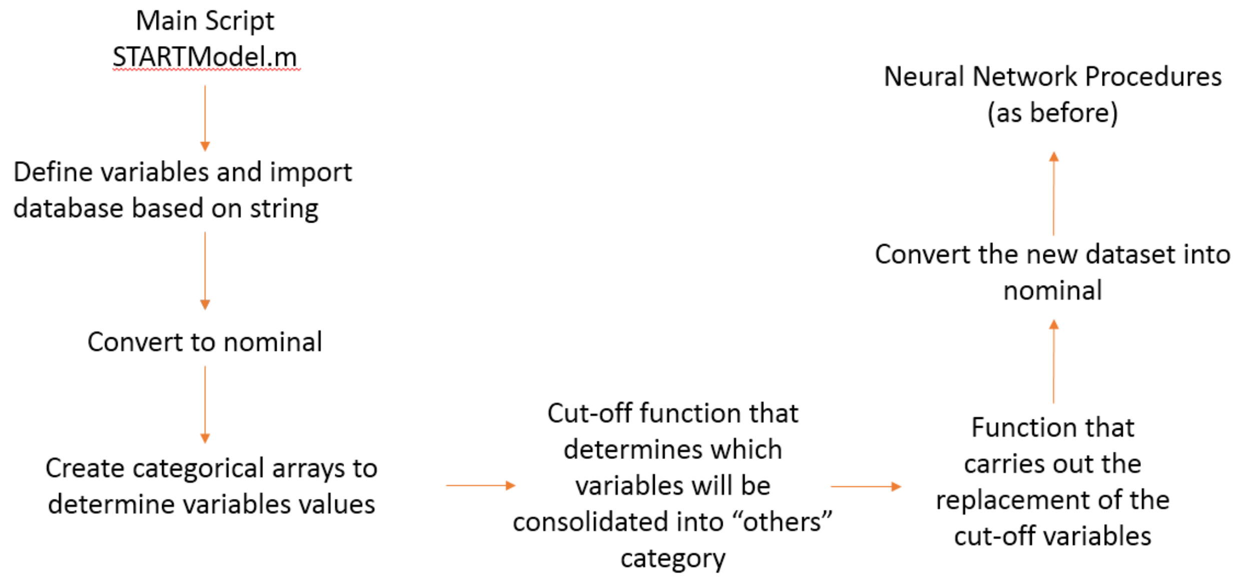

Main Script
STARTModel.m
Define variables and import database based on string
Convert to nominal
Create categorical arrays to determine variables values
Cut-off function that determines which variables will be consolidated into "others" category
Function that carries out the replacement of the cut-off variables
Convert the new dataset into nominal
Neural Network Procedures (as before)

Figure 2. Current Script Structure Flow Diagram.

Generalization

During the 2014 Summer Research Team (SRT) effort, we focused mainly on the GTD database. In the current follow-on project effort, we have successfully modified the machine learning code, so that the Matlab scripts can now import all START-formatted databases directly from their Excel spreadsheet files. The key changes to the scripts are explained below.

Import Function

- Added switch statement that determines which database will be imported into the run.
- Removed formatting issues in databases in which NaN entries would show up long past the actual scope of the database variables.
- Added header extraction statements to ensure that all variable names in START databases format properly under the Matlab environment.
- Added switch statement that selects which variables from the chosen database will be imported into Matlab, also allowing us to import the full contents of the database, if necessary.
- Added code to consolidate all iterations of an "Unknown" variable, and consolidated into one single value.

Main Script

- Added switch statement for calling import function based on the user's chosen database.

These features are the first step towards making the user-interface, once the Matlab code has been converted to a programming language using the Matlab Coder.

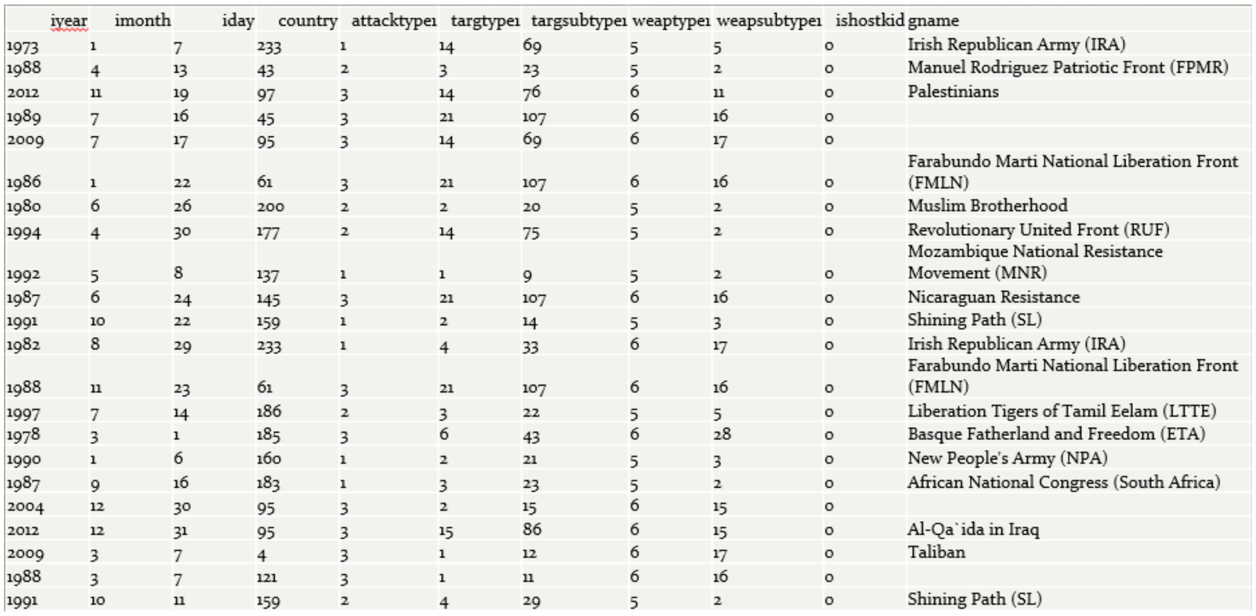
Global Terrorism Database (GTD) Updated Results

For GTD, the subset of data consisted of:

- **Inputs:** iyear, imonth, iday, country, attacktype1, targtype1, targsubtype1, weaptype1, weapsubtype1, ishostkid
- **Target:** gname

The input variables represent the date, the country in which the attack occurred, what type of attack it was, what was the target, the type of weapon used in the attack and whether or not the situation involved hostage kidnapping. These input variables were used to find the perpetrator (group name) of the attack.

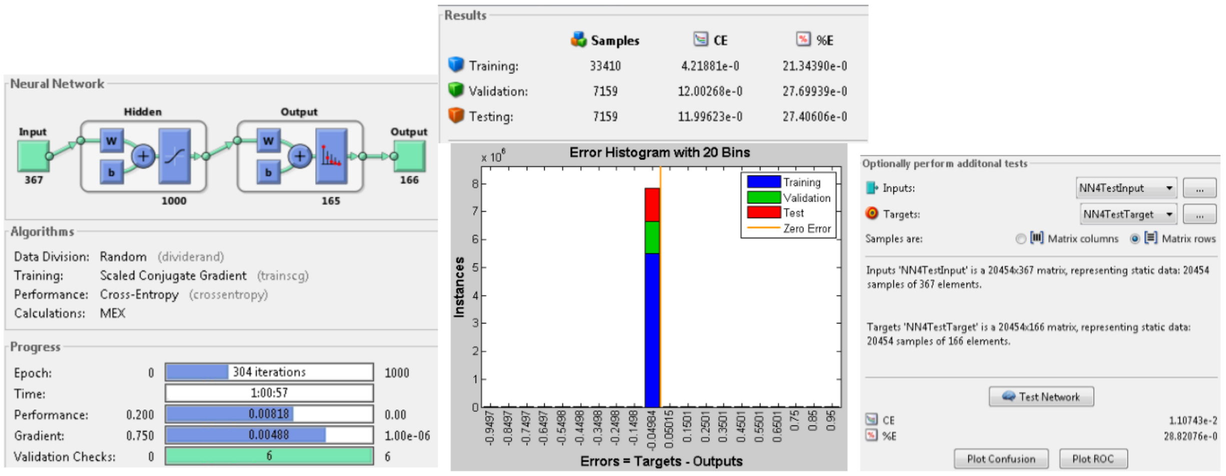
Figure 3 shows a snippet of the database subset that we used for GTD neural network analysis.



iyear	imonth	iday	country	attacktype1	targtype1	targsubtype1	weaptype1	weapsubtype1	ishostkid	gname
1973	1	7	233	1	14	69	5	5	0	Irish Republican Army (IRA)
1988	4	13	43	2	3	23	5	2	0	Manuel Rodriguez Patriotic Front (FPMR)
2012	11	19	97	3	14	76	6	11	0	Palestinians
1989	7	16	45	3	21	107	6	16	0	
2009	7	17	95	3	14	69	6	17	0	
1986	1	22	61	3	21	107	6	16	0	Farabundo Marti National Liberation Front (FMLN)
1980	6	26	200	2	2	20	5	2	0	Muslim Brotherhood
1994	4	30	177	2	14	75	5	2	0	Revolutionary United Front (RUF)
1992	5	8	137	1	1	9	5	2	0	Mozambique National Resistance Movement (MNR)
1987	6	24	145	3	21	107	6	16	0	Nicaraguan Resistance
1991	10	22	159	1	2	14	5	3	0	Shining Path (SL)
1982	8	29	233	1	4	33	6	17	0	Irish Republican Army (IRA)
1988	11	23	61	3	21	107	6	16	0	Farabundo Marti National Liberation Front (FMLN)
1997	7	14	186	2	3	22	5	5	0	Liberation Tigers of Tamil Eelam (LTTE)
1978	3	1	185	3	6	43	6	28	0	Basque Fatherland and Freedom (ETA)
1990	1	6	160	1	2	21	5	3	0	New People's Army (NPA)
1987	9	16	183	1	3	23	5	2	0	African National Congress (South Africa)
2004	12	30	95	3	2	15	6	15	0	
2012	12	31	95	3	15	86	6	15	0	Al-Qa`ida in Iraq
2009	3	7	4	3	1	12	6	17	0	Taliban
1988	3	7	121	3	1	11	6	16	0	
1991	10	11	159	2	4	29	5	2	0	Shining Path (SL)

Figure 3. Screenshot of the Database Subset Used in the GTD Study.

Figure 4 shows the results that have arisen from the implementation of the cutoff function, an improvement of 8% relating to misclassification error and also a reduction in time from 72-100 hours to 1 hour. In its current implementation, the cutoff is set at 50 entries and we reduced the number of responses for gname from 2500+ to 165 (this suggests that many entries are one-shot terrorist attacks). Byproducts of developing the cutoff function are the arrays developed, which allows the user to see what variable names and values are cutoff (and there is also the possibility of creating a non-cutoff array, effectively splitting the data for analysis purposes).



Neural Network

Algorithms

Data Division:	Random	(dividerand)
Training:	Scaled Conjugate Gradient	(trainscg)
Performance:	Cross-Entropy	(crossentropy)
Calculations:	MEX	

Progress

Epoch:	0	304 iterations	1000
Time:		1:00:57	
Performance:	0.200	0.00818	0.00
Gradient:	0.750	0.00488	1.00e-06
Validation Checks:	0	6	6

Results

	Samples	CE	%E
Training:	33410	4.21881e-0	21.34390e-0
Validation:	7159	12.00268e-0	27.69939e-0
Testing:	7159	11.99623e-0	27.40606e-0

Optionally perform additonal tests

Inputs: NN4TestInput

Targets: NN4TestTarget

Samples are: Matrix columns [✓ Matrix rows]

Inputs 'NN4TestInput' is a 20454x367 matrix, representing static data: 20454 samples of 367 elements.

Targets 'NN4TestTarget' is a 20454x166 matrix, representing static data: 20454 samples of 166 elements.

CE	1.10743e-2
%E	28.82076e-0

Figure 4. Results for GTD Pattern Recognition Neural Network.

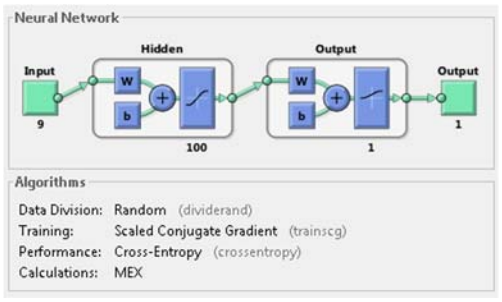
Profiles of Incidents Involving CBRN Use by Non-state Actors (POICN) Results

We utilized the Breiger dataset [2] as the subset to analyze for our pattern recognition studies.

- **Inputs:** RussiaNIS, MiddleEast, SouthAsia, is.bio, is.chem, perp.lone, perp.relex, perp.cult, perp.ethnonat.
- **Target:** eventtype

The inputs describe the location of the attack (Russia and Newly Independent States, Middle East or South Asia), the type of attack (biological or chemical) and the perpetrator (lone actor, cult, religious extremist or ethnonational). The target is whether an event is type A (Attempted Acquisition of CBRN weapon) or a type B event (Possession/Use of CBRN weapon).

Figure 5 shows the neural network training statistics and Figure 6 shows the confusion matrix for the results.



Neural Network

Algorithms

Data Division:	Random	(dividerand)
Training:	Scaled Conjugate Gradient	(trainscg)
Performance:	Cross-Entropy	(crossentropy)
Calculations:	MEX	

Figure 5. Pattern Recognition Neural Network setup for the POICN dataset.

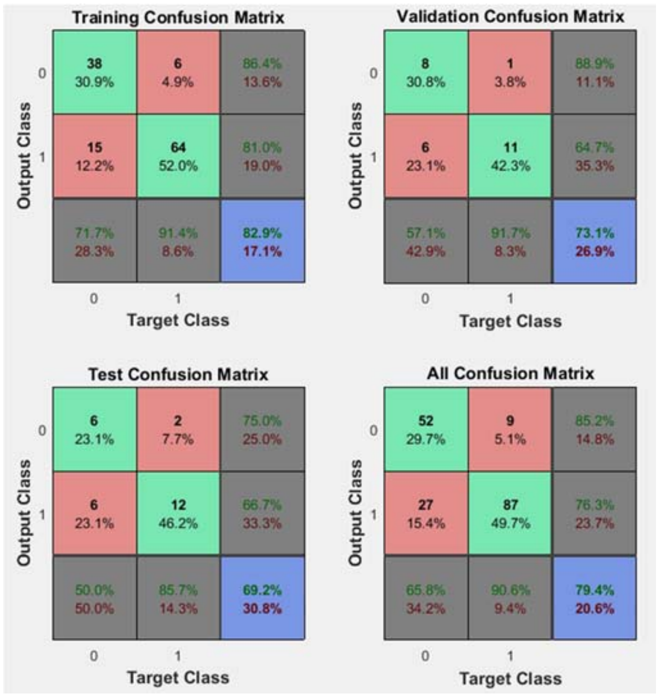


Figure 6. Confusion Matrix for Training/Test/Validation/All Dataset for POICN using Breiger's data.

