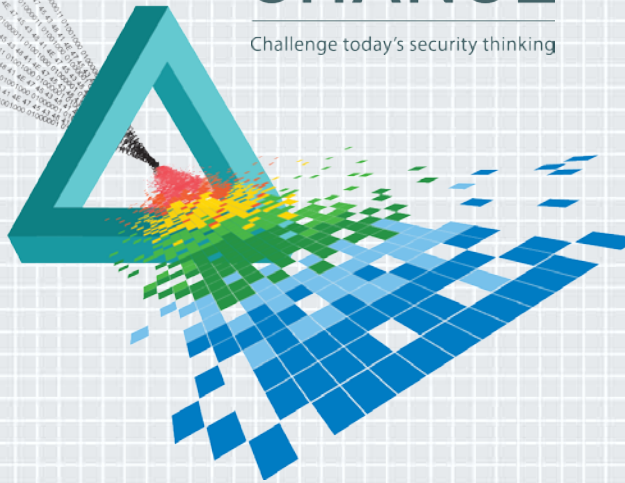
RSAConference2015
San Francisco | April 20-24 | Moscone Center

CHANGE
Challenge today's security thinking



SESSION ID: HTA-T09

Structural Entropy Analysis for Automated Malware Classification

Matt Wolff and Glenn Chisholm

Chief Data Scientist • Chief Technology Officer
Cylance, Inc.
@cylanceinc

Overview: The Problem

- Malware can obfuscate its purpose by encrypting or compressing code segments within the PE structure.
- First-order obfuscation techniques (encryption and compression) can disguise the malicious commands so that they pass through traditional "signature-based detection" of anti-virus programs.
- Meta-obfuscation techniques (e.g. null content insertion) can then disguise the encryption and compression so that the file passes through entropy filters.

Overview: The Contribution

- We describe a technique called **structural entropy analysis** which can help to detect the presence of suspicious patterns of obfuscation.
- Structural entropy analysis goes beyond mean entropy analysis and mere “detection of packing” to find suspicious *patterns* of entropy change.
  - It quantifies the "amount of structure" in the entropy signal; finds the location of mean change points; and computes the distribution of energy across multiple spatial scales.
  - It then derives 387 mathematical features which are jointly predictive of the file being malware.
- Structural entropy is much harder for a malware writer to control than commonly used measures such as mean entropy alone.
- SE performs well (87.2% accuracy) on predicting malicious behavior for completely novel files "in the wild.” Moreover, it can easily be combined with other types of features.

Apply Slide: Relevance to Audience

- For Architects & Consumers of Machine Learning-Based Anti-Malware Systems:
  - We illustrate the usefulness of incorporating 520 structural entropy features within a machine learning classifier.
- For Architects & Consumers of Detection-Based Anti-Malware Systems:
  - We provide an in-depth example of how "soft-constraint" features can be built for a machine-learning type system.

Mean entropy: A first clue to packing

High entropy is a symptom of "packing" (compression or encryption)

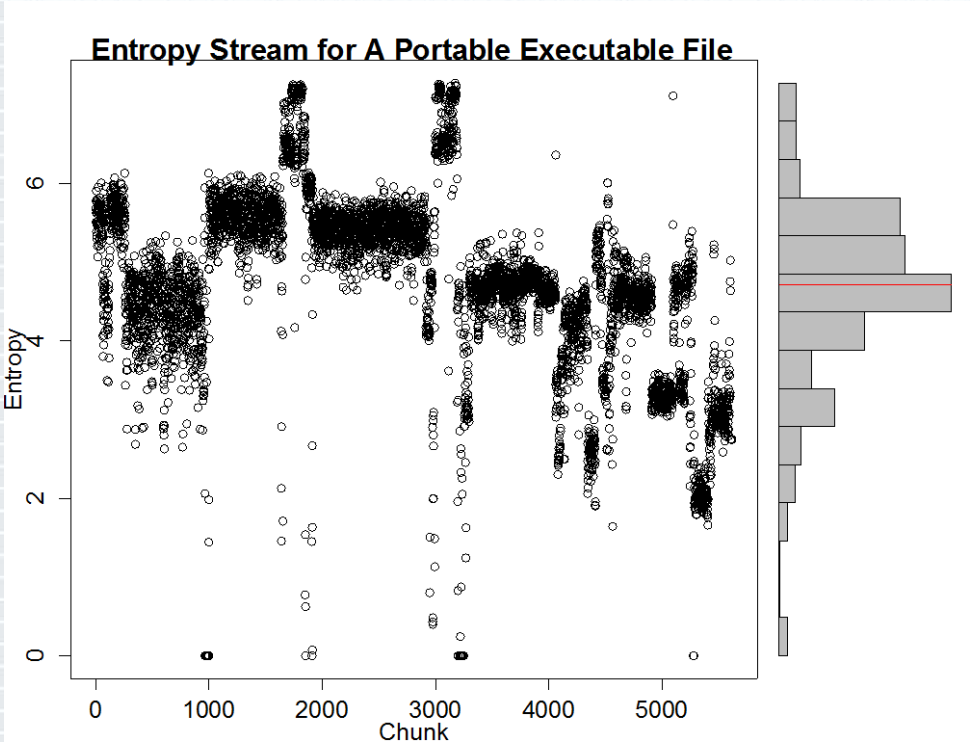
Table 1. Computed statistical measures based on four training sets.

DATA SETS	AVERAGE ENTROPY	99.99% CONFIDENCE INTERVALS (LOW TO HIGH)	HIGHEST ENTROPY (AVERAGE)	99.99% CONFIDENCE INTERVALS (LOW TO HIGH)
Plain text	4.347	4.066 – 4.629	4.715	4.401 – 5.030
Native executables	5.099	4.941 – 5.258	6.227	6.084 – 6.369
Packed executables	6.801	6.677 – 6.926	7.233	7.199 – 7.267
Encrypted executables	7.175	7.174 – 7.177	7.303	7.295 – 7.312

- Entropy calculated over neighboring bins of 256 bytes
- Minimum possible entropy = 0 bits; maximum possible entropy = 8 bits

Lyda, R., & Hamrock, J. (2007). Using entropy analysis to find encrypted and packed malware. IEEE Security & Privacy, (2), 40-45.

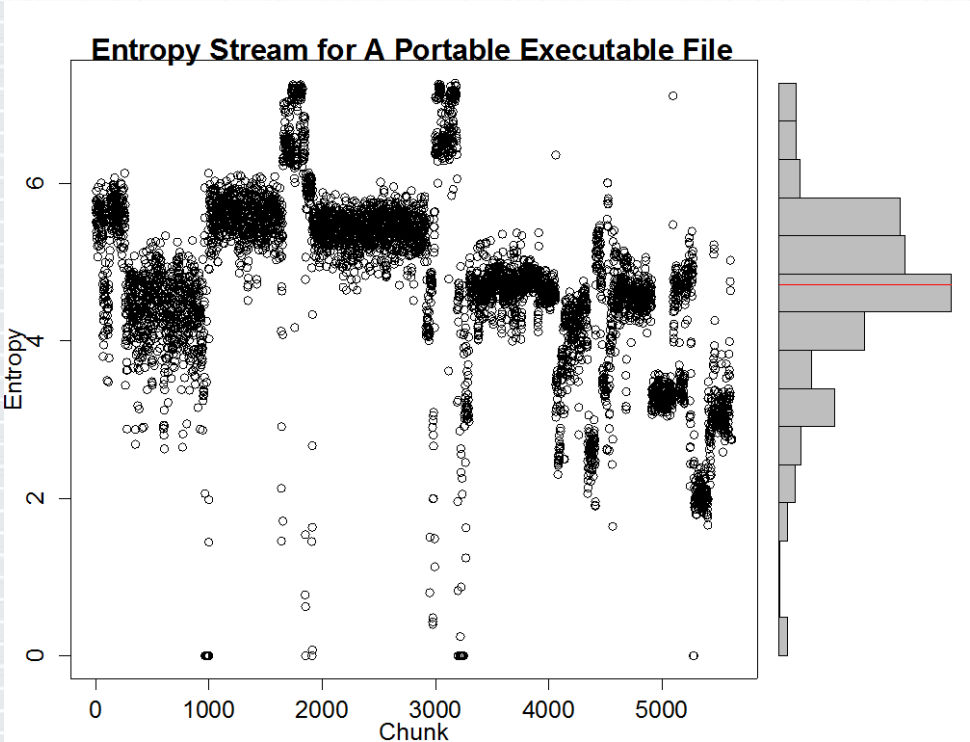
Structural entropy: "Time series" approach



- We treat the file as an **entropy time series.** We want to take into account sequential structure

- The entropy measurements, taken at adjacent points in the code, are not independent and identically distributed, but have strong spatial correlations (based on proximity)

Structural entropy: "Time series" approach

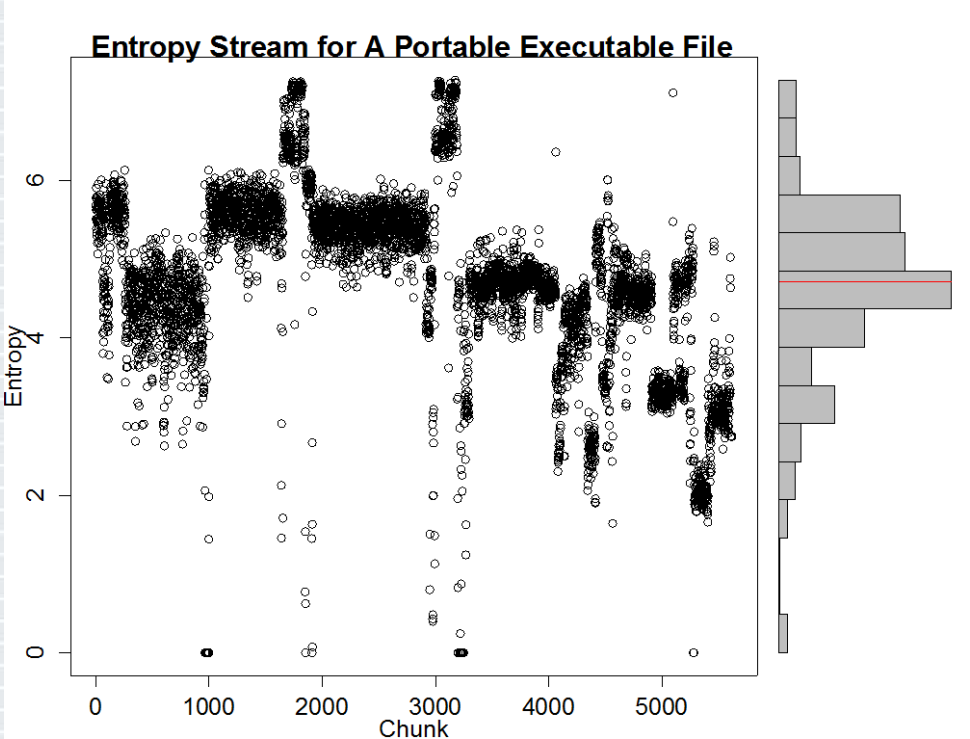


- The executable file's raw machine code (in hex) is split into non-overlapping chunks of fixed length, typically 256 bytes.

$$H(c) = -\sum_{i=1}^{n} p_i(c) \log_2 p_i(c),$$

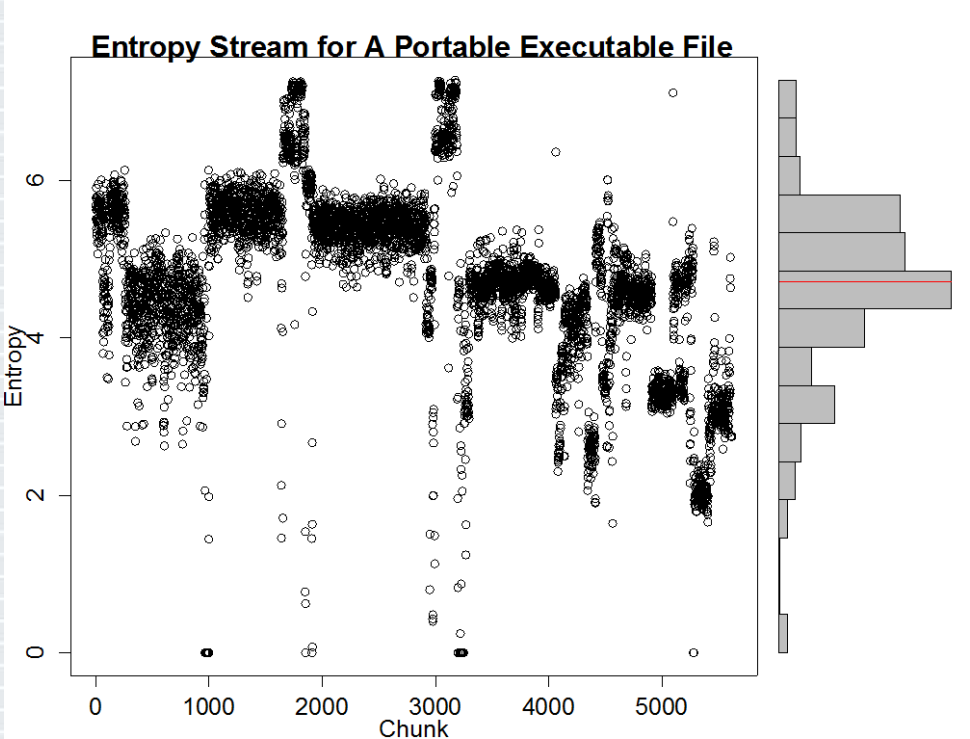
- Where
  - H is entropy
  - c is a chunk of code
  - n is the number of possible characters (here n=256)
  - p_i is the probability of each character in the chunk

Nonstationary Time Series: Which tools can we use?