Due to the small size of the POICN database, we were also able to analyze the data using different training methods, in addition to the default Scaled Congruent Gradient (SCG) method typically used for the pattern recognition neural network. We observed that Levenberg-Marquardt, Resilient Backpropagation and the other training methods all converged at 20.6% total confusion misclassification error after enough confusion loop iterations. Of all the training methods, Resilient Backpropagation converged with the least iterations.

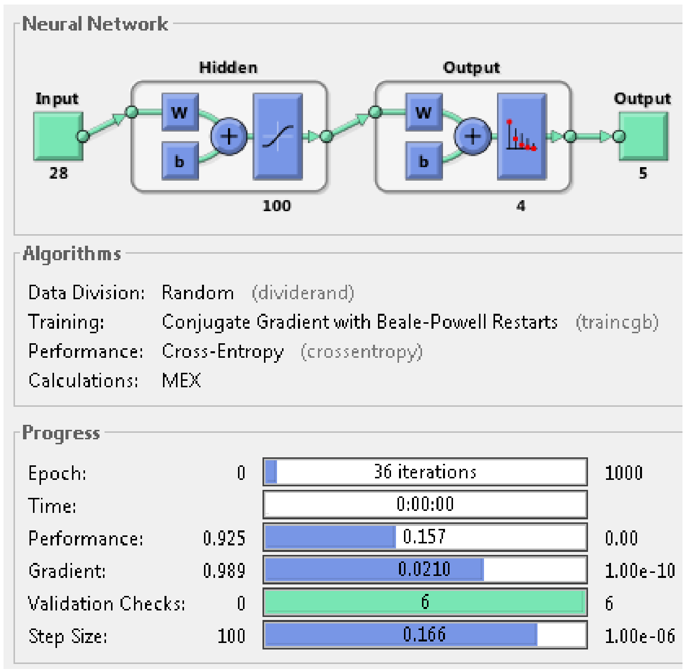
Profiles of Individual Radicalization in the United States (PIRUS) Results

We utilized the following input and target variables as a subset.

- **Inputs:** Radicalization_Islamist, Radicalization_Far_Right, Radicalization_Far_Left, Radicalization_Single_Issue, Ideological_Sub_Category1, Radical_Behaviors
- **Target:** Group_Membership

The input variables represent what type of radicalized ideal the person of interest in question has/had (islam, far right, far left, single issue ideal), his/her ideological (military, christian, animal rights, etc) and the radical behaviors they exhibit. The target variable we looked for is whether these input variables are related to whether or not an individual was part of a group related to his ideologies (i.e. is one ideology or radicalized ideal more prone to groups that share their cause rather than act alone).

Figures 7-8 show the neural network configuration and the confusion matrix for this subset. The misclassification error converged to ~33%.



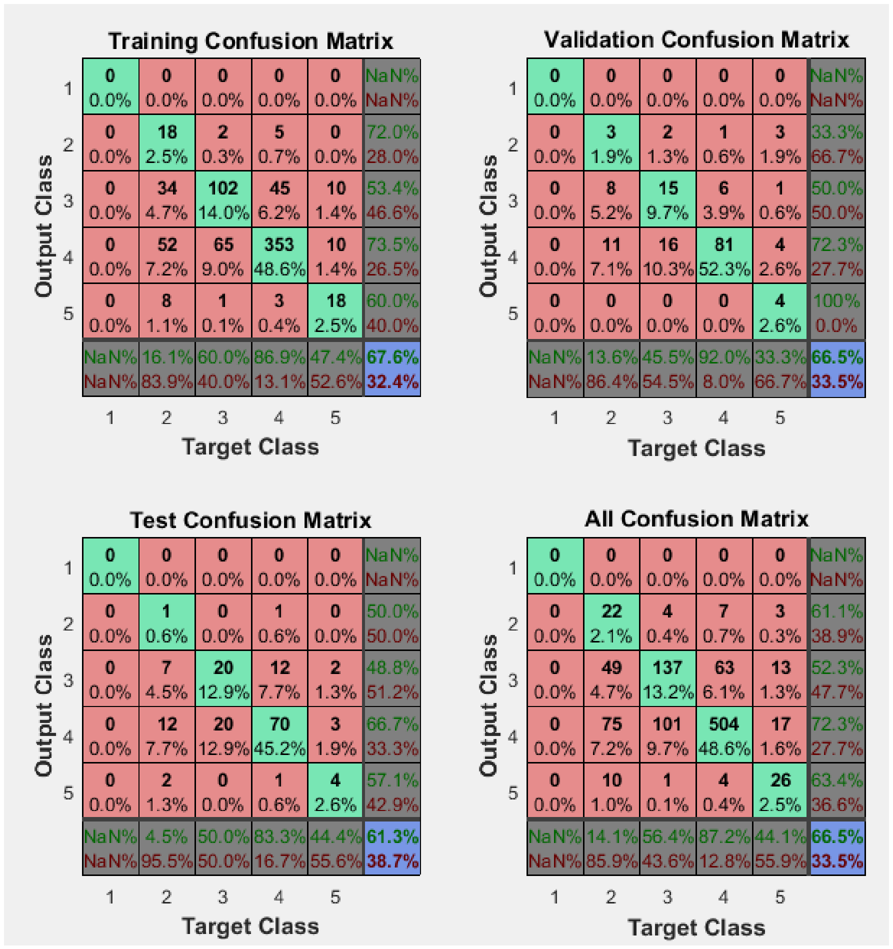
Algorithms

Data Division:	Random (dividerand)
Training:	Conjugate Gradient with Beale-Powell Restarts (traincgb)
Performance:	Cross-Entropy (crossentropy)
Calculations:	MEX

Progress

Epoch:	0	36 iterations	1000
Time:		0:00:00	
Performance:	0.925	0.157	0.00
Gradient:	0.989	0.0210	1.00e-10
Validation Checks:	0	6	6
Step Size:	100	0.166	1.00e-06

Figure 7. Pattern Recognition Neural Network Setup for PIRUS Dataset.



Training Confusion Matrix

Output Class \ Target Class	1	2	3	4	5	
1	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
2	0 0.0%	18 2.5%	2 0.3%	5 0.7%	0 0.0%	72.0% 28.0%
3	0 0.0%	34 4.7%	102 14.0%	45 6.2%	10 1.4%	53.4% 46.6%
4	0 0.0%	52 7.2%	65 9.0%	353 48.6%	10 1.4%	73.5% 26.5%
5	0 0.0%	8 1.1%	1 0.1%	3 0.4%	18 2.5%	60.0% 40.0%
	NaN% NaN%	16.1% 83.9%	60.0% 40.0%	86.9% 13.1%	47.4% 52.6%	67.6% 32.4%

Validation Confusion Matrix

Output Class \ Target Class	1	2	3	4	5	
1	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
2	0 0.0%	3 1.9%	2 1.3%	1 0.6%	3 1.9%	33.3% 66.7%
3	0 0.0%	8 5.2%	15 9.7%	6 3.9%	1 0.6%	50.0% 50.0%
4	0 0.0%	11 7.1%	16 10.3%	81 52.3%	4 2.6%	72.3% 27.7%
5	0 0.0%	0 0.0%	0 0.0%	0 0.0%	4 2.6%	100% 0.0%
	NaN% NaN%	13.6% 86.4%	45.5% 54.5%	92.0% 8.0%	33.3% 66.7%	66.5% 33.5%

Test Confusion Matrix

Output Class \ Target Class	1	2	3	4	5	
1	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
2	0 0.0%	1 0.6%	0 0.0%	1 0.6%	0 0.0%	50.0% 50.0%
3	0 0.0%	7 4.5%	20 12.9%	12 7.7%	2 1.3%	48.8% 51.2%
4	0 0.0%	12 7.7%	20 12.9%	70 45.2%	3 1.9%	66.7% 33.3%
5	0 0.0%	2 1.3%	0 0.0%	1 0.6%	4 2.6%	57.1% 42.9%
	NaN% NaN%	4.5% 95.5%	50.0% 50.0%	83.3% 16.7%	44.4% 55.6%	61.3% 38.7%

All Confusion Matrix

Output Class \ Target Class	1	2	3	4	5	
1	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	NaN% NaN%
2	0 0.0%	22 2.1%	4 0.4%	7 0.7%	3 0.3%	61.1% 38.9%
3	0 0.0%	49 4.7%	137 13.2%	63 6.1%	13 1.3%	52.3% 47.7%
4	0 0.0%	75 7.2%	101 9.7%	504 48.6%	17 1.6%	72.3% 27.7%
5	0 0.0%	10 1.0%	1 0.1%	4 0.4%	26 2.5%	63.4% 36.6%
	NaN% NaN%	14.1% 85.9%	56.4% 43.6%	87.2% 12.8%	44.1% 55.9%	66.5% 33.5%

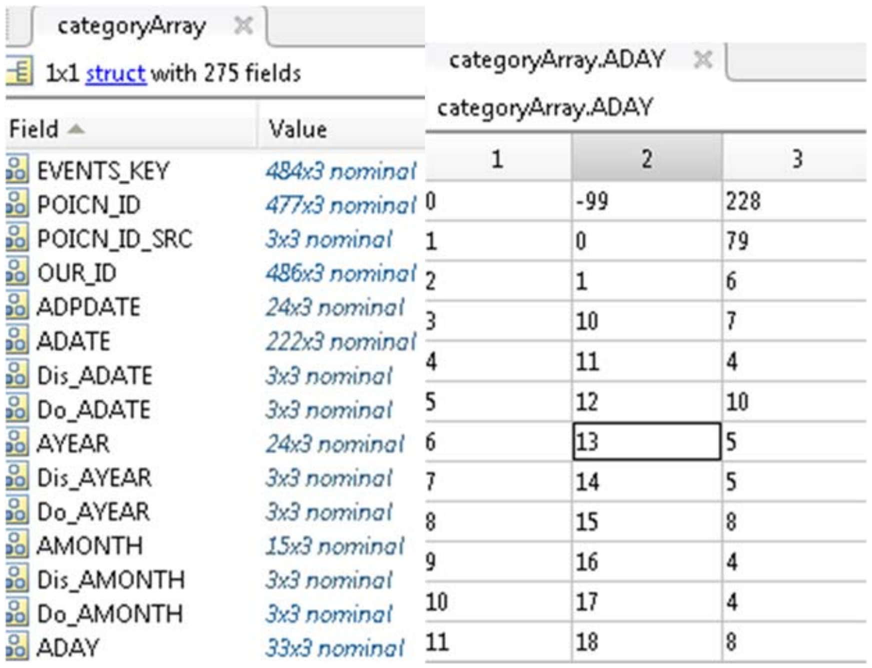
Figure 8. Confusion Matrix for Training/Test/Validation/All Dataset for PIRUS.

IMPROVING PATTERN RECOGNITION | MISSING DATA

Multiple additions have been done on the original script that was used for the purpose of generating and training the neural networks based on the START databases.

- We can now extract the amount of categorical options for each variable of the database, ensuring that we know the upper/lower limits of the possible values, as well as identify how many times each category occurs in the database.
    - This is done on the original categorical data, as well as the converted dummy/binary data, with the purpose of ensuring that the conversions encompass the entire scope of the original data.
- Development of clustering and regression functions utilizing the neural network toolbox to run, which is similar to the current process of pattern recognition. We have created an Amazon EC2 instance to allow for faster analysis of data, and used to test all of these processes for random sets of data.
- New "Cut-off" - Modified the previously developed confusion analysis scripts to cause them to loop to attempt to find a result below a certain cut-off. For some pairs of data sets, it has been possible to get percentages as low as 1.5%.
    - This approach, coupled with the correlator function and looping through all pairs of data was used as part of the missing data population function.

Figure 9 shows a screenshot of the categorical array (left) and a sample for attack date variable (right). The three columns shown for attack date, from left to right, represent the nominal code used by the conversion, the actual value in the original database and the number of occurrences that variable is seen in the database. Figure 10 shows the cut-off array (left) showing the variables and how many are getting cut (e.g. 4 values are getting cut-off from ADAY variable) and the values for ADAY that got cut-off due to low threshold.

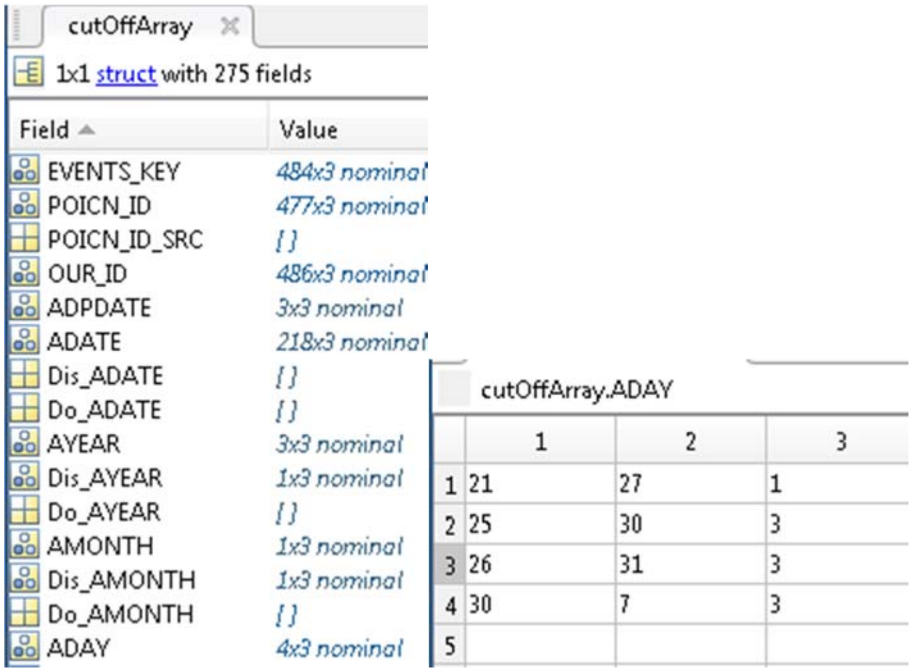

categoryArray
1x1 struct with 275 fields

Field	Value
EVENTS_KEY	484x3 nominal
POICN_ID	477x3 nominal
POICN_ID_SRC	3x3 nominal
OUR_ID	486x3 nominal
ADPDATE	24x3 nominal
ADATE	222x3 nominal
Dis_ADATE	3x3 nominal
Do_ADATE	3x3 nominal
AYEAR	24x3 nominal
Dis_AYEAR	3x3 nominal
Do_AYEAR	3x3 nominal
AMONTH	15x3 nominal
Dis_AMONTH	3x3 nominal
Do_AMONTH	3x3 nominal
ADAY	33x3 nominal

categoryArray.ADAY
categoryArray.ADAY

1	2	3
0	-99	228
1	0	79
2	1	6
3	10	7
4	11	4
5	12	10
6	13	5
7	14	5
8	15	8
9	16	4
10	17	4
11	18	8

Figure 9. Screenshot of the categorical array splitting each variable into the possible values the variable can take (left) and the values and occurences for the attack date variable (right) with its corresponding nominal code in column 1.


cutOffArray

1x1 struct with 275 fields

Field	Value
EVENTS_KEY	484x3 nominal
POICN_ID	477x3 nominal
POICN_ID_SRC	[ ]
OUR_ID	486x3 nominal
ADPDATE	3x3 nominal
ADATE	218x3 nominal
Dis_ADATE	[ ]
Do_ADATE	[ ]
AYEAR	3x3 nominal
Dis_AYEAR	1x3 nominal
Do_AYEAR	[ ]
AMONTH	1x3 nominal
Dis_AMONTH	1x3 nominal
Do_AMONTH	[ ]
ADAY	4x3 nominal

cutOffArray.ADAY

	1	2	3
1	21	27	1
2	25	30	3
3	26	31	3
4	30	7	3
5			

Figure 10. Screenshot of the cut-off array indicating the number of entries per variable that occurred less than the specified threshold (in this case 4) and the values for the variables that get cut-off and consolidated into "others" for attack day.