- Problem: This is a **non-stationary** time series.
  - Statistical features (mean, variance, etc.) change over the file.
- But most time series approaches assume stationarity.
  - E.g. Fourier decompositions, autoregressive models, moving average models.

Nonstationary Time Series: Which tools can we use?



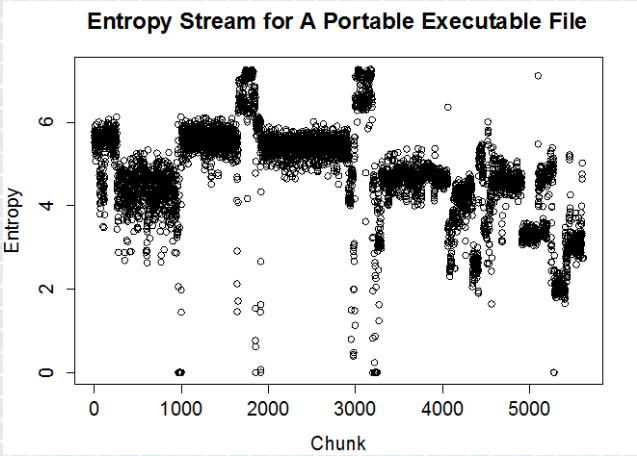
- Solution: We characterize entropy streams as **mean-change point** time series.
  - Structure: local stable means plus noisy variation around that mean
  - Local stable means have an unknown length.

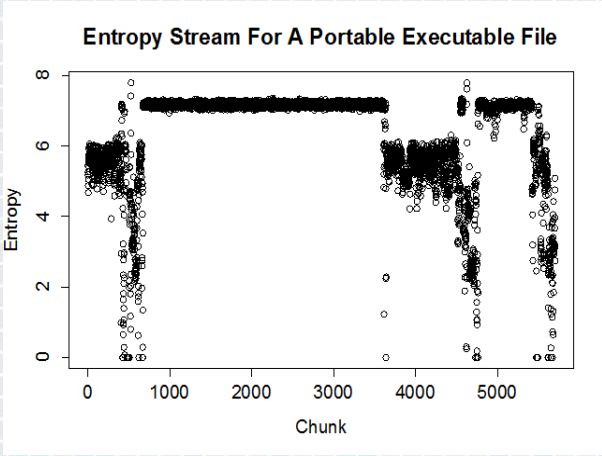
Nonstationary Time Series: Three Modeling Techniques

	Method	Purpose
1	Detrended Fluctuation Analysis	How non-stationary is the entropy signal?
2	Mean Change Point Modeling	Find shifts in entropy
3	Wavelet Energy Decomposition	Which scales show entropic change?

Statistical tools which discover the underlying structure of software entropy

	Method	Purpose
1	Detrended Fluctuation Analysis	How non-stationary is the entropy signal?
2	Mean Change Point Modeling	Find shifts in entropy
3	Wavelet Energy Decomposition	Which scales show entropic change?

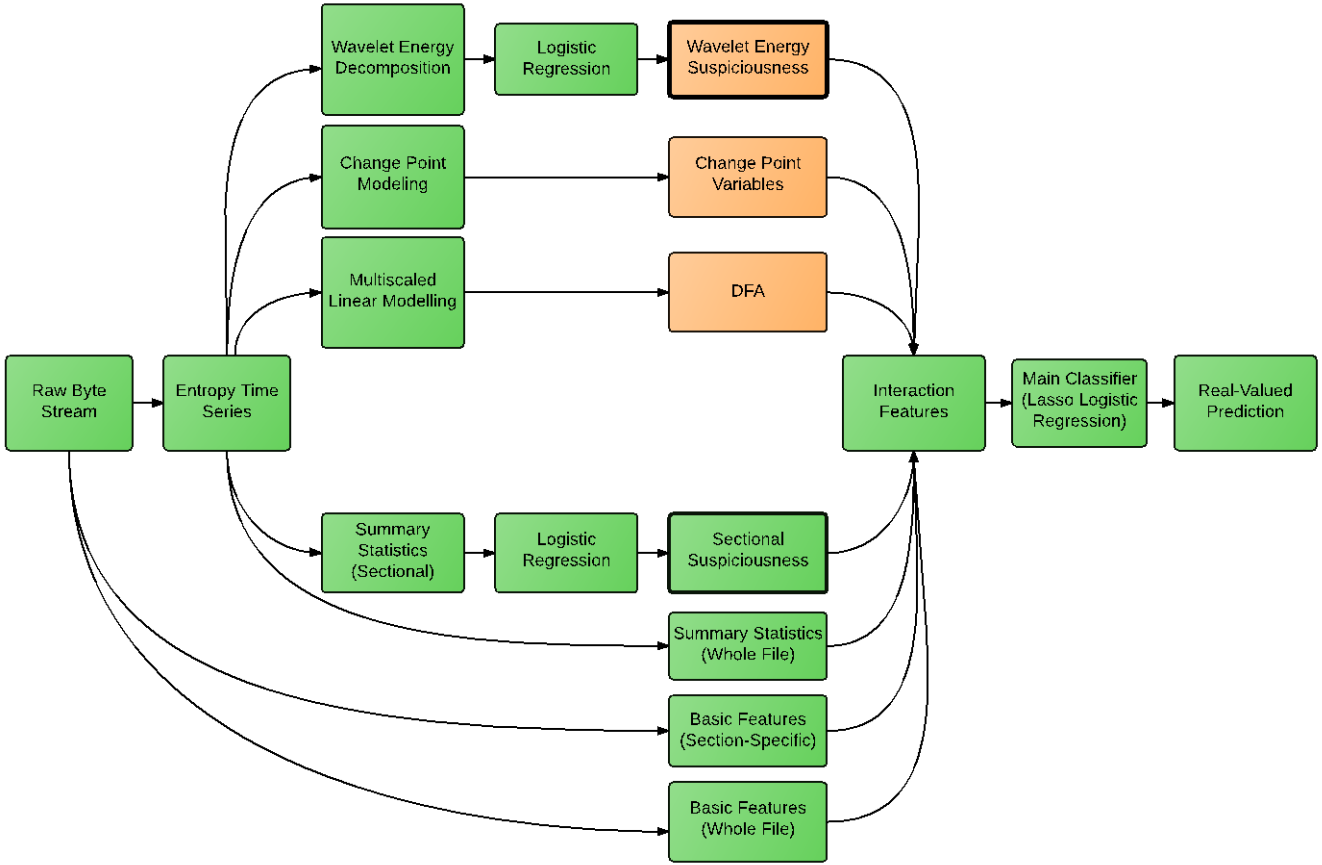




This Work vs. Previous Work

- Moreover, by applying the big-picture modeling perspective of non-stationary time series, we **broaden the notion** of structural entropy
- In particular, we gain
  - At least two new techniques (DFA, MCPM)
  - Development of wavelet techniques (Multi-Resolution Analysis)
  - A multi-tiered workflow (see “Structural Entropy Analysis Flowchart”)