In addition, we are also able to study the R and P-values for the correlation between every pair of variables in the datasets used for the analysis.

RESULTS

- We pinpointed the issues that arise as a result of the binary implementation.
- The cutoff function improves overall percentages for all variables types when the dataset is large.
- For small data sets where the training time is low, dummy variables are good enough.

The implementation of the cut-off function to remove some irrelevant data (i.e. data that appears very few times and which entries might not be in one of the data sets as a result) has allowed us to improve the misclassification error by 8% and reduced the neural network training time to 1 hour. In our previous quarterly progress report we made mention of a wide variety of predictor/response combinations and the errors corresponding to each; however, the implementation of the cutoff has solved the problem. Using tools like Amazon EC2 Cloud Computing significantly reduces the neural network training time. Deepthought2 supercomputing would improve this even further.

In order to further improve on the misclassification percentages, we have worked on implementing other functions that look at relationships between variables. For example, the correlator function compares every pair combination of variables used for the data set; this allows us to see how variables are related to each other and to the predictor. In addition to correlation, we also looked into implementing multiple other neural network types (i.e. clustering, regression), aside from the current pattern recognition NN approach. For the current effort, as the code grows in size to accommodate multiple function, it becomes increasingly difficult to control all aspects of the coding process. In addition, due to the increasing data manipulation that takes place at the data import function, the script run time without running neural network training is continuously on the rise.

LOGICAL REGRESSION AND NEURAL NETWORKS COMPARISON STUDIES

The focus was mainly on exploring the statistical mechanical/thermodynamics aspects of neural networks, and presenting the initial efforts of this exploration. First, we present a brief discussion on one example of the statistical mechanical analog of neural networks, followed by work on this topic based specifically on data from the POICN database.

The neural network variant that will be used to explore this connection is referred to as a Hopfield Network, among many other names, such as an associative memory network, etc. The Hopfield Network developed by John Hopfield in 1982 [3] consists of a fully connected cyclic array of neurons, where the output of each neuron is fed into some other neurons [4], as depicted in Figure 11.

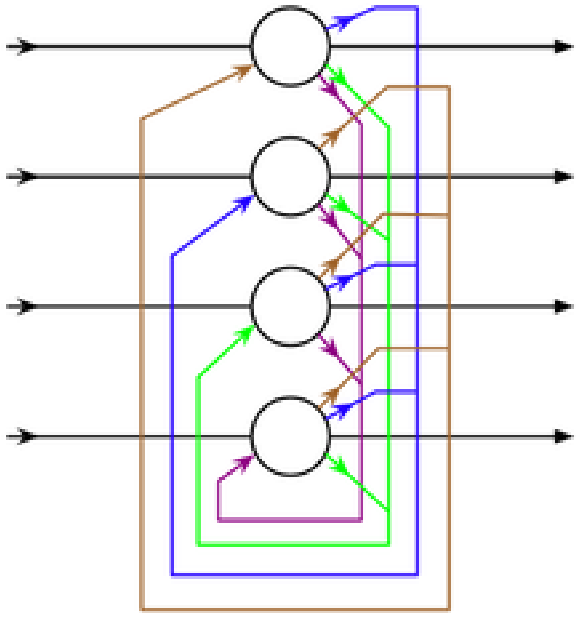


Figure 11. A Hopfield Network Consisting of 4 Nodes. (https://en.wikipedia.org/wiki/Hopfield_network)

Hence the neurons would continuously transmit signals back and forth to one another until a stable equilibrium is reached. The cyclic and bi-directional nature of the feedback loop of Hopfield nets is what differentiates Hopfield networks from the feed-forward networks that were discussed and created earlier. In the directed graph representation of feed-forward networks, signals are only allowed to propagate in one direction towards the output [4].

Each neuron of a Hopfield network, represented as a node can be continuously activated up to some threshold, at which point it is able to fire. Conversely, the network may also be made to operate in a discrete fashion by outputting the binary values of ±1 at the respective asymptotic values of ±∞ of the sigmoid activation function tanh (a_i*v/2) displayed in Figure 12.

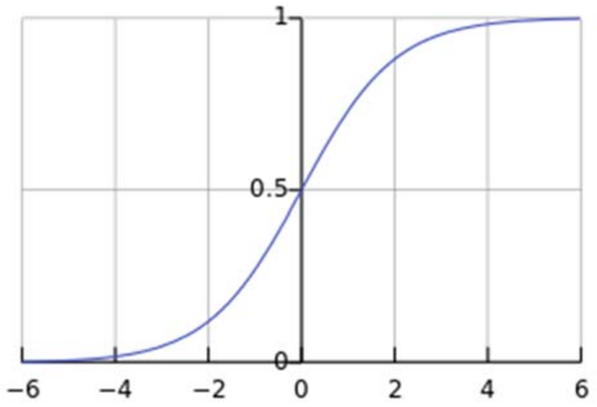


Figure 12. Sigmoid Activation/"logit" Function. (https://en.wikipedia.org/wiki/Sigmoid_function)

The correspondence between statistical mechanics and the Hopfield neural net is usually described in terms of the analogy between the "Energy" functional of this type of neural network shown in Equation 1, and the 2-dimensional Ising spin Hamiltonian as compared in Table 1.

$$H_N = E = \left(-\frac{1}{2}\right) \sum_{i \neq j} W_{ij} \sigma_i \sigma_j \qquad (1)$$

Table 1: Analogy between the Hopfield network "Energy" functional and the 2-D Ising spin Hamiltonian.

Hopfield Network	**2-D Ising Spin System**
E, "Energy Functional"	H_N, "Hamiltonian"
W_{ij}, Matrix of weights of neuron connections	W_{ij}, Nearest neighbor interaction energy of spins
σ_j, neuron in state j	σ_j, spin in state j

In short, the cooperative interaction of the neurons of the Hopfield network, where the neuron weights are adjusted during the minimization of the "Energy" functional corresponds to the induced ferromagnetic character and/or long-range order of the 2-D Ising spin system, where the flipping of a spin at one lattice site affects the behavior of spins at sites further away.

Below in Table 2 is an excerpt of the input data used in the creation of a Hopfield network. Although the nine variables are the same as those used by Breiger and Melamed [2], the actual data was taken from the more recent version of the POICN database we have been using throughout our research.

Table 2: Portion of the input data used in creating a Hopfield network.

RussiaNIS	**Middle East**	**South Asia**	**is.bio**	**is.chem**	**perp.lone**	**perp.relex**	**perp.cult**	**perp.ethnonat**
0	0	0	0	1	0	1	0	0
0	0	0	0	0	0	1	0	0
0	0	0	0	1	0	0	1	0
0	0	0	1	1	0	1	0	0
0	0	0	1	1	0	0	0	0
0	1	0	1	1	0	1	0	0
0	0	0	1	1	0	1	0	0
0	0	0	1	1	0	1	0	0
0	0	1	0	1	0	1	0	0
0	0	1	0	1	0	1	0	0
0	0	0	1	1	0	1	0	0
0	0	1	0	1	0	0	0	0

Figure 13 is a plot of an activation map of the input data used in the creation of the Hopfield net based on POICN data, displayed in the form of a heatmap, where each pairwise unit of the data is likely to be a type B event (Possession/Use of CBRN) weapon is shaded in red, and the type A events (Attempted Acquisition of CBRN weapon) is shaded white. The data used in the construction of this figure was for all 300 plus events in the POICN database.

A major result of this final report regarding the work associated with the POICN database, is the partial resolution of the previously mentioned conceptual difficulty of determining the equivalence of the roles played by the similar quantities of the logistic regression coefficients and neural network weights. However, we present a brief review of some other work related to another goal discussed in the third quarterly report, which was the study of the possible statistical thermodynamic analogies related to the neural network analysis applied to the classification of the two "Type A/ Type B" broad categories of CBRN events via the POICN data.

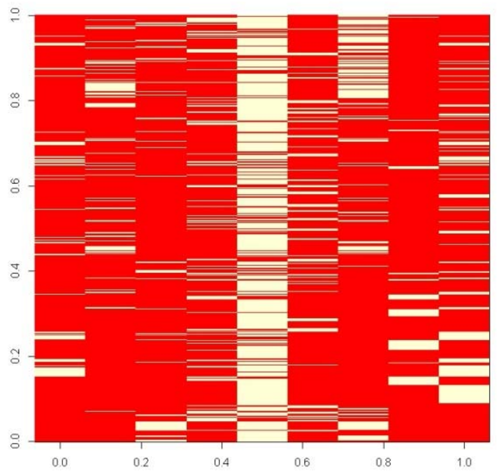


Figure 13. Heatmap that shows for each pairwise unit in POICN data the amount of cases that are type B Posession/CBRN events (red), and type A Acquisition events (white).

According to Dreisetl et al [5], in the case of logistic regression, the coefficients determined based on data from the input data matrix D, are usually obtained by using the method of maximum likelihood estimation, which falls under the category of a parametric method. For neural networks (e.g. feedforward neural networks), the methods involved in obtaining the network weights, such as that involving minimizing the cross-entropy are non-parametric methods. Therefore, according to [5], because of this difference, the parameters' role of providing the relative contribution of each variable to the outcome of a logistic regression, usually does not apply to the weights of neural networks. Hence, the non-equality between the regression coefficients, and feedforward neural network weights, which was an issue discussed in earlier reports appears to have been a plausible outcome.

GOALS

A. Pattern Recognition and Missing Data Methodology

- We have pinpointed the issues that arise as a result of the binary implementation.
- The cutoff function improves overall percentages for all variables types when the dataset is large.
- For small data sets where the training time is low, dummy variables are good enough.

(i) The implementation of the cut-off function to remove irrelevant data (i.e. data that appears very few times and which entries might not be in one of the data sets as a result) has allowed us to improve the misclassification error by 8% and reduced the neural network training time to 1 hour.

(ii) Earlier we have mentioned a wide variety of predictor/response combinations and the errors corresponding to each, but the implementation of the cutoff has eliminated the need to worry about that.

(iii) Using tools like Amazon EC2 Cloud Computing (and Deepthought2 supercomputing), the neural network training time has decreased significantly.

B. Logical Regression and Neural Networks Comparison Studies

- Adjustment of the R-Program script to obtain data from current and more recent POICN data (e.g. inclusion of missing R-functions, addition of necessary packages, debugging, etc.).
- Moved from the creation of feed-forward logistic perceptrons to the initial creation of Hopfield type neural networks.

BARRIERS AND STRATEGIES

- Amazon EC2 cloud computing resource worked well to speed up the neural network training process.
- We have made effort to compress the data files during the initial neural network import process in order to speed up the training phase of the machine learning algorithm. However, other mechanisms, such as choice of cut-off functions, high performance cluster computing through Amazon EC2 reduced the training times substantially, whereby data compression was not a significant requirement for the data fitting.

TECHNICAL ASSISTANCE NEEDS

- Continued IT support at the START Center, University of Maryland, for any possible collaborative research effort.

ADDITIONAL INFORMATION

- Build strong partnerships with research personnel at a DHS Center of Excellence.
- It is important to work closely with researchers at the partnering DHS Center of Excellence and seek their input and advice at each critical aspect of your project, so that the key outcomes and deliverables of your effort will be useful to DHS.

DELIVERABLES

(See Appendix A and Appendix B)

The deliverables include:

- Matlab scripts for utilizing the machine learning algorithms developed for the purpose of finding patterns, missing data regression and clustering in START databases.
  - The Matlab Coder/Compiler toolboxes can be used to convert the Matlab Scripts into a programming language (e.g. C/C#/C++, Visual Basic and Java) and then use the resulting code to develop a user-interface.
- Matlab implementation of Breiger's R scripts.

FUTURE WORK

The experience that the Howard University research team gained while working on this project, in collaboration with researchers at the START DHS Center of Excellence at the University of Maryland, has opened up several new and important research questions that we would like to pursue at Howard University, and these are summarized below.

- Implementation of our Matlab methodology and script submission to Howard University's Stars Cluster: http://helios.physics.howard.edu/
- Code Optimization to speed up execution phase.
- Study the concept of Vectorization using R and implementing similar routines using it.
- Study Neural Networks with Hashing as a way of data compression.
- Use Simulink to add a Model-based Design layer to our research implementations.
- Expand and study other events (such as weather, etc.) using this approach.
- Study logistic regression on other START databases, besides POICN.
- Use neural networks as a data compression tool for images and videos and subsequent pattern recognition.
- Implementation and fusion of social science techniques with physical and mathematical concepts to develop hybrid machine learning algorithms.
- Expansion of current START DHS efforts dealing with static information, such as databases to dynamic information, such as streaming data.
- Development of a global social events database query and pattern recognition tool that uses established data gathering social sciences techniques - in order to address missing data concerns and make predictions on future social events based on trends that have occurred in the past. For example, the economic condition decay over a period of years can be used for determining if similar conditions in other time periods can result in potential riots or coups, similar to what occurred during the Arab spring.
- Generalization and combination of social event data from multiple sources (IARPA, DHS, DARPA, DOD, etc.) into one global, encompassing system that catalogs and makes determinations based on user needs.

1. Research Presentations

1. Summer 2015 Progress Report presentation at START, May 15, 2015.
2. Fall 2015 Progress Report presentation at START, September 4, 2015.
3. Presentation to Paul Tandy of Defense Threat Reduction Agency (DTRA) at START, October 13, 2015.
4. Spring 2016 Progress Report presentation at START, February 5, 2016.
5. DHS LEAP Workshop invited oral and poster presentation, Washington, DC, March 30, 2016.

2. American Association of Physics Teachers (AAPT) Conference Presentations:

The following paper [DB05] "Application of Statistical Mechanics and Neural Networks for Large Databases," P. Misra, D. Casimir, R. Garcia-Sanchez, will be presented orally at the AAPT Summer 2016 Meeting in Sacramento, California, July 16-20, 2016.

The following poster [PST2D07] titled “Neural Networks and Matlab Algorithms for Pattern Recognition in Databases,” R. Garcia-Sanchez, D. Casimir, P. Misra, G. Ackerman, M. Binder, will also be presented at the AAPT Summer 2016 Meeting in Sacramento, California, July 16-20, 2016.

REFERENCES:

[1] Innovative Algorithm and Database Development Relevant to Counterterrorism and Homeland Security Efforts at START, Report to the Office of University Programs, Science and Technology Directorate, U.S. Department of Homeland Security, August 2014.

[2] Ronald L. Breiger and David Melamed, "The Duality of Organizations and Their Attributes: Turning Regression Modeling "Inside Out"", Contemporary Perspectives on Organizational Social Networks. Research in the Sociology of Organizations, vol. **40**, pp. 263-275, 2014, Emerald Group Publishing Limited.

[3] J.J. Hopfield, "Neural networks and physical systems with emergent collective computational abilities", Proceedings of the National Academy of Sciences of the USA, vol. **79** no. 8, pp. 2554-2558, April 1982.

[4] Neural Networks: A Tutorial, Michael Chester, Prentice Hall, 1993.

[5] Stephan Dreisetl, and Lucila Ohno-Machado, "Logistic regression and artificial neural network classification models: a methodology review", Journal of Biomedical Informatics, **35**, 2002, pp. 352-359.

APPENDIX A: MATLAB SCRIPTS

MAIN SCRIPT

```
%%Script to create START-based networks.

%% Import data from spreadsheet

% Set the flag value of the import function
% 1 - Look for files, if those files do not exist or do not have a STARTSet
% variable, then run the import function
% 2 - force an import. Use this if you have changed the includedVars,
% causing the files to be out of date.
% 3 - Check if STARTSet exists in the workspace, if it does, use that
% version. This is just for my own sake while I debug the program till
% completion.
importflag = 3;

% Flag for the database you want to use for import.
datasetflag = 'GTD';

switch importflag
  case 1
    if(exist('gtd.mat','file'))
      load('gtd.mat')
    elseif(exist('poicn.mat','file'))
      load('poicn.mat')
    elseif(exist('pirus.mat','file'))
      load('pirus.mat')
    %Keep adding elseifs for whatever other databases you want to use
    end
    if(~exist('STARTSet','var'))
      STARTSet = importSTARTDatabase(datasetflag);
    end
  case 2
    STARTSet = importSTARTDatabase(datasetflag);
  case 3
    if(~exist('STARTSet','var'))
      STARTSet = importSTARTDatabase(datasetflag);
    end
end

%% Sort by desired response
```

```
% Sort rows by Response variable (typically last variable in dataset)
headers = STARTSet.Properties.VarNames;
[allRows,allCols] = size(STARTSet);
STARTSet = sortrows(STARTSet,allCols,'ascend');
unknownCutOff = find(strcmp(STARTSet.(headers{end}),'-99'), 1, 'last' );

% Flags for nominal conversion
categoryT = true(1,allCols);
catPredT = categoryT(1:end-1);

% Convert all the categorical variables into nominal arrays
STARTNominalSet = convertToNominal(STARTSet);

%% Category Tracking
% Only used internally for my own knowledge. The actual use occurs within
% the cutOff function below.
[categoryArray,totalVarCat] = getCategoryArray(STARTNominalSet);

%% Bunch o' stuff to create thresholds for low occuring variable values
[cutOffArray,cutOffSize] = categoryCutOff(STARTNominalSet);

% Use the cutoff array to consolidate entries in the dataset
STARTConsolidatedSet = consolidateVars(STARTSet,cutOffArray,cutOffSize);

% Create a nominal dataset based on the consolidated
STARTNominalConsolidatedSet = convertToNominal(STARTConsolidatedSet);
[nominalCategoryArray,nominalTotalVarCat] = getCategoryArray(STARTNominalConsolidatedSet);

%% Convert to Binary
STARTNumSet = double(STARTNominalConsolidatedSet)-1;
[STARTBinSet,STARTBinMat,binVarSize]                                           =
convertToBinary(STARTNominalConsolidatedSet,STARTNumSet);

%% Splitting the unified category dataset
STARTPredictionSet = STARTNominalConsolidatedSet(1:unknownCutOff,1:end-1);
STARTTrainingTestSet = STARTNominalConsolidatedSet(unknownCutOff+1:end,1:end);

% Determine set sizes.
[nrowsT, ncolsT] = size(STARTTrainingTestSet);
[nrowsP, ncolsP] = size(STARTPredictionSet);
```

```
%% Dividing Inputs/Targets from Datasets
% Training set

% Binary Inputs
%inputs = STARTBinMat(unknownCutOff+71:end,1:sum(binVarSize(1:end-1)));
%targets = STARTBinMat(unknownCutOff+1:end,sum(binVarSize(1:end-1))+1:sum(binVarSize(1:end)));

% Dummy Indicator Inputs
inputs = STARTNumSet(unknownCutOff+1:end,1:end-1);
targets = STARTNumSet(unknownCutOff+1:end,end);

% Prediction set - no targets since they are blank.

%PredictionInputs = STARTBinMat(1:56905,1:sum(binVarSize(1:end-1)));
PredictionInputs = STARTNumSet(1:unknownCutOff,1:end-1);

%% Partition dataset into training and test sets
cv = cvpartition(length(STARTTrainingTestSet),'holdout',0.30);

% Train set crossvalidation
TrainInputs = inputs(training(cv),:);
TrainTargets = targets(training(cv),:);
% Test set crossvalidation
TestInputs = inputs(test(cv),:);
TestTargets = targets(test(cv),:);

%% Replace if(false) with if(~false) to activate the desired section.

%% NN3 uses dummy indicator variables for predictors but not responses. Faster, but R is not as high as
NN2.
%Convert Categorical predictors to Dummy Indicator Variables
if(~false)
  [NN3TrainInputs, NN3TrainTargets, NN3TestInputs, NN3TestTargets] =
preparedataNN(STARTTrainingTestSet, catPredT, cv);
end

%% NN4 uses dummy indicator variables for predictors and responses by using
% category instead of catPred, then prepareNN converts all of the dataset's categorical variables into
dummy indicator variables.
if(~false)
```

```
    %Convert categorical predictor and response columns to dummy indicator variables
    [NN4TrainingSet, ~, NN4TestSet, ~] = preparedataNN(STARTTrainingTestSet, categoryT, cv);
    %Split the Train/Test Predictor NN outputted by the function into Input and Target (since it includes
both
    %predictors and responses as a result of the above function)
    NN4TrainInput = NN4TrainingSet(:,1:size(NN3TrainInputs,2));
    NN4TrainTarget = NN4TrainingSet(:,size(NN3TrainInputs,2)+1:end);
    NN4TestInput = NN4TestSet(:,1:size(NN3TrainInputs,2));
    NN4TestTarget = NN4TestSet(:,size(NN3TrainInputs,2)+1:end);

    % Clear temporary variables
    %clearvars NN4TrainingSet NN4TestSet;

end

%%
%poicn_NN1 = patternet(1024);
%poicn_NN1 = configure(poicn_NN1, NN4TrainInput',NN4TrainTarget');
%[poicn_NN2,stats] = train(poicn_NN1, NN4TrainInput',NN4TrainTarget','useParallel','yes');
```

IMPORT DATABASE SCRIPT

```
function importedDatabase = importSTARTDatabase(workbookString,sheetName)

%% Import data from spreadsheet
% Auto-generated by MATLAB on 2015/04/28 13:54:35

%Flags that might pop up in the dataset and will be all converted to -99.
unknownFlags = {'Unknown' '-66' '-9' ''};

%% Input handling
% Select the directory location of the workbook file chosen.
switch upper(workbookString)
  case 'GTD'
    workbookFile = 'U:\START Matlab\Databases\GTD\globalterrorismdb_0814dist.xlsx';
  case 'POICN'
    workbookFile = 'U:\START Matlab\Databases\POICN\POICN 2.35 2015 (Weapons+Orgs 2011 w
TORG).xlsx';
  case 'PIRUS'
    workbookFile = 'U:\START Matlab\Databases\PIRUS\PIRUS v1.5 deidentified version.xlsx';
  otherwise
    disp('Incorrect database string')
end

% If no sheet is specified, read first sheet
if nargin == 1 || isempty(sheetName)
  sheetName = 1;
end

%% Import the data
[~, ~, raw] = xlsread(workbookFile, sheetName);
%raw(cellfun(@(x) ~isempty(x) && isnumeric(x) && isnan(x),raw)) = {'-99'};

%% Extract headers from the imported data
% This way, you can see what index correspondsto what header for the purpose
% of selecting variables with includeVars later in the code.
importHeaders = raw(1,:);
cols = size(raw,2);

%Cut-off the weird NaN rows that pop up from Excel's poor formatting.
for i = 1:cols
  if(isnan(importHeaders{i}))
```

```
        importHeaders = importHeaders(1:i-1);
        raw = raw(:,1:i-1);
        break;
    end
end

% Remove the first row, which corresponds to the var names from the dataset
raw(1,:) = [];
cols = size(raw,2);

%% Determine what variable corresponds to each column
varCatalog = [num2cell(1:cols)',importHeaders']; %#ok<*NASGU>

%% Select Variables to Use based on the indexes in importHeaders
% switch upper(workbookString)
%     case 'GTD'
%         includedVars = [2,3,4,8,29,35,37,81,83,109,59];
%         cellVectors = raw(:,includedVars);
%     case 'POICN'
%         includedVars = [5,12,15,51,112,115,118,121,127,204,208,212,213,217,221,222,124];
%         cellVectors = raw(:,includedVars);
%     case 'PIRUS'
%         includedVars = [];
%         cellVectors = raw(:,includedVars);
% end

%Used internally for whenever I want to grab the entire database variables
includedVars = 1:size(raw,2);
cellVectors = raw;

[includedRows,includedCols] = size(cellVectors);
includedHeaders = cell(1,includedCols);

for i = 1:includedCols
    includedHeaders{i} = importHeaders{includedVars(i)};
    includedHeaders(i) = regexprep(includedHeaders(i),'/','');
    includedHeaders(i) = regexprep(includedHeaders(i),' ','');
    includedHeaders(i) = regexprep(includedHeaders(i),'-','');
end

%% Convert numeric entries into non-numeric
```

```
for j = 1:includedCols
   for i = 1:includedRows
      if(isnumeric(cellVectors{i,j}) && ~isnan(cellVectors{i,j}))
         cellVectors{i,j} = num2str(cellVectors{i,j});
         if(sum(strcmp(cellVectors{i,j},unknownFlags)))
            cellVectors{i,j} = '-99';
         end
      elseif(sum(strcmp(cellVectors{i,j},unknownFlags)) || sum(isnan(cellVectors{i,j})))
         cellVectors{i,j} = '-99';
      end
   end
end

%% Create table
importedDatabase = dataset;

%% Allocate imported array to column variable names
for i = 1:includedCols
   importedDatabase.(includedHeaders{i}) = cellVectors(:,i);
end
```

NOMINAL CONVERSION SCRIPT

```
%% Convert all the categorical variables into nominal arrays
function NominalSet = convertToNominal(STARTSet)

headers = STARTSet.Properties.VarNames;
allCols = size(STARTSet,2);
category = false(1,allCols);
NominalSet = dataset;

for i = 1:allCols
  if isa(STARTSet.(headers{i}),'cell') || isa(STARTSet.(headers{i}),'nominal')
    category(i) = true;
    try
      NominalSet.(headers{i}) = nominal(STARTSet.(headers{i}));
      i
    catch ME
      switch ME.identifier
      case 'MATLAB:categorical:MaxNumCategoriesExceeded'
      NominalSet.(headers{i}) = [];
      end
    end
  end
end
```

CATEGORY SORTING SCRIPT

```
%% Create an array that contains each variables occurences.
function [getCatArray,totalVarCat] = getCategoryArray(STARTSet)

headers = STARTSet.Properties.VarNames;
allCols = size(STARTSet,2);

% Determine number of existing categories for each header
for i = 1:allCols
   totalVarCat.(headers{i}) = max(double(STARTSet.(headers{i})));
   getCatArray.(headers{i}) = [];
end

% Determine said categories for each header (double | nominal) Shift by -1
% to match binary start at 0 (reduces "binary" values from 2 bits to 1 bit)
categoryValues = nominal(double(STARTSet)-1);

for i = 1:allCols
    getCatArray.(headers{i})                              =                              [unique([categoryValues(:,i),
STARTSet.(headers{i})],'rows'),nominal(histc(double(STARTSet.(headers{i})),1:totalVarCat.(headers{i}))
)];
end
```

REPETITION CUT-OFF SCRIPT

```
%Cut-off Function
function [cutOffArray,cutOffSize,nonCutOffArray] = categoryCutOff(STARTSet)

headers = STARTSet.Properties.VarNames;
allCols = size(STARTSet,2);
[categoryArray,totalVarCat] = getCategoryArray(STARTSet);

% Initialize the cut-off structure
for i = 1:allCols
   cutOffArray.(headers{i}) = [];
end

% Duplicate cutoff array for noncutoff indeces.
nonCutOffArray = cutOffArray;

% Separate the category struct into noncutoff and cutoff arrays.
cutOff = 10;
for i = 1:allCols
   k = 0;
   for j = 1:totalVarCat.(headers{i})
      if(str2num(char(categoryArray.(headers{i})(j,3))) < cutOff) %#ok<ST2NM>
         cutOffArray.(headers{i}) = [cutOffArray.(headers{i});categoryArray.(headers{i})(j,:)];
         k = k+1;
% If you want to know which values did not get cutoff by the threshold
% value.
%       else
%          nonCutOffArray.(headers{i}) = [nonCutOffArray.(headers{i});categoryArray.(headers{i})(j,:)];
      end
   end
   %Determine the size of each cutoff structure property and create a
   %struct of said values.
   cutOffSize.(headers{i}) = k;
end
```

VARIABLE CONSOLIDATION SCRIPT

```
%Cut-off Function
function [cutOffArray,cutOffSize,nonCutOffArray] = categoryCutOff(STARTSet)

headers = STARTSet.Properties.VarNames;
allCols = size(STARTSet,2);
[categoryArray,totalVarCat] = getCategoryArray(STARTSet);

% Initialize the cut-off structure
for i = 1:allCols
   cutOffArray.(headers{i}) = [];
end

% Duplicate cutoff array for noncutoff indeces.
nonCutOffArray = cutOffArray;

% Separate the category struct into noncutoff and cutoff arrays.
cutOff = 10;
for i = 1:allCols
   k = 0;
   for j = 1:totalVarCat.(headers{i})
      if(str2num(char(categoryArray.(headers{i})(j,3))) < cutOff) %#ok<ST2NM>
         cutOffArray.(headers{i}) = [cutOffArray.(headers{i});categoryArray.(headers{i})(j,:)];
         k = k+1;
% If you want to know which values did not get cutoff by the threshold
% value.
%        else
%           nonCutOffArray.(headers{i}) = [nonCutOffArray.(headers{i});categoryArray.(headers{i})(j,:)];
      end
   end
   %Determine the size of each cutoff structure property and create a
   %struct of said values.
   cutOffSize.(headers{i}) = k;
end
```