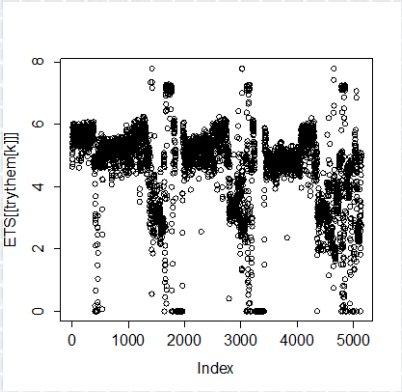
Structural Entropy Analysis Flowchart

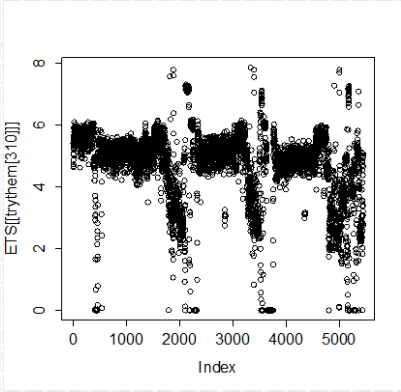


Features are shown in the 5th and 6th columns. Model-based features are shown in orange. Propensity features, which require pre-processing by logistic regression, have thick borders.

This Work vs. Previous Work

- Previous work: structural entropy analysis **computes similarity** between 2 files.
  - This goal comes from a signature-based approach.
- This project: structural entropy analysis **extracts predictive features** from 1 file.
  - This goal comes from a machine-learning approach.





e.g. Baysa, Donabelle, Richard M. Low, and Mark Stamp. "Structural entropy and metamorphic malware." *Journal of Computer Virology and Hacking Techniques*, 9.4 (2013): 179-192.

RSAConference2015

San Francisco | April 20-24 | Moscone Center

First Modeling Technique: Detrended Fluctuation Analysis (DFA)

#RSAC

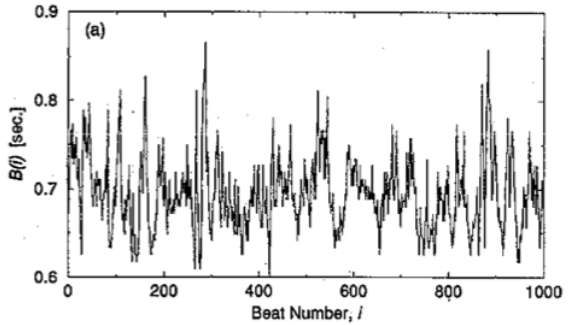
Detrended Fluctuation Analysis (DFA): Intro

- Derived from random fractal theory.
- Generalizes the Hurst statistic for stationary processes, which quantifies the extent to which time series exhibit power-law (i.e. unusually slowly decaying) correlations.
- Fits piecewise linear models to integrated time series at different spatial scales.
- Has been applied to non-stationary signals in bioinformatics (neuronal spiking, DNA sequences, heart rate dynamics, human gate analysis) and theoretically studied in physics.

Chen, Z., Ivanov, P. C., Hu, K., & Stanley, H. E. (2002). Effect of nonstationarities on detrended fluctuation analysis. *Physical Review E*, 65(4), 041107.

Detrended Fluctuation Analysis (DFA): Algorithm

Heart beat interval time series

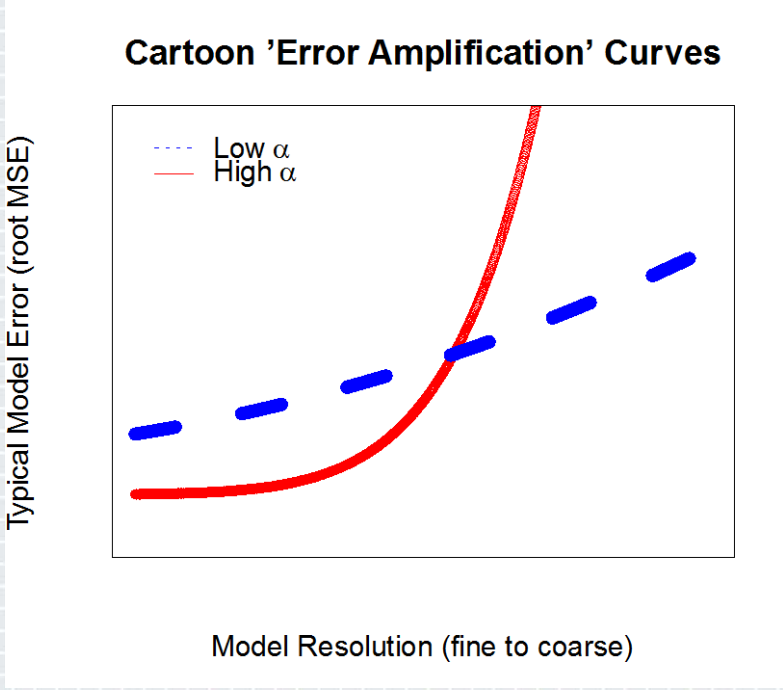


Integrated version

0. Input: Discrete time series. Make it dyadic (so its length is $T = 2^J$ for some integer J.)
1. Integrate: Compute the running sum of the detrended time series.
2. Loop: Choose a scale size or resolution level $j = 2, \ldots, J - 1$.
   - 2.1 Partition: Divide the time series into $T / 2^j$ "boxes", with $n = 2^j$ points per box. Note that the number of boxes decreases (and so the partition is more coarse) as j increases.
   - 2.2 Local OLS Fit: Within each box, get the OLS (Ordinary Least Squares) regression line.
   - 2.3 Detrend: Compute the residuals around the local regression lines.
   - 2.4 Calculate RMSE fluctuation: Define R(n) as the standard residual (RMSE, or Root Mean Squared Error) from the piecewise linear model at resolution level j.
3. Calculate scaling exponent: Fit a linear regression to log $R(n)$ as a function of log n, where $n = 2^j$ describes the resolution level. Get the slope of that line. This is the DFA statistic, α.

Detrended Fluctuation Analysis (DFA): Concept

α: How does error amplify as models simplify?