BINARY CONVERSION SCRIPT

```
%% Convert to Binary
function [STARTBinSet,STARTBinMatrix,binVarSize] = convertToBinary(STARTSet,STARTNumSet)

headers = STARTSet.Properties.VarNames;
[allRows,allCols] = size(STARTSet);
totalVarCat = max(double(STARTSet(:,1:end)));

%Determine Binary Variable Sizes
binVarSize = zeros(1,allCols);

for i = 1:allCols
  binVarSize(i) = size(de2bi(totalVarCat(i)-1),2);
end
% For clarification, this dataset contains the name of the variables and
% the size of the binary string associated to it to make easier to read.
% I.e. iyear = 6 means that the first six binary values in the binary
% matrix correspond to iyear and so subsequently.
binVarSizeSet = mat2dataset(binVarSize);
binVarSizeSet.Properties.VarNames = headers;

%Create a numeric dataset and allocate space for the binary dataset
STARTBinSet = struct;

%Create a dataset with Binary values
for i = 1:allCols
   STARTBinSet.(headers{i}) = de2bi(STARTNumSet(:,i));
end
% Create a matrix with binary values by appending each variable's binary
% strings.
% Note: Can be changed later after developing a means to determine
% the full size of the final binary matrix.
STARTBinMatrix = zeros(allRows,sum(binVarSize));
offset = 1;

for i = 1:allCols
   STARTBinMatrix(:,offset:offset+binVarSize(i)-1) = STARTBinSet.(headers{i});
   offset = offset + binVarSize(i);
end

clearvars i;
```

NEURAL NETWORK FORMAT DATA SCRIPT

```
function [ XtrainNN, YtrainNN, XtestNN, YtestNN] = preparedataNN(GTDTrainingTestSet, catPred, cv )
%preparedataNN Prepare predictors/response for Neural Networks
%   When using neural networks the appropriate way to include categorical
%   predictors is as dummy indicator variables. An indicator variable has
%   values 0 and 1.

% Copyright 2013 The MathWorks, Inc.

% |dummyvar| accepts a numeric or categorical column vector (predictor
% variable), and returns a matrix of indicator variables. The dummy
% variable design matrix has a column for every group, and a row for every
% observation.

% Continuous predictors
X1 = double(GTDTrainingTestSet(:,~catPred));

% Each categorical predictor converted into dummy indicator variables
X2 = [];
categoricalPred = find(catPred);
for i = 1:length(categoricalPred)
   cP = double(GTDTrainingTestSet(:,categoricalPred(i)));
   X2 = [X2 dummyvar(cP)];  %#ok<AGROW>
end
% Predictor matrix
X = [X1 X2];

% Response
Y = double(GTDTrainingTestSet(:,end)) - 1;

% Use the same partition for cross validation
% Training set
XtrainNN = X(training(cv),:);
YtrainNN = Y(training(cv),:);
% Test set
XtestNN = X(test(cv),:);
YtestNN = Y(test(cv),:);

end
```

CONFUSION LOOP SCRIPT

```
%% Import the Data
if ~exist('poicn_NN1') %#ok<EXIST>
  load('I:\START_Data_Summer2014\POICN\breigerdata.mat', 'RaulBreigerDatasetX')

  %% Convert the Data to Neural Network Format
  RaulBreigerMatrixX = cell2mat(table2cell(dataset2table(RaulBreigerDatasetX)));
  mat_pred = RaulBreigerMatrixX(:,1:9);
  mat_targ = RaulBreigerMatrixX(:,10);

  %% Initialize Variables
  min_training_confusion = 100;
  min_test_confusion = 100;
  min_validation_confusion = 100;
  min_total_confusion = 100;

  poicn_NN1 = patternnet(100);
end

%% Configure and train the neural network
poicn_NN1 = patternnet(175);
poicn_NN1.trainFcn = 'trainbr';
poicn_NN1.performFcn = 'mse';
poicn_NN1.trainParam.epochs = 3000;
poicn_NN1 = configure(poicn_NN1, mat_pred',mat_targ');

for i = 1:10000
  [poicn_NN2,stats] = train(poicn_NN1,mat_pred',mat_targ');
  mat_out = poicn_NN2(mat_pred');

  %% Calculate confusion and performance
  % Training
  train_mask = mat_targ' .* stats.trainMask{1};
  train_performance = perform(poicn_NN2, train_mask, mat_out);
  train_confusion = confusion(train_mask, mat_out);

  % Test
  test_mask = mat_targ' .* stats.testMask{1};
  test_performance = perform(poicn_NN2, test_mask, mat_out);
```

```
    test_confusion = confusion(test_mask, mat_out);

    % Validation
    val_mask = mat_targ' .* stats.valMask{1};
    val_performance = perform(poicn_NN2, val_mask, mat_out);
    val_confusion = confusion(val_mask, mat_out);

    % Total
    total_confusion                                                                    =
confusion([train_mask(:,stats.trainInd),val_mask(:,stats.valInd),test_mask(:,stats.testInd)],[mat_out(:,stat
s.trainInd), mat_out(:,stats.valInd), mat_out(:,stats.testInd)]);

    if(train_confusion < min_training_confusion)
       min_train_confusion = train_confusion;
    end

    if(test_confusion < min_test_confusion)
       min_test_confusion = test_confusion;
    end

    if(val_confusion < min_validation_confusion)
      min_validation_confusion = val_confusion;
    end

    if(total_confusion < min_total_confusion)
      min_total_confusion = total_confusion;
    end

    i
    train_confusion
    test_confusion
    val_confusion
    total_confusion

    if(total_confusion < 0.20)
       break
    end
end
%% Plot Confusion
plotconfusion(train_mask(:,stats.trainInd),mat_out(:,stats.trainInd),'Training',
val_mask(:,stats.valInd),mat_out(:,stats.valInd),'Validation',
test_mask(:,stats.testInd),mat_out(:,stats.testInd),'Test',
[train_mask(:,stats.trainInd),val_mask(:,stats.valInd),test_mask(:,stats.testInd)],[mat_out(:,stats.trainInd),
mat_out(:,stats.valInd), mat_out(:,stats.testInd)],'All')
```