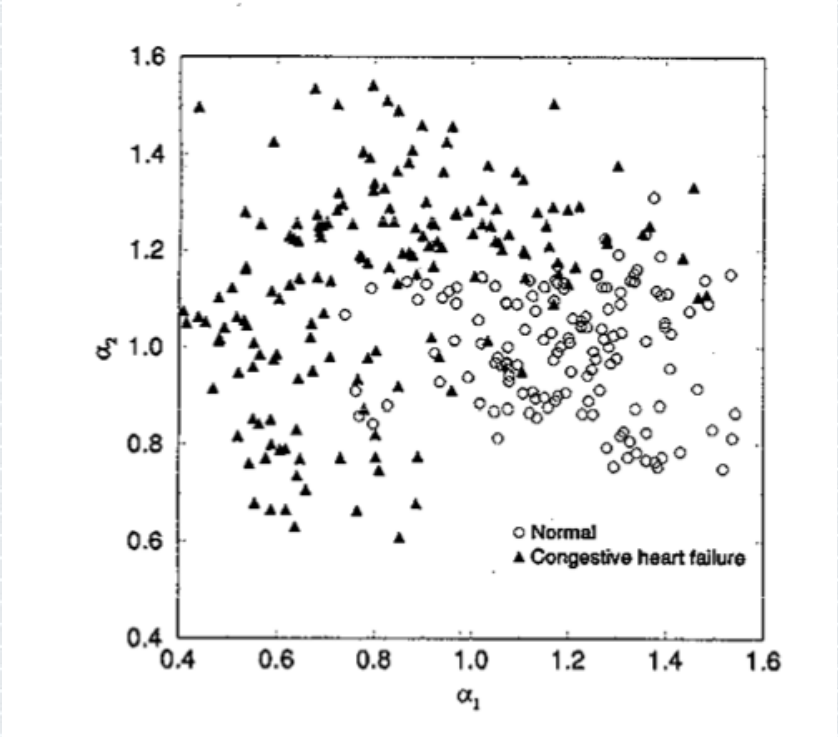


The DFA statistic, α is therefore simply the exponent giving the best fit to the equation

$$R(n) = c\, n^{\alpha} \qquad (1)$$

where n describes the coarseness of the piecewise linear model for the entropy time series, $R(n)$ describes the size of the "typical" residual (or error) for that model, and c is a constant which is ignored.

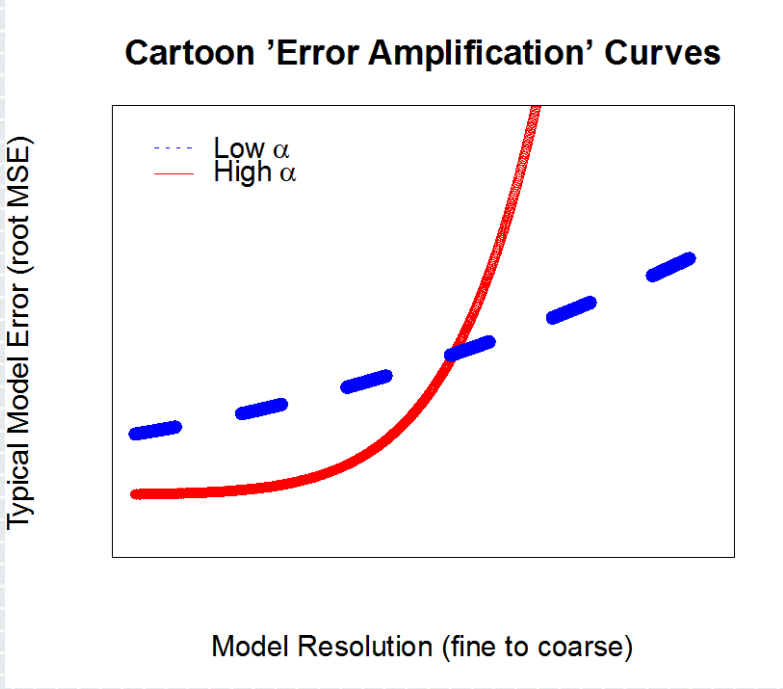
Detrended Fluctuation Analysis (DFA): Utility in Previous Research



Peng, C. K., Havlin, S., Stanley, H. E., & Goldberger, A. L. (1995). Quantification of scaling exponents and crossover phenomena in nonstationary heartbeat time series. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 5(1), 82-87.

Detrended Fluctuation Analysis (DFA): Application to Software Entropy

Question: How would this technique apply to a mean change point time series?



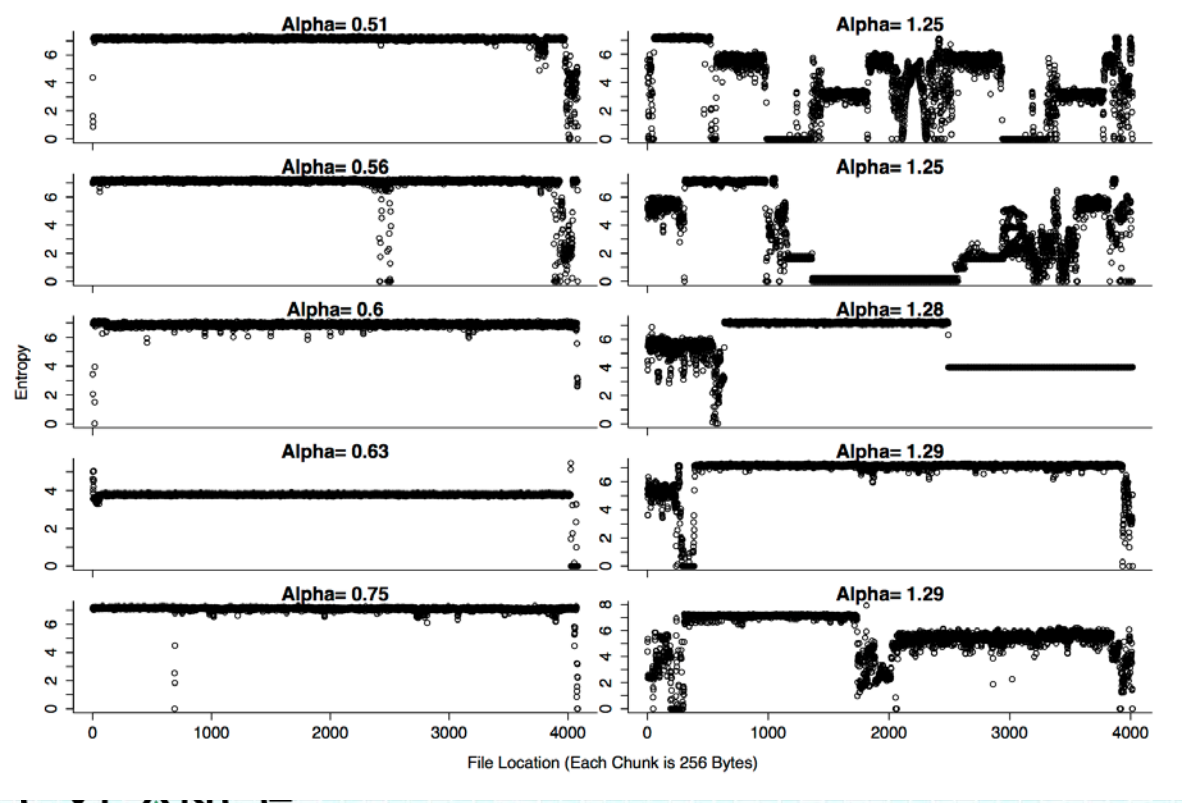
The DFA statistic, α is therefore simply the exponent giving the best fit to the equation

$$R(n) = c\, n^{\alpha} \qquad (1)$$

where n describes the coarseness of the piecewise linear model for the entropy time series, $R(n)$ describes the size of the "typical" residual (or error) for that model, and c is a constant which is ignored.