APPRENDIX B: LOGISTIC REGRESSION R-SCRIPTS

MAIN SCRIPT

```
# D. Casimir, Thurs 4 Jun 2015. Added the three function obtained from Breiger,
#(find.number.of.points, linesRB, & plotRB) attached in separate File.
# DONE MAKING RECODE PROGRAM
#Also data frames created from the more recent POICN excel data are prefixed with Casimir_

RN <- Casimir_Events_RN_EVENTS
TARG <- Casimir_Events_TARG
namesRN <-names(RN)
namesTARG <- names(Casimir_Events_TARG)
AYEAR <- RN$AYEAR # class integer. 159 cases 1998-2009 (all) + 80 caes 1990-97 (Islamist) + 219 NA
#           Lauren slide for BTR July 2012
AYEAR9811 <- which(AYEAR >= 1998)
AYEAR9097 <- which(AYEAR <= 1997)

IAPDATE <- Casimir_Events_RN_EVENTS$IAPDATE
IAPDATE9811 <- which(IAPDATE >= 1998)
IAPDATE9097 <- which(IAPDATE <= 1997)


ACOUNTRY <-Casimir_Events_TARG$ACOUNTRY

Event_type <- Casimir_Events_RN_EVENTS$EVENT_TYPE
Protoplot <- 0 + (Event_type == 0)
Plot <- 0 + (Event_type == 1)
Attempt_acquisition <- 0 + (Event_type == 2)
Possession_nonweaponized <- 0 + (Event_type == 3)
Possession_weapon <- 0 + (Event_type == 4)
Threat_with_possession <- 0 + (Event_type == 5)
Attempted_Use <- 0 + (Event_type == 6)
Use <- 0 + (Event_type == 7)


agent_type <- rep(0,471)
is.chem <- (RN$CHEM==1)
is.bio <- (RN$BIO==1)
is.rad <- (RN$RAD==1)
is.nuc <- (RN$NUC==1)
s <- is.chem + is.bio + is.rad + is.nuc
# Note: there are substantial overlaps, e.g., 31 are chem AND bio.
#                              7 are chem AND rad.
agentBYtype <- matrix(0,4,8)
agentBYtype[1,] <- (table(is.chem, RN$EVENT_TYPE))[2,]
agentBYtype[2,] <- (table(is.bio,  RN$EVENT_TYPE))[2,]
agentBYtype[3,] <- (table(is.rad,  RN$EVENT_TYPE))[2,]
agentBYtype[4,] <- (table(is.nuc,  RN$EVENT_TYPE))[2,]
rownames(agentBYtype) <- c("chem","bio","rad","nuc")
colnames(agentBYtype)<-0:7
# BEWARE OF DOUBLE-COUNTING IN THE ABOVE TABLE!!

HighSophist <- Casimir_Events_RN_EVENTS$ATTACK_SOPHISTICATION
HighSophist <- recode(HighSophist, old=1, new=0)
HighSophist <- recode(HighSophist, old=2, new=0)
HighSophist <- recode(HighSophist, old=3, new=1)

HiInterest <- Casimir_Events_RN_EVENTS$HEIGHT_INTEREST
HiInterest_Dis <- Casimir_Events_RN_EVENTS$Dis_HEIGHT
HiInterest_Doubt <- Casimir_Events_RN_EVENTS$Do_HEIGHT

tempid.loc.in.sheet <- function(id, sheet=Casimir_Events_RN_EVENTS) {
 # Finds the row-number(s) (that is, "loc") in "sheet" of a given
 #   numerical TEMP_ID
 found <- sheet$TEMP_ID == id
 found <- which(found)
 return(found)
}

RNid <- Casimir_Events_RN_EVENTS$TEMP_ID
my.PERPTYPE <- rep(0, 471)
PERPTYPE <- Casimir_Events_PERP$PERP_TYPE
for (k in 1:length(RNid)) {
 loc.k <- tempid.loc.in.sheet(RNid[k], sheet=Casimir_Events_PERP)
 # if(length(loc.k) != 1) {print(c(k,PERPTYPE[loc.k]))}
 my.PERPTYPE[k] <- PERPTYPE[loc.k[1]] # Take FIRST type for PERPTYPE
 # Note: only 10 cases are multiples, and ONLY 4 cases of multiple TYPES
 # CAN USE "my.PERPTYPE" with any variable in Events_RN (it's same order)
 }


# Discrepancies: First number is k, next are tempid.loc in Events_PERP
# [1]  5  5  6  7  8  9 10
# [1] 22 27 28 29
# [1] 23 30 31
# [1] 47 55 56 57
# [1] 51 61 62 63
# [1]  54  66 475
# [1] 57 69 70
# [1] 59 72 73
# [1] 67 81 82
# [1] 148 163 164

doQ <- function(T) {
 a <- T[1,1]; b <- T[1,2]; c <- T[2,1]; d <- T[2,2]
 q <- (a*d) - (b*c)
 q <- q / ((a*d) + (b*c))
 return(q)
}

casualties <- pmax((Casimir_Events_RN_EVENTS$TKILL > 5), (Casimir_Events_RN_EVENTS$TWOUND >
20))
perp.lone <- 0 + (my.PERPTYPE == 1)
perp.1issue <- 0 + (my.PERPTYPE == 3)
perp.cult <- 0 + (my.PERPTYPE == 4)
perp.relex <- 0 + (my.PERPTYPE == 5)
perp.rightwing <- 0 + (my.PERPTYPE == 6)
perp.leftwing <- 0 + (my.PERPTYPE == 7)
perp.ethnonat <- 0 + (my.PERPTYPE == 8)


# MY EFFORT TO FIND "REGION" USING "do.region.r" PROGRAM
#  NOTE: IF MULTIPLE RECORDS FOR SAME TEMP_ID, ONLY THE FIRST IS CODED
#    FOR REGION.
# Prouced:
#   multiple.TEMP_ID is 0 if only one country given, and N if N countries are listed.
#      Index numbers in the above refer to ROW-NUMBER of Events_RN_Events
#   myRegion is a 471-long set of regions corresponding to rows of RN
lookup.list <- (do.region(Casimir_Events_TARG$ACOUNTRY))$lookup.list
multiple.TEMP_ID <- rep(0,471)
myRegion <- rep(0,471)
for (k in 1:471) {
 thisID <- Casimir_Events_RN_EVENTS$TEMP_ID[k]
 w <- which(Casimir_Events_TARG$TEMP_ID == thisID)
```

```
  if (length(w) > 1) {multiple.TEMP_ID[k] <- length(w)}
  thiscountry <- Casimir_Events_TARG$ACOUNTRY[w[1]]
  ww <- which(lookup.list[,1] == thiscountry)
  if(identical(ww, integer(0))) {myRegion[k] <- -99}
  else {myRegion[k] <- lookup.list[ww,2]}
}
# Done producing 'myRegion'
AfricaMiddleEast <- (myRegion == 10) | (myRegion == 11)
SubSahAfrica <- myRegion == 11
MiddleEast <- myRegion == 10
CentSouthAmer <- (myRegion == 2) | (myRegion == 3)
RussiaNIS <- myRegion == 12
WestEur <- myRegion == 8
NorthAmer <- myRegion == 1
SouthAsia <- myRegion == 6
EastAsia <- myRegion == 4

credibility <- Casimir_Events_RN_EVENTS$CREDIBILITY

objectivity <- Casimir_Events_SOURCES$Objectivity
competence <- Casimir_Events_SOURCES$Competence

compound.perp <- perp.rightwing | perp.leftwing
compound.perp <- compound.perp + 0
# use.cases <- which((IAPDATE >= 1998) & (credibility == 3) & (AYEAR >= 1998))
  use.cases <- which((IAPDATE >= 1998) & (credibility == 3))
# use.cases <- which(AYEAR >= 1998) & (credibility == 3)
X <- cbind(RussiaNIS, MiddleEast, SouthAsia, is.bio, is.chem, perp.lone, perp.relex, perp.cult,
perp.ethnonat)
# X <- cbind(RussiaNIS, MiddleEast, SouthAsia, is.bio, is.chem, compound.perp)
UseAUse <- Event_type >= 6
PlotAcq <- (Event_type == 1) | (Event_type == 2)
# m1 <- lm(Event_type[use.cases] ~ X[use.cases,])
# m1 <- glm(UseAUse[use.cases] ~ X[use.cases,], family=binomial())

doubt.yes <- which(Casimir_Events_RN_EVENTS$DOUBT_TERR[use.cases] == 1)
doubt.no  <- which(Casimir_Events_RN_EVENTS$DOUBT_TERR[use.cases] != 1)

interest.hi <- which(HiInterest[use.cases] == 1)
interest.low <- which(HiInterest[use.cases] != 1)
```

```
# commented out 28 May 2015 per Breiger's response. (Daniel Casimir) bs.c <- A %*%
diag(as.vector(logit.phat))
# D. Casimir (28 May 2015) interest1 <- rowSums(bs.c[,interest.low])
# D. Casimir (28 May 2015) interest2 <- rowSums(bs.c[,interest.hi])
# D. Casimir (28 May 2015) interest <- cbind(interest1, interest2)
# D. Casimir (28 May 2015) colnames(interest) <- c("InterestLOW", "InterestHIGH")

m1 <- glm((Event_type >= 4)[use.cases] ~ X[use.cases,], family=binomial())

# Scale X:
Z <-scale(X)
m2 <- glm((Event_type >=4)[use.cases] ~ Z[use.cases,], family=binomial())
# "m2" has same df, deviance, z-values (EXCEPT for the intercept) as "m1"
# ===> m2 IS TABLE 3 (STANDARDIZED) IN BREIGER-MURRAY-PINSON ISA 2013 PAPER.

# ===> BECAUSE OF A STUPIDITY, Z[use.cases,] does not have column means of 0, etc. Would prefer to run
"m2a" (just below), which has identical deviances, df, but different coefficients:

Zprime <- X[use.cases,]
Zprime <- scale(Zprime)
m2a <- glm((Event_type >= 4)[use.cases] ~ Zprime, family=binomial())

# Let J refer to the **3** variables: is.chem, perp.relex, perp.ethnonat
Jprime <- Zprime[,c(5,7,9)] # is.chem, perp.relex, perp.ethnonat
m2j <- glm((Event_type >= 4)[use.cases] ~ Jprime, family=binomial())

# =====> SHOW THAT BETAhat = V (inv(S)) t(U) Y* 1
u187 <- matrix(1, 187, 1)
Zstar<-cbind(u187, Zprime)
svd.Z <- svd(Zstar)
Ustar <- svd.Z$u
Vstar <- svd.Z$v
Sstar <- diag(svd.Z$d)
names <- c("u", "RussiaNIS", "MidEast","SouthAsia","is.bio","is.chem","perp.lone","perp.relex", "perp.cult",
"perp.ethnonat")
rownames(Vstar)<-names
phat2 <- m2$fitted.values # Note that phat2 is identical to phat2a
logit.phat2 <- log(phat2 / (1 - phat2))
Ystar <- diag(logit.phat2)
VS1Ut <- Vstar %*% (solve(Sstar)) %*% t(Ustar)
mytry <- VS1Ut %*% Ystar
mytry <-as.vector(rowSums(mytry))
# mytry IS SAME AS coef(m2a)

# Adjoin Y-hat produced by the FULL (10-predictor) version:
Z.adj <- cbind(u187, Zprime, logit.phat2) # This is "Zstar" (about 17 lines up) with Yhat adjoined in last
column
svd.Za <- svd(Z.adj)
Ustar.Za <- svd.Za$u
Vstar.Za <- svd.Za$v
Sstar.Za <- diag(svd.Za$d)
rownames(Vstar.Za) <- c(names, "Yhat=logit.phat2")
# dim(Vstar.Za) is 11 x 11
h <- Vstar.Za[1:10,1:10] %*% Vstar.Za[11,1:10]
# cor(h, coef(m2a)) == 1 exactly

# IMPORTANT RESULT ==> logit.phat is an eigenvector of UUt
UUt <- Ustar %*% t(Ustar) # Ustar is from svd of cbind(u187, Zprime) --> see about 25 lines above
a <- UUt %*% logit.phat
# a is identical to logit.phat2. Therefore, logit.phat2 is an eigenvector of UUt, with eigenvalue 1.

# Explore cosines:

VVt.Za <- Vstar.Za[,1:10] %*% t(Vstar.Za[,1:10]) # Use all variables plus Yhat, AND all dimensions, 1 thru
10
d <- diag(VVt.Za)
d <- sqrt(d)
cosmat10 <- VVt.Za / outer(d,d)
h <- cosmat10[1:10,11] # cor(h, coef(m2a)) is +.9986
# Try this:
# eigplot(cosmat10)
# Also try using 3-d plot of first 3 eigenvectors of cosmat10
#
# Now look at rows:
Unorm <- Ustar.Za[,1:10] %*% Sstar.Za[1:10,1:10]
i <- Unorm %*% Vstar.Za[11,1:10]
# cor (i, logit.phat2) is 1 exactly. In fact, i is IDENTICAL to logit.phat2



# Two-dim version, for all the variables:
b.2dim <- Vstar[,1:2] %*% (solve(Sstar[1:2,1:2]) %*% t(Ustar[,1:2])) %*% logit.phat2
```

```
names <- c("u", "RussiaNIS", "MidEast","SouthAsia","is.bio","is.chem","perp.lone","perp.relex", "perp.cult",
"perp.ethnonat")
rownames(b.2dim)<-names
rownames(Vstar)<-names
phat.2dim <- Ustar[,1:2] %*% t(Ustar[,1:2]) %*% logit.phat2
Zadj <- cbind(Zstar, phat.2dim)
svd.Zadj <- svd(Zadj)
U.adj <- svd.Zadj$u
V.adj <- svd.Zadj$v
S.adj <- diag(svd.Zadj$d)
rownames(V.adj) <- c(names, "Y")
h <- V.adj[1:10, 1:2] %*% V.adj[11,1:2]
# cor(h, b.2dim) is 1 exactly
both <- rbind(U.adj, V.adj)
plotRB(both[,1], both[,2], labels=rownames(both))


# ===> DO THE SAME FOR J:
u187 <- matrix(1, 187, 1)
Jstar <- cbind(u187, Jprime)
svd.J <- svd(Jstar)
Uj <- svd.J$u
Vj <- svd.J$v
Sj <- diag(svd.J$d)
phatj <-m2j$fitted.values
logit.phatj <- log(phatj / (1 - phatj))
Yj <- diag(logit.phatj)
VS1Ut.j <- Vj %*% (solve(Sj)) %*% t(Uj)
mytry.j <- VS1Ut.j %*% Yj
mytry.j <- as.vector(rowSums(mytry.j))
# mytry.j is the same as coef(m2j)

# Two-dim. version, using only the 3 variables in J:
b.2 <- (Vj[,1:2] %*% solve(Sj[1:2,1:2]) %*% t(Uj[,1:2])) %*% logit.phatj
rownames(b.2) <- c("u", "is.chem", "perp.relex", "perp.ethnonat")
phat.2j <- Uj[,1:2] %*% t(Uj[,1:2]) %*% logit.phatj
Jadj <- cbind(Jstar, phat.2j)
svd.Jadj <- svd(Jadj)
U.Jadj <- svd.Jadj$u
V.Jadj <- svd.Jadj$v
rownames(V.Jadj) <- c("u","is.chem","perp.relex", "perp.ethnonat","Y")
```

```
S.Jadj <- diag(svd.Jadj$d)
h <- V.Jadj[1:4,1:2] %*% V.Jadj[5,1:2]
# cor(b.2, h) is 1 exactly
both <- rbind(U.Jadj, V.Jadj)
plotRB(both[,1], both[,2], labels=rownames(both))

HiSophist187 <- HighSophist[use.cases]
HiSophist187 <- HiSophist187 != 0
which.Hi <- which(HiSophist187 != 0)
which.Lo <- which(HiSophist187 == 0)
col.a <- rowSums(VS1Ut[,which.Lo] %*% Ystar[which.Lo, which.Lo])
col.b <- rowSums(VS1Ut[,which.Hi] %*% Ystar[which.Hi, which.Hi])
TABLE4 <-cbind(col.a, col.b)
rownames(TABLE4) <- names(coef(m2a))
colnames(TABLE4) <- c("SophLo", "SophHi")

# Plot it: First get a blank graph with corners (0, -1), (0, 1), (3, -1), (3,1)
# NOTE: This requires function: linesRB
plot(c(0,0,3,3), c(-1, 1, 1, -1), type="n")
for (i in 1:nrow(TABLE4)) {
  linesRB(c(1, TABLE4[i,1]), c(2, TABLE4[i,2]))
}

# WORK WITH COSINES:
UUtstar <- Ustar %*% t(Ustar)
d <- diag(UUtstar)
d <-sqrt(d)
COSstar <- UUtstar  / outer(d,d)
library(sna)
mycoord <- c(-20.14479,-19.21202,44.83537,-37.75267,37.36818,-
25.56892,35.99688,29.51316,36.28663,11.52812,33.73902,-
24.58958,39.47126,51.17357,33.07953,7.79261,30.28459,49.66746,38.92467,-16.98254,-22.58932,-
18.87523,-22.05191,9.85711,9.89825,29.99984,31.85572,34.49736,-45.91312,-
32.47355,47.78265,49.5978,34.88295,37.76811,44.99698,-
9.7064,12.96753,47.21731,28.91857,35.83492,32.24072,7.6963,51.23695,-
47.7837,45.39042,47.10198,45.57796,45.96372,44.90599,47.25341,46.4369,-49.13254,47.07053,-
49.87877,7.40048,-46.00227,-15.2268,7.26292,-9.3927,-16.8065,-11.60032,-12.95147,-
8.21381,6.46211,-13.48738,-14.71254,-31.79794,-13.65707,6.85464,-
28.08296,7.89778,14.93488,14.06743,5.21494,-7.08884,8.53425,9.76479,-18.42724,-12.95069,-
9.92625,-10.34502,-6.79258,13.71132,35.21729,11.01299,8.88605,-8.23138,-
10.99224,7.70899,12.32693,8.87971,-22.11386,-25.40907,-7.81205,13.97382,7.38578,10.56909,-
22.80111,13.22613,-28.45038,33.11492,34.21748,-7.3708,-23.98374,-13.15535,-
36.93572,12.00088,11.36354,6.57151,49.39958,-29.46948,-28.83174,4.36248,11.32325,-26.98753,-
16.05463,-35.68276,-6.90647,48.44326,50.04958,40.20827,12.29276,-6.55005,-26.69661,-
11.37848,51.07735,48.50086,47.71113,50.769,50.3225,49.2259,-37.58783,-
34.86307,4.32958,11.70272,-10.1208,12.5812,-7.22898,-36.34089,-24.21686,-12.33593,-29.06957,-
41.76171,4.95991,8.86205,13.83296,10.33817,14.09471,8.71348,14.78582,-7.68715,-8.78852,8.51391,-
6.11746,13.38205,32.98503,-23.77013,-12.53287,48.32994,39.81636,-50.8087,11.55041,10.75368,-
29.36416,6.10702,5.99319,-46.48181,-47.5808,-47.32206,-46.05616,-51.1394,-45.56341,-50.03698,-
49.23007,-49.42357,-22.9579,-22.44255,-2.60412,-9.10808,25.14103,-26.29639,32.43996,-20.9855,-
23.48761,-38.70871,-26.3691,-19.69149,31.03433,-0.53262,28.31052,-33.58388,-19.76975,-
0.96555,25.92515,-12.94827,-20.77377,-23.60503,-22.35291,16.80446,24.3326,-24.65469,-26.06864,-
19.80011,9.03255,30.28361,1.18429,1.34239,25.18219,32.40561,-3.52961,-39.64154,-
30.97188,2.2775,-22.50325,-21.51857,-19.25625,-35.04052,-3.82078,8.31536,-0.89243,-0.00508,-
4.33228,0.53873,0.32094,-3.67829,-1.57622,12.9886,-
4.79328,9.90325,48.9449,11.251,46.67022,41.6186,-13.77143,-13.3,-6.89094,-7.96661,-
12.99708,20.88729,-7.45562,40.43644,32.90096,-9.19778,19.6051,27.25288,-32.27673,-
33.70415,21.40757,46.8192,-45.28842,17.23431,-37.9865,45.20128,43.45996,-6.44007,-13.57571,-
9.0687,18.64538,-25.25277,42.89743,48.36618,-13.54156,-
6.19471,23.35997,16.97937,23.54489,30.44374,33.18735,-
7.22387,19.99276,18.02412,48.22398,28.21792,-36.64156,33.03259,26.73081,31.97503,-8.38933,-
27.23963,-10.67189,-8.82369,44.17226,16.8221,22.30187,9.77028,28.28248,-
8.42169,45.83498,24.129,34.14378,27.92407,-5.1706,-11.99846,-5.26126,-4.75021,28.83389,-29.3159,-
10.07267,28.12804,-13.30667,-2.64322,-0.26387,-5.06648,0.31514,-1.61356,-3.26102,-6.28689,-
3.75519,43.98315,45.30919,-44.95953,23.62685,-11.33188,-7.31047,-21.09875,-42.42276,-9.62894,-
0.90567,43.0716,-31.20289,-36.93419,-30.10188,-32.28602,-36.28143,-35.45735,-12.28904,-
6.74826,41.47617,-41.14464,22.59268,30.22218,-20.47798,-
6.93428,11.10672,26.72216,11.37038,47.14654,42.04651,-
9.57937,41.8595,48.00481,15.51931,14.23441,14.05332,12.54615,13.79725,14.06028,12.64682,10.997
2,15.33712)
mycoord <- matrix(mycoord, ncol=2)
whichA <- which(Event_type[use.cases] <=3)
whichB <- which(Event_type[use.cases] >=4)
mycol <-rep("blue", 187)
mycol[whichB] <- "palevioletred"
mysides <-rep(4, 187)
mysides[whichB]<-360
gplot(COSstar > .60, vertex.col=mycol, vertex.sides=mysides, edge.col="grey58", coord=mycoord,
main="Cos > . 60, blue squares = Type A [seek], red circles = Type B [possess]")

# Region, as used in Breiger-Murray-Pinson:
col.R <- rep("tan", 187)
col.R[MiddleEast[use.cases]] <- "red"
col.R[SouthAsia[use.cases]] <- "green"
col.R[RussiaNIS[use.cases]] <- "blue"

# Sophistication:
col.Soph <-rep("lightblue", 187)
col.Soph[which.Hi]<-"royalblue"
gplot(COSstar > .60, vertex.col=col.Soph, vertex.sides=mysides, edge.col="grey58", coord=mycoord,
main="Cos > . 60, Dark Blue = Highly Sophisticated, squares = Type A [seek], circles = Type B [possess]")

# Type of weapon (deals with multiples by preferring N > R > B > C):
col.T <- rep("black", 187)
col.T[is.chem[use.cases]] <- "tan"
col.T[is.bio[use.cases]] <- "green"
col.T[is.rad[use.cases]] <- "blue"
col.T[is.nuc[use.cases]] <- "red"

gplot(COSstar > .60, label=1:187, vertex.col=col.T, vertex.sides=mysides, edge.col="grey58",
coord=mycoord, main="Cos>.60, Chem(tan), Bio(green), Rad(blue), Nuc(red), squares=TypeA (seek),
circles=TypeB(possess)")

# Year: 1998-2001 (yellow), 2002-06 (light green), 2007-11 (dark green)
col.YR <-rep("yellow", 187)
w <-(IAPDATE >= 2002) & (IAPDATE <= 2006)
w <- w[use.cases]
col.YR[w] <- "green3"
w <- (IAPDATE[use.cases] > 2006)
col.YR[w] <- "darkgreen"

# Find components:
comps <- component.dist((COSstar > .60), connected="weak")
comp.sizes <- comps$csize
comp.membership <- comps$membership

# Clusters: paperN means the cluster labeled N in the Figures in Breiger-Murray-Pinson

# Paper Cluster # 1
paper1 <- which(comp.membership==5) # Index numbers out of 187: 10, 16, 37, 42, ...
xlate1 <- use.cases[paper1]

# Paper Cluster # 2 == rad, nuc
```

```
paper2 <- which(comp.membership==4) # Index numbers out of 187: 8, 9, 11, 17, ...
xlate2 <- use.cases[paper2] # Index numbers out of 471: 21, 23, 25, 35, ...
not.paper2 <-rep(1,187)
not.paper2[paper2] <- 0
not.paper2 <- which(not.paper2 == 1)
ppp2yes <- rowSums(VS1Ut[,paper2] %*% Ystar[paper2,paper2])
ppp2no  <- rowSums(VS1Ut[,not.paper2] %*% Ystar[not.paper2, not.paper2])
show2 <-cbind(ppp2no, ppp2yes)
rownames(show2) <- names(coef(m2a))
colnames(show2) <- c("NOTpaper2", "paper2")

# Middle-Eastern cluster (Paper cluster # 3)
paper3 <-which(comp.membership==2) # Index numbers out of 187: 3, 14, 18, 31, ...
xlate3 <-use.cases[paper3] # Index numbers out of 471: 13, 30, 37, 53, ...
not.paper3 <- rep(1, 187)
not.paper3[paper3] <- 0
not.paper3 <- which(not.paper3 == 1)
pp3yes <- rowSums(VS1Ut[,paper3] %*% Ystar[paper3, paper3])
pp3no <- rowSums(VS1Ut[,not.paper3] %*% Ystar[not.paper3,not.paper3])
show3 <- cbind(pp3no, pp3yes)
rownames(show3) <- names(coef(m2a))
colnames(show3) <- c("NOTpaper3", "paper3")

# Cluster 3b: Delete the two hangers-on (#110, #159) and also the two furthest away from them (#119,
#120)
paper3b <- paper3
w1 <- which(paper3b == 110)
w2 <- which(paper3b == 159)
w3 <- which(paper3b == 119)
w4 <- which(paper3b == 120)
paper3b[c(w1,w2,w3,w4)] <- 0
paper3b <- paper3b[paper3b > 0]
xlate3b <- use.cases[paper3b]
not.paper3b <- rep(1, 187)
not.paper3b[paper3b] <- 0
not.paper3b <- which(not.paper3b == 1)
pp3byes <- rowSums(VS1Ut[,paper3b] %*% Ystar[paper3b, paper3b])
pp3bno <- rowSums(VS1Ut[,not.paper3b] %*% Ystar[not.paper3b, not.paper3b])
show3b <- cbind(pp3bno, pp3byes)
rownames(show3b) <- names(coef(m2a))
colnames(show3b) <- c("NOTpaper3b", "paper3b")
```

```
# Cluster # 4
paper4 <- which(comp.membership == 3) # Index numbers out of 187: 5, 7, 13, 15
xlate4 <- use.cases[paper4] # Index numbers out of 438: 18, 20, 28, 31

# Cluster # 5 :: other region, chemical
paper5 <- which(comp.membership == 6) # Index numbers out of 187: 24, 25, 64, 69
xlate5 <- use.cases[paper5] # Index numbers out of 438: 44, 45, 126, 135,
not.paper5 <- rep(1, 187)
not.paper5[paper5] <- 0
not.paper5 <- which(not.paper5 == 1)
pp5yes <- rowSums(VS1Ut[,paper5] %*% Ystar[paper5, paper5])
pp5no <- rowSums(VS1Ut[,not.paper5] %*% Ystar[not.paper5, not.paper5])
show5 <- cbind(pp5no, pp5yes)
rownames(show5) <- names(coef(m2a))
colnames(show5) <- c("NOTpaper5", "paper5")


# resLone with Cluster 5:
resLone <- lm(Zstar[,7] ~ Zstar[,c(2:6, 8:10)])$residuals
try.it <- resLone / sum(resLone ^2) # Identical to VS1Ut[7,]
this.phat <- m2a$fitted.values
logit.this.phat <- log(this.phat / (1 - this.phat))
m.try <- lm(logit.this.phat ~ resLone)
#   coef(m.try) == (mean of logit.this.phat, mult. reg. slope) == (.24906, -.2327)
mycol <- rep("tan", 187)
mycol[paper5] <- "forestgreen"
np <- find.number.of.points(resLone, logit.this.phat)
plot(resLone, logit.this.phat, col=mycol, pch=16, cex= (np/5), xlab="resLone", ylab="logit.this.phat",
main="Blue = reg. = -.2327; Cluster 5 (contrib. to reg = +.2582)")
abline(coef(m.try), col="red", lty="dashed")
abline(mean(logit.this.phat),0, col="lightgrey")
abline(0, 5000, col="lightgrey")
# NOTE!!! mult. reg. coef. == -.2327 ==
#   sum(resLone * logit.this.phat) / sum(resLone ^2)
# AND the contribution of cluster "paper5" to that coefficient == +.2528 ==
#   sum(resLone[paper5] * logit.this.phat[paper5]) / sum(resLone ^2)
# Cluster # 6
paper6 <- which(comp.membership == 9) # Index numbers out of 187: 55, 58, 74, 85, ...
xlate6 <- use.cases[paper6] # Index numbers out of 438: 108, 118, 154, 189, ...

# Cluster 9 :: Type B, 2007-11, South Asia, chemical
paper9 <- c(29,44,52,54,56,161,167,168,169,170,171,172,173,174,187)
xlate9 <- use.cases[paper9]
not.paper9 <- rep(1, 187)
not.paper9[paper9] <- 0
not.paper9 <- which(not.paper9 == 1)
pp9yes <- rowSums(VS1Ut[,paper9] %*% Ystar[paper9, paper9])
pp9no <- rowSums(VS1Ut[,not.paper9] %*% Ystar[not.paper9,not.paper9])
show9 <- cbind(pp9no, pp9yes)
rownames(show9) <- names(coef(m2a))
colnames(show9) <- c("NOTpaper9", "paper9")

# Cluster 9b: All in cluster 9 except for the two events (#29 and #44) with direct ties to cluster 10
paper9b <- paper9[3:15]
xlate9b <- use.cases[paper9b]
not.paper9b <- rep(1, 187)
not.paper9b[paper9b] <- 0
not.paper9b <- which(not.paper9b == 1)
pp9byes <- rowSums(VS1Ut[,paper9b] %*% Ystar[paper9b, paper9b])
pp9bno <- rowSums(VS1Ut[,not.paper9b] %*% Ystar[not.paper9b, not.paper9b])
show9b <- cbind(pp9bno, pp9byes)
rownames(show9b) <- names(coef(m2a))
colnames(show9b) <- c("NOTpaper9b", "paper9b")

# resRelex using Zstar for Cluster paper9b
resRelex <- lm(Zstar[,8] ~ Zstar[,c(2:7, 9:10)])$residuals
try.it <- resRelex / sum(resRelex ^2) # Identical to VS1Ut[8,]
this.phat <- m2a$fitted.values
logit.this.phat <- log(this.phat / (1 - this.phat))
m.try <- lm(logit.this.phat ~ resRelex)
#   coef(m.try) == (mean of logit.this.phat, mult. reg. slope) == (.24906, -1.2733)
mycol <- rep("tan", 187)
mycol[paper9b] <- "blue"
np <- find.number.of.points(resRelex, logit.this.phat)
plot(resRelex, logit.this.phat, col=mycol, pch=16, cex= (np/5), xlab="resRelex", ylab="logit.this.phat",
main="Blue = reg. = -1.2733; Cluster 9b (contrib. to reg = +.1103)")
abline(coef(m.try), col="red", lty="dashed")
abline(mean(logit.this.phat),0, col="lightgrey")
abline(0, 5000, col="lightgrey")
# NOTE!!! mult. reg. coef. == -1.2733 ==
#   sum(resRelex * logit.this.phat) / sum(resRelex ^2)
# AND the contribution of cluster "paper9b" to that coefficient == +.1103 ==
#   sum(resRelex[paper9b] * logit.this.phat[paper9b]) / sum(resRelex ^2)

# resLone with Cluster 9b:
resLone <- lm(Zstar[,7] ~ Zstar[,c(2:6, 8:10)])$residuals
try.it <- resLone / sum(resLone ^2) # Identical to VS1Ut[7,]
this.phat <- m2a$fitted.values
logit.this.phat <- log(this.phat / (1 - this.phat))
m.try <- lm(logit.this.phat ~ resLone)
#   coef(m.try) == (mean of logit.this.phat, mult. reg. slope) == (.24906, -.2327)
mycol <- rep("tan", 187)
mycol[paper9b] <- "blue"
np <- find.number.of.points(resLone, logit.this.phat)
plot(resLone, logit.this.phat, col=mycol, pch=16, cex= (np/5), xlab="resLone", ylab="logit.this.phat",
main="Blue = reg. = -.2327; Cluster 9b (contrib. to reg = +.0475)")
abline(coef(m.try), col="red", lty="dashed")
abline(mean(logit.this.phat),0, col="lightgrey")
abline(0, 5000, col="lightgrey")
# NOTE!!! mult. reg. coef. == -.2327 ==
#   sum(resLone * logit.this.phat) / sum(resLone ^2)
# AND the contribution of cluster "paper9b" to that coefficient == +.0475 ==
#   sum(resLone[paper9b] * logit.this.phat[paper9b]) / sum(resLone ^2)

# Look at three clusters: {the rest}, {5}, {9b}
the.rest <- rep(1, 187)
the.rest[paper5] <- 0
the.rest[paper9b] <- 0
the.rest <- which(the.rest != 0)
pptherest <- rowSums(VS1Ut[,the.rest] %*% Ystar[the.rest, the.rest])
show.5.9b <- cbind(pptherest, pp5yes, pp9byes)
rownames(show.5.9b) <- names(coef(m2a))
colnames(show.5.9b) <- c("the.rest", "paper5", "paper9b")

# Cluster 10
paper10 <- c(4,106,112,117,132,133,139,142,143,164)
xlate10 <- use.cases[paper10]
```

```
# Cluster 11
paper11 <- c(20,59,60,61,62,63,65,68,80,81,82,87,88,94,103,105,118,123,125,138,151,152,158)
xlate11 <- use.cases[paper11]

# Cluster 12
paper12 <- c(1,2,6,12,21,22,23,104,140,157)
xlate12 <- use.cases[paper12]

mycol <- rep("black", 187)
mycol[paper1] <- "red"
mycol[paper2] <- "white"
mycol[paper3] <- "forestgreen"
mycol[paper5]<- "orange"
mycol[paper6] <- "midnightblue"
mycol[paper9] <- "red"
mycol[paper10] <- "blue"
mycol[paper11] <- "green"
mycol[paper12] <- "pink"
mylabs <-rep("", 187)
mylabs[29] <-"29"
gplot(COSstar > .60, vertex.col=mycol, vertex.cex=1.3, edge.col="grey58", coord=mycoord, main="Cos>.60,
some clusters shown by color", label=mylabs, interactive=TRUE)

# The following plot shows interaction of Chem & Bio cases within the Middle East cluster. (Note: case 38
is chem + bio)
my.col<-col.T # (weapons coding)
my.col[38]<-"orange" # Case 38 is chem + bio
gplot(COSstar[paper3,paper3] > .60, vertex.col=my.col[paper3], vertex.cex=2, label.cex=.85, label.pos=5,
label=paper3, main="Middle East Cluster, tan=Chem, green = Bio, orange = both")

Xint <- cbind(RussiaNIS, MiddleEast, SouthAsia, is.bio, is.chem, perp.lone, perp.relex, perp.cult,
perp.ethnonat, RussiaNIS*is.bio)
Zint <- scale(Xint)
mint <- glm((Event_type >= 4)[use.cases] ~ Zint[use.cases,], family=binomial())

u <-rep(1,471)
Z1 <-cbind(u,Z)
svdz <- svd(Z1[use.cases,])
Uz <- svdz$u
Vz <- svdz$v
Sz <- diag(svdz$d)
```

```
A <- Vz %*% solve(Sz) %*% t(Uz) # size is 10 x 187
phat <- m2$fitted.values
logit.phat <- log(phat / (1-phat))
logit.phat.minus.mean <- logit.phat - mean(logit.phat)
bs.z <- A %*% diag(as.vector(logit.phat.minus.mean)) # size is 10 x 187
# Note: rowSums(bs.z) are IDENTICAL to coef(m2) !!!

resRuss <- lm(Z1[use.cases,2] ~ Z1[use.cases, 3:10])$residuals # include an intercept term
cor(A[2,], resRuss) # == 1 !!!
try.it <- resRuss / sum(resRuss^2) # try.it is IDENTICAL to A[2,] !!!
m.try <- lm(logit.phat.minus.mean ~ resRuss)
# coef(m.try) has intercept = ZERO and slope = 6.5682, identical to the mult. reg. coef. from m2 !!!
roundY <- round(logit.phat.minus.mean, 2)
roundX <- round(resRuss, 2)
t <- table(roundY)
plot(roundX, roundY, cex=(.1*t))
abline(coef(m.try))
mycol <- rep("black",187)
w <- which(perp.ethnonat[use.cases] == 1)
mycol[w] <- "red"
plot(logit.phat.minus.mean, resRuss, col=mycol)
abline(coef(m.try), col="blue")

resMidE <- lm(Z1[use.cases,3] ~ Z1[use.cases, c(2,4:10)])$residuals
try.it <- resMidE / sum(resMidE^2) # Identical to A[3,]
m.try <- lm(logit.phat.minus.mean ~ resMidE) # intercept 0, slope = Mult Reg coef from "m2"
mycol <-rep("black",187)
w <- which(is.chem[use.cases] == 1)
mycol[w] <- "red"
plot(resMidE, logit.phat.minus.mean, main="for res.MidE, is.chem in Red", col=mycol)
abline(coef(m.try), col="blue", lty="dashed")

resSoAsia <- lm(Z1[use.cases,4] ~ Z1[use.cases, c(2:3, 5:10)])$residuals
try.it <- resSoAsia / sum(resSoAsia ^2) # Identical to A[4,]
m.try <- lm(logit.phat.minus.mean ~ resSoAsia) # intercept 0, slope Mult Reg coef.
plot(resSoAsia, logit.phat.minus.mean, main="for res.SoAsia")
abline(coef(m.try), col="green")

# South Asia with INTEREST:
resSoAsia <- lm(Z1[use.cases,4] ~ Z1[use.cases, c(2:3, 5:10)])$residuals
try.it <- resSoAsia / sum(resSoAsia ^2) # Identical to A[4,]
```

```
m.try <- lm(logit.phat.minus.mean ~ resSoAsia) # intercept 0, slope Mult Reg coef.
mycol <-rep("blue", 187)
mycol[interest.hi] <- "red"
plot(resSoAsia, logit.phat.minus.mean, pch=16, col=mycol, main="for res.SoAsia, red = HIGH-INTEREST")
abline(coef(m.try), col="green")

resBio <- lm(Z1[use.cases, 5] ~ Z1[use.cases, c(2:4, 6:10)])$residuals
try.it <- resBio / sum(resBio ^2) # Identical to A[5,]
m.try <- lm(logit.phat.minus.mean ~ resBio) # intercept 0, slope Mult Reg coef
plot(resBio, logit.phat.minus.mean, main = "for resBio")
abline(coef(m.try), col="purple")

# resBio using Zstar for cluster paper2 ***************************************************
resBio <- lm(Zstar[, 5] ~ Zstar[, c(2:4, 6:10)])$residuals
try.it <- resBio / sum(resBio ^2) # Identical to VS1Ut[5,]
this.phat <- m2a$fitted.values
logit.this.phat <- log(this.phat / (1 - this.phat))
logit.this.phat.minus.mean <- logit.this.phat - mean(logit.this.phat)
m.try.minus.mean <- lm(logit.this.phat.minus.mean ~ resBio)
#       coef(m.try.minus.mean) == (0, mult. reg. slope of -.0993)
m.try <- lm(logit.this.phat ~ resBio)
#       coef(m.try) == (mean of logit.this.phat, mult. reg.slope) == (.24906, -.0993)
mycol <-rep("black", 187)
mycol[paper2] <- "red"
plot(resBio, logit.this.phat, col=mycol, pch=16, xlab="resBio", ylab="logit.this.phat")
abline(coef(m.try), col="blue", lty="dashed")
abline(mean(logit.this.phat),0,col="lightgrey")
abline(0, 5000, col="lightgrey")
# NOTE!!! mult. reg. coef. == -.0993 ==
#   (sum(resBio * logit.this.phat)) / sum(resBio ^ 2)
# AND the contribution of cluster "paper2" to that coefficient == +.2670 ==
#   (sum(resBio[paper2] * logit.this.phat[paper2])) / sum(resBio ^ 2)

# resRelex using Zstar for Cluster paper2
resRelex <- lm(Zstar[,8] ~ Zstar[,c(2:7, 9:10)])$residuals
try.it <- resRelex / sum(resRelex ^2) # Identical to VS1Ut[8,]
this.phat <- m2a$fitted.values
logit.this.phat <- log(this.phat / (1 - this.phat))
m.try <- lm(logit.this.phat ~ resRelex)
#   coef(m.try) == (mean of logit.this.phat, mult. reg. slope) == (.24906, -1.2733)
mycol <- rep("black", 187)
```

```
mycol[paper2] <- "red"
plot(resRelex, logit.this.phat, col=mycol, pch=16, xlab="resRelex", ylab="logit.this.phat")
abline(coef(m.try), col="blue", lty="dashed")
abline(mean(logit.this.phat),0, col="lightgrey")
abline(0, 5000, col="lightgrey")
# NOTE!!! mult. reg. coef. == -1.2733 ==
#   sum(resRelex * logit.this.phat) / sum(resRelex ^2)
# AND the contribution of cluster "paper2" to that coefficient == -.3838 ==
#   sum(resRelex[paper2] * logit.this.phat[paper2]) / sum(resRelex ^2)

# resRelex using Zstar for Several Clusters
resRelex <- lm(Zstar[,8] ~ Zstar[,c(2:7, 9:10)])$residuals
try.it <- resRelex / sum(resRelex ^2) # Identical to VS1Ut[8,]
this.phat <- m2a$fitted.values
logit.this.phat <- log(this.phat / (1 - this.phat))
m.try <- lm(logit.this.phat ~ resRelex)
#   coef(m.try) == (mean of logit.this.phat, mult. reg. slope) == (.24906, -1.2733)
mycol <- rep("black", 187)
mycol[paper3] <- "red"
mycol[paper9] <- "blue"
mycol[paper5] <- "pink"
mycol[paper6] <- "green"
np <- find.number.of.points(resRelex, logit.this.phat)
plot(resRelex, logit.this.phat, col=mycol, pch=16, xlab="resRelex, Cluster 3", ylab="logit.this.phat",
cex=(np/5), main = "red 3 / blue 9 / pink 5/ green 6")
abline(coef(m.try), col="blue", lty="dashed")
abline(mean(logit.this.phat),0, col="lightgrey")
abline(0, 5000, col="lightgrey")

# Try it without points numbered 110 and 159 (Cluster3main , relex)
paper3main <- paper3
paper3main[which(paper3==159)] <- 0
paper3main[which(paper3==110)] <- 0
paper3main <- paper3main[paper3main > 0]
resRelex <- lm(Zstar[,8] ~ Zstar[,c(2:7, 9:10)])$residuals
try.it <- resRelex / sum(resRelex ^2) # Identical to VS1Ut[8,]
this.phat <- m2a$fitted.values
logit.this.phat <- log(this.phat / (1 - this.phat))
m.try <- lm(logit.this.phat ~ resRelex)
#   coef(m.try) == (mean of logit.this.phat, mult. reg. slope) == (.24906, -1.2733)
mycol <- rep("black", 187)
```

```
mycol[paper3main] <- "purple"
np <- find.number.of.points(resRelex, logit.this.phat)
plot(resRelex, logit.this.phat, col=mycol, pch=16, xlab="resRelex, Cluster3-MAIN", ylab="logit.this.phat",
cex=(np/2), main="Cluster3-MAIN")
abline(coef(m.try), col="blue", lty="dashed")
abline(mean(logit.this.phat),0, col="lightgrey")
abline(0, 5000, col="lightgrey")




# resBio with interest
resBio <- lm(Z1[use.cases, 5] ~ Z1[use.cases, c(2:4, 6:10)])$residuals
try.it <- resBio / sum(resBio ^2) # Identical to A[5,]
m.try <- lm(logit.phat.minus.mean ~ resBio) # intercept 0, slope Mult Reg coef
mycol<-rep("blue", 187)
mycol[interest.hi]<-"red"
plot(resBio, logit.phat.minus.mean, col=mycol, pch=16, main = "resBio - High Interest = red")
abline(coef(m.try), col="purple")


resChem <- lm(Z1[use.cases, 6] ~ Z1[use.cases, c(2:5, 7:10)])$residuals
try.it <- resChem / sum(resChem^2) # Identical to A[6,]
m.try <- lm(logit.phat.minus.mean ~ resChem)
w <- which(Event_type[use.cases] >= 4)
mycol <- rep("black", 187)
mycol[w] <- "red"
plot(resChem, logit.phat.minus.mean, main="for resChem, in red is Observed WEAPONS", col=mycol)
abline(coef(m.try), col="midnight blue")

# CHEM WITH DOUBT:
resChem <- lm(Z1[use.cases, 6] ~ Z1[use.cases, c(2:5, 7:10)])$residuals
try.it <- resChem / sum(resChem^2) # Identical to A[6,]
m.try <- lm(logit.phat.minus.mean ~ resChem)
mycol <- rep("blue", 187)
mycol[doubt.yes] <- "red"
plot(resChem, logit.phat.minus.mean, pch=16, main="for resChem, red is HIGH-DOUBT", col=mycol)
abline(coef(m.try), col="midnight blue")

# CHEM WITH INTEREST:
```

```
resChem <- lm(Z1[use.cases, 6] ~ Z1[use.cases, c(2:5, 7:10)])$residuals
try.it <- resChem / sum(resChem^2) # Identical to A[6,]
m.try <- lm(logit.phat.minus.mean ~ resChem)
mycol <- rep("blue", 187)
mycol[interest.hi] <- "red"
plot(resChem, logit.phat.minus.mean, pch=16, main="for resChem, in red is HIGH-INTEREST", col=mycol)
abline(coef(m.try), col="midnight blue")

# loner

resRelEx <- lm(Z1[use.cases, 8] ~ Z1[use.cases, c(2:7, 9:10)])$residuals
try.it <- resRelEx / sum(resRelEx^2) # Identical to A[8,]
m.try <- lm(logit.phat.minus.mean ~ resRelEx)
w <- which(Event_type[use.cases] >= 4)
mycol <- rep("black", 187)
mycol[w] <- "red"
plot(resRelEx, logit.phat.minus.mean, type = "n", col=mycol, main="for RelEx, with Observed WEAPONS in
red")
text(resRelEx, logit.phat.minus.mean, labels=use.cases)
abline(coef(m.try), col="brown")

# RelEx with INTEREST
resRelEx <- lm(Z1[use.cases, 8] ~ Z1[use.cases, c(2:7, 9:10)])$residuals
try.it <- resRelEx / sum(resRelEx^2) # Identical to A[8,]
m.try <- lm(logit.phat.minus.mean ~ resRelEx)
w <- which(Event_type[use.cases] >= 4)
mycol <- rep("blue", 187)
mycol[interest.hi] <- "red"
plot(resRelEx, logit.phat.minus.mean, pch=16, col=mycol, main="for RelEx, with HIGH-INTEREST in red")
abline(coef(m.try), col="brown")


resCult <- lm(Z1[use.cases,9] ~ Z1[use.cases, c(2:8, 10)])$residuals
try.it <- resCult / sum(resCult^2) # Identical to A[9,]
m.try <- lm(logit.phat.minus.mean ~ resCult)
plot(resCult, logit.phat.minus.mean, main="for resCult, yuch")
abline(coef(m.try), col="pink")

resEthNat <- lm(Z1[use.cases, 10] ~ Z1[use.cases, 2:9])$residuals
try.it <- resEthNat / sum(resEthNat^2) # Identical to A[10,]
m.try <- lm(logit.phat.minus.mean ~ resEthNat)
```

```
plot(resEthNat, logit.phat.minus.mean, main="for resEthNat")
abline(coef(m.try), col="orange")

# EthNat with INTEREST
resEthNat <- lm(Z1[use.cases, 10] ~ Z1[use.cases, 2:9])$residuals
try.it <- resEthNat / sum(resEthNat^2) # Identical to A[10,]
m.try <- lm(logit.phat.minus.mean ~ resEthNat)
mycol <- rep("blue", 187)
mycol[interest.hi] <- "red"
plot(resEthNat, logit.phat.minus.mean, pch=16, col=mycol, main="for resEthNat")
abline(coef(m.try), col="orange")

# Try kmeans on Uz:
K2 <- kmeans(Uz, centers=2, iter.max=10000, nstart=10000)
K3 <- kmeans(Uz, centers=3, iter.max=10000, nstart=10000)
K4 <- kmeans(Uz, centers=4, iter.max=10000, nstart=10000)
K5 <- kmeans(Uz, centers=5, iter.max=10000, nstart=10000)


X2 <- cbind(is.bio, is.chem, is.nuc)
Y <- (Event_type >=3)[use.cases]
Y <- Y + 0
myX <- X2[use.cases,]
myX <- myX + 0
m1 <- glm(Y ~ myX, family=binomial())

X3 <- cbind(is.bio, is.chem, is.rad)
myX3 <- X3[use.cases,]
myX3 <- myX3 + 0
m2 <- glm(Y ~ myX3, family=binomial())
# use <- which(credibility >= 2)
# Omit 1-issue:
# m1 <- glm(casualties[use] ~ perp.lone[use]  + perp.cult[use] + perp.relex[use]  + perp.rightwing[use] +
perp.leftwing[use] + perp.ethnonat[use], family=binomial())

# m2 <- glm(casualties ~ perp.cult + perp.relex + perp.rightwing + perp.leftwing + perp.ethnonat)
```