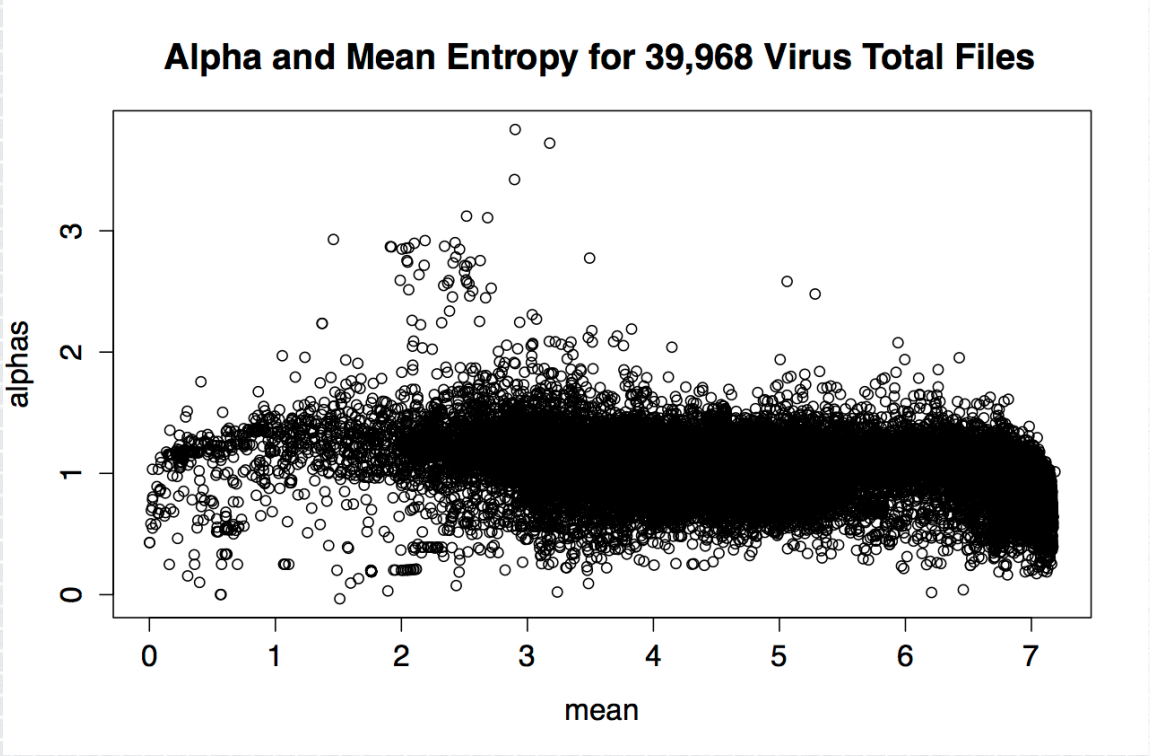
Detrended Fluctuation Analysis (DFA): Pilot Test

The DFA statistic, α describes the level of structure in the entropy time series



- Pilot study: n=93 files that had between 4000 and 5000 entropy blocks
- Shown are files with:
  - 5 lowest α's (left column)
  - 5 highest α's (right column)
- Note that, as we argued, files with larger α tend to be more "structured": they flip between stable local means

Detrended Fluctuation Analysis (DFA): Our Data



DFA is essentially uncorrelated with mean entropy!

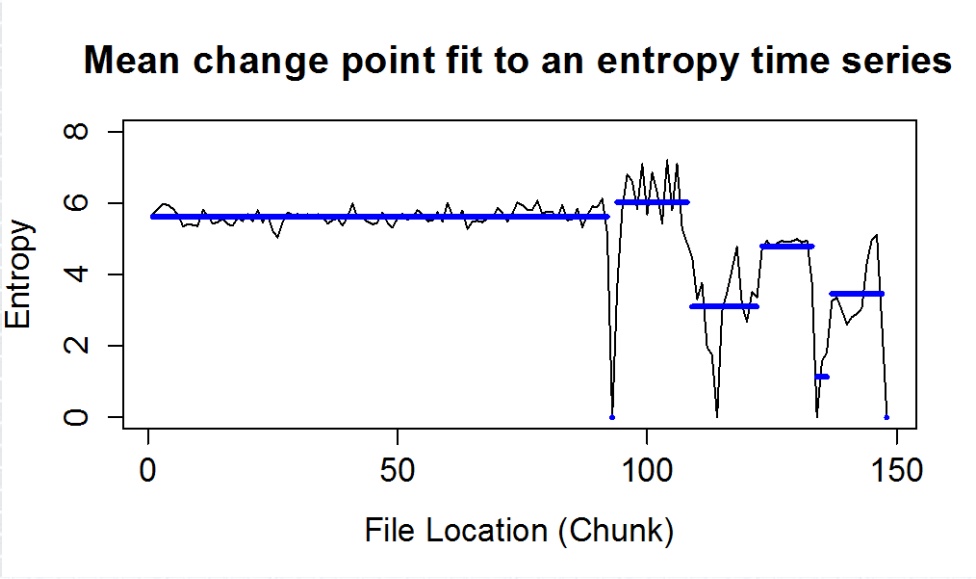
Second Modeling Technique: Mean Changepoint Modeling (MCPM)

Mean Changepoint Modeling: Intro

- A major problem for reverse engineering of software is to identify the entry points for encrypted or compressed sections.
- Mean change point modeling can automatically determine the inherent sections of a file according to entropy-based **code regimes** (native code, strings, padding, encryption, compression)
- These code regimes needn't align with author-designated section labels: text, rsrc, reloc, data, etc.

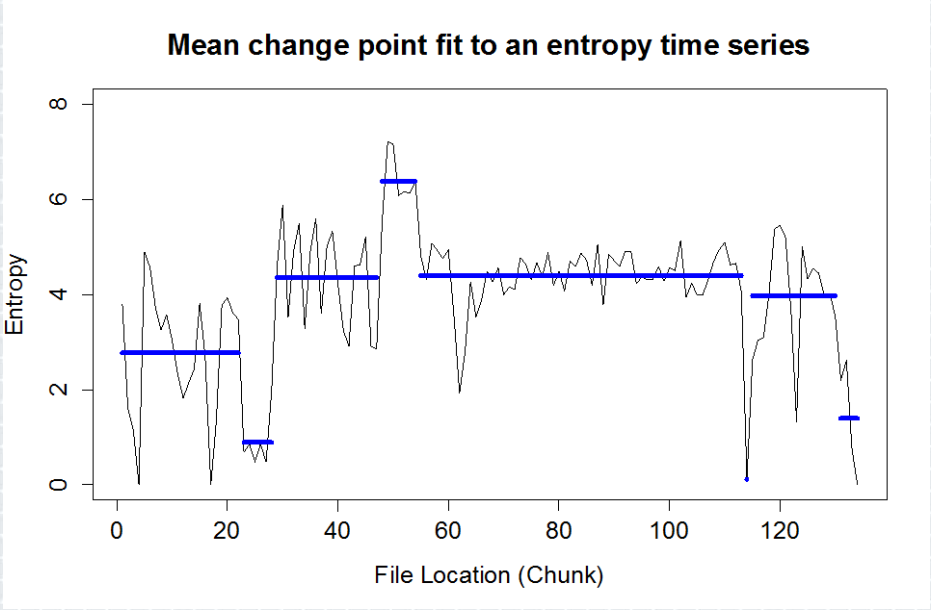
Mean Changepoint Modeling: Concept

- Input: an ordered sequence of data, $Y_{1:T} = (Y_1, \ldots, Y_T)$.
- Output: The number of change points, m, along with their positions $\tau_{1:m} = (\tau_1, \ldots, \tau_m)$.



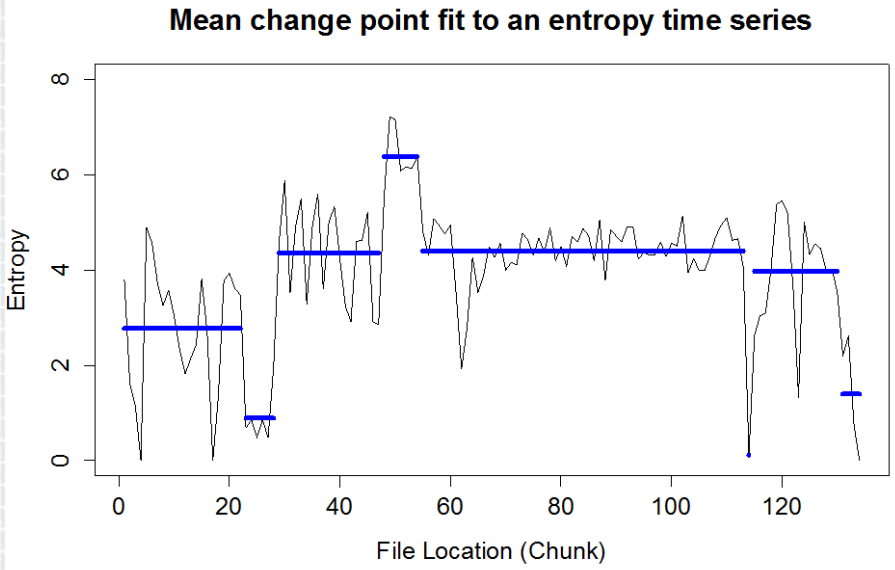
The m change points split the data into $m+1$ segments, with the ith segment containing data subset $y_{(\tau_{i-1}+1):\tau_i}$.

Mean Changepoint Modeling: Example Fit



	1	2	3	4	5	6	7	8
local Mean	2.77	0.89	4.36	6.39	4.4	0.12	3.98	1.4
length	22	6	19	7	59	1	16	4

Mean Changepoint Modeling: Features



	1	2	3	4	5	6	7	8
local Mean	2.77	0.89	4.36	6.39	4.4	0.12	3.98	1.4
length	22	6	19	7	59	1	16	4

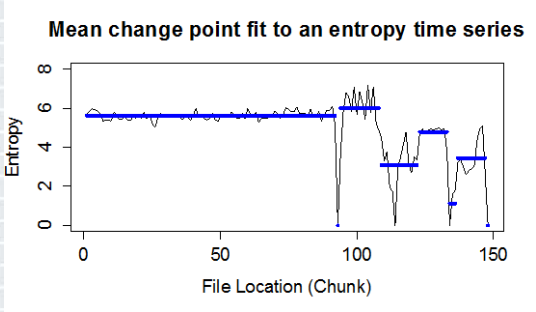
maxJumpSize	3.864
minJumpSize	–4.287
maxLongestRunPct	0.44
meanRunLength	0.077
stableProp	0.44
entropyRuns	1.882
meanLongestRun	4.403
realFileMean	4.403
maxResid	2.645

These features characterize the size and prevalence of jumps, the value of landmark mean entropies, and the fit of the model.

Mean Changepoint Modeling: Algorithm

The mean change point model solves

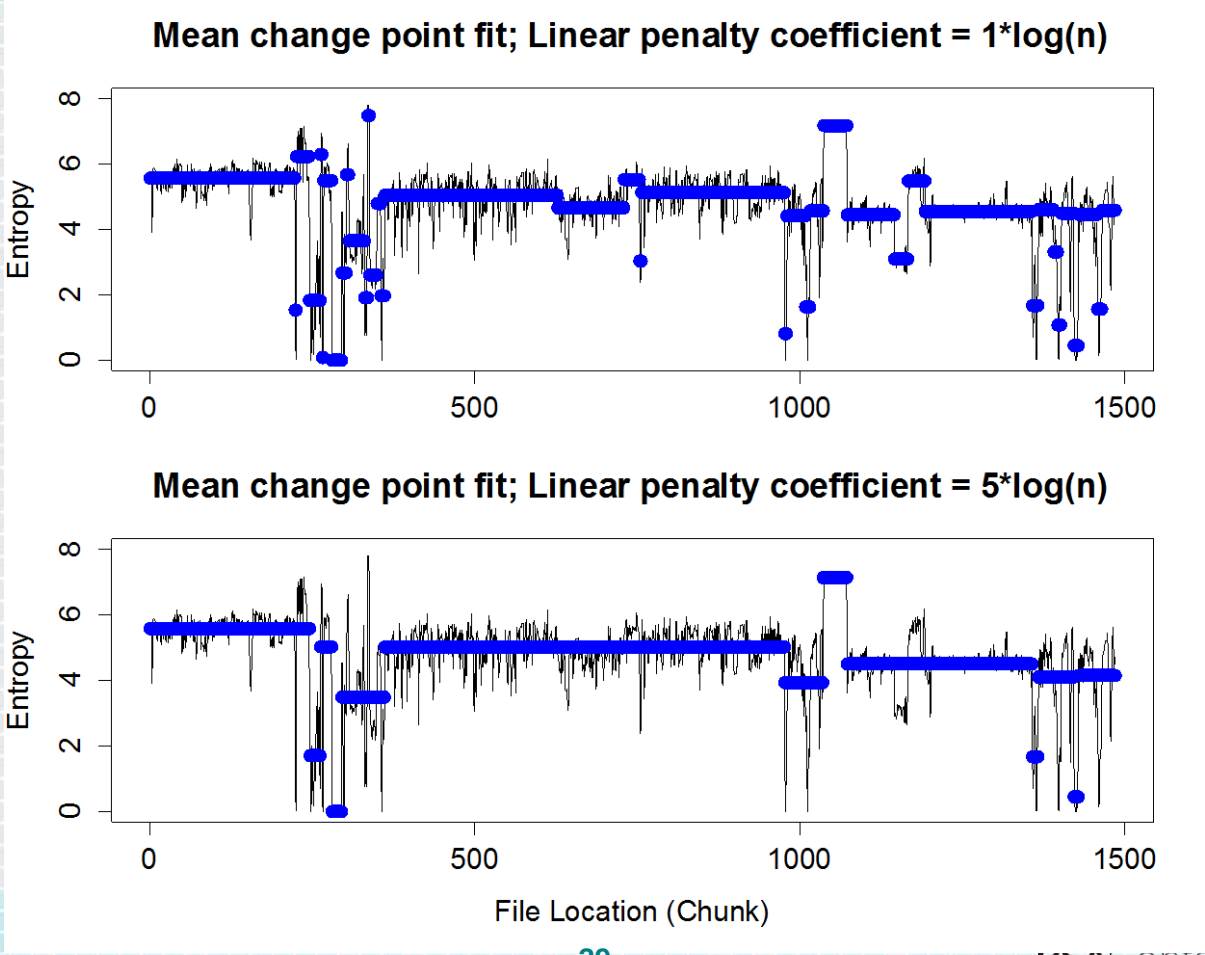
$$\underset{\tau_1,\ldots,\tau_m;m}{\arg\min} \sum_{i=1}^{m+1} \underbrace{[\mathcal{E}(y_{\tau_{i-1}+1:\tau_i})]}_{\text{fit error}} + \underbrace{\mathcal{P}(m)}_{\text{penalty}}$$