RECODE FUNCTIONS

```
# LOAD POICN2014.RData:

# Make RECODE program:
recode <- function(v,old,new) {
  # recode(v,old,new)
  # replaces each occurrence of "old" in
  #    vector v with the value "new".
  # NA can be a value for "new" but NOT
  #   for "old".
  # Has the virtue of working even if
  #   some values of v are NA's.
  N<-length(v)
  for (i in 1:N) {
    if (is.na(v[i])) i<-i+1
    else if
    (v[i]==old) v[i]<-new }
  return(v)
}

plotRB<-function(x,y,labels="",labelcex=0.6,pointcex=1,pos=1,offset=0.2,  pch=21,  type="p",  xlab="Dim  1",
ylab="Dim 2", main="Title of Plot",sub="", labelcolor="black", myfont=1, col="black", aspect=1) {

# This [aspect=1] sets aspect to equal axes.
# Alternatively, use aspect=NA

plot.new() # Clears the existing screen
plot(x,y,xlab=xlab,ylab=ylab,main=main,sub=sub,type=type,pch=pch, col=col, cex=pointcex,asp=aspect)
# Set axes to equal aspects
# type="n" for no printing of points

# pch can take values 0 through 18
# pch=19: solid circle,
# pch=20: bullet (smaller circle),
# pch=21: circle,
# pch=22: square,
# pch=23: diamond,
# pch=24: triangle point-up,
# pch=25: triangle point down.
# myfont=1: plain text
# myfont=2: bold text
text(x,y,labels=labels,cex=labelcex,pos=pos,offset=offset,col=labelcolor, font=myfont)
# pos=1 for printing labels below points; =2 for labels to the left of points;
#   =3 for labels above points; =4 for labels to the right of points.
# offset = offset of label from point, in fractions of a character.

abline(h=0,col="darkgray")   # lines through the origin
abline(v=0,col="darkgray")

#identify(d1,d2,labels=nn)    # Use mouse button to identify points

# Use:  locator(1,"p") to locate the (x,y) coordinates of 1 point.
}
linesRB <- function(point1,point2,color="black",type="solid") {
x1 <- point1[1]
y1 <- point1[2]

x2 <- point2[1]
y2 <- point2[2]

XX <- c(x1,x2)
YY <- c(y1,y2)

lines(XX,YY,lty=type,col=color)
}
find.number.of.points <- function(X, Y) {
 # X and Y are same-length vectors -- typically, points to be plotted
 # "out" is the NUMBER OF POINTS that have each X,Y combination.
 # Therefore, you can run the "plot" program as, e.g.,
 #    plot(X,Y, cex= (out/7))  # where 7 is arbitary scaling constant.
 n <- length(X)
 out <-rep(0, n)
 for (i in 1:n) {
  Xi <- X[i]; Yi <- Y[i]
  t <- sum((X == Xi) & (Y == Yi))
  out[i] <- t
 }
 return(out)
}
```