- $\mathcal{E}$ is an error function (twice the negative log likelihood, where data is assumed to be normally distributed around the local mean with some unknown variance σ^2)
- $\mathcal{P}(m) = p\ log(n)m$ is the Schwartz Information Criterion (SIC), a penalty to guard against over fitting
  - n = number of data points in entropy time series
  - m = number of change points
  - p = number of additional parameters introduced by adding a single change point (here p=1)

We minimize this function through the "PELT" (Pruned Exact Linear Time) algorithm, which is $\mathcal{O}(n)$.

Mean Changepoint Modeling: Limitations



Third Modeling Technique: Wavelet Decomposition

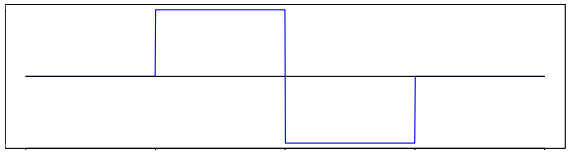
Wavelet Transforms: Intro

- The Wavelet Transform is a generalization of the Fourier Transform.
- The Wavelet Transform we have chosen is known as a Haar Wavelet Transform.
  - We can project the original entropy signal onto a family of square waves.
- By re-representing the signal in terms of changes at various scales, the Haar Wavelet Transform performs a **multi-resolution analysis**, revealing how entropic changes in the computer program are distributed across small, intermediate, and large scales.

Haar Wavelet Decompositions

The so called "mother function", $\psi(t)$ is defined by:

$$\psi(t) = \begin{cases} 1, & t \in [0, 1/2) \\ -1, & t \in [1/2, 1) \\ 0, & \text{otherwise} \end{cases}$$



A collection of dyadically scaled and translated wavelet functions $\psi_{j,k}(t)$ for resolution j in location k is then formed:

$$\psi_{j,k}(t) = 2^{j/2}\psi(2^j t - k)$$

- The resolution levels, $j \in \{0, \ldots, J-1\}$, are ordered coarse- to fine-grained
- Location is in $k \in \{0, \ldots, K = 2^j - 1\}$

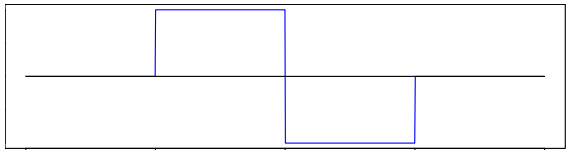
Haar Wavelet Decompositions

The "wavelet coefficients" $\{d_{j,k}\}$ are given by

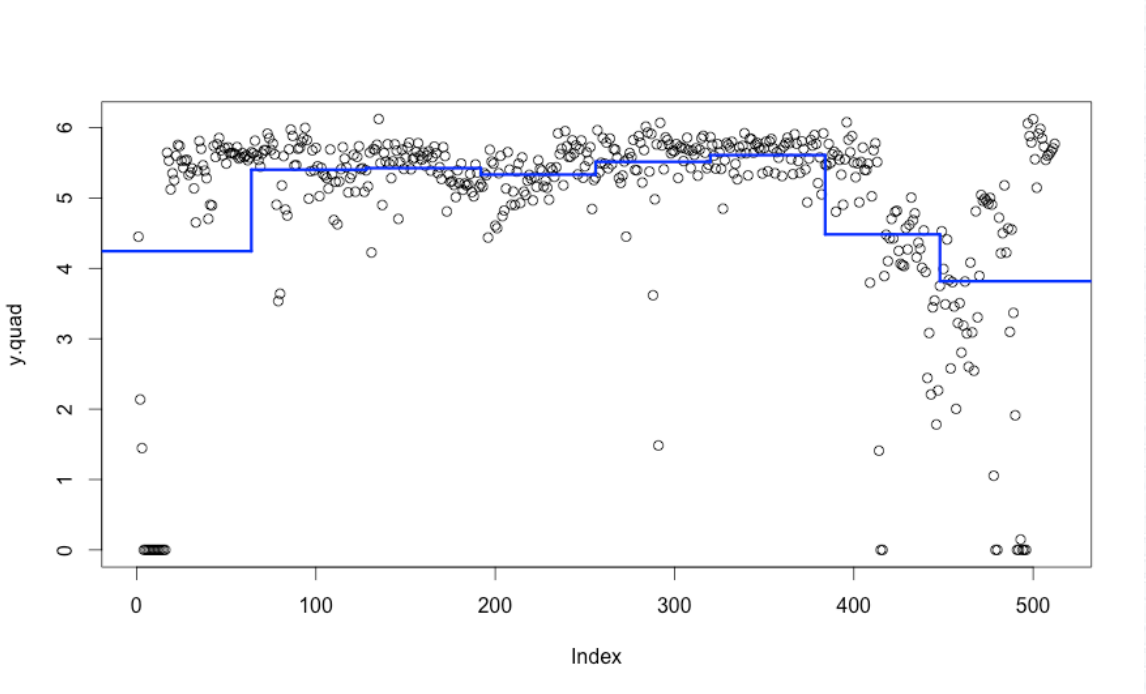
$$d_{j,k} = < x, \psi_{j,k} > = \sum_{t=1}^{T} x(t)\psi_{j,k}(t)$$

The functional approximations are then obtained:

$$x_{j+1}(t) = x_j(t) + \sum_{k=0}^{2^j - 1} d_{j,k}\psi_{j,k}(t)$$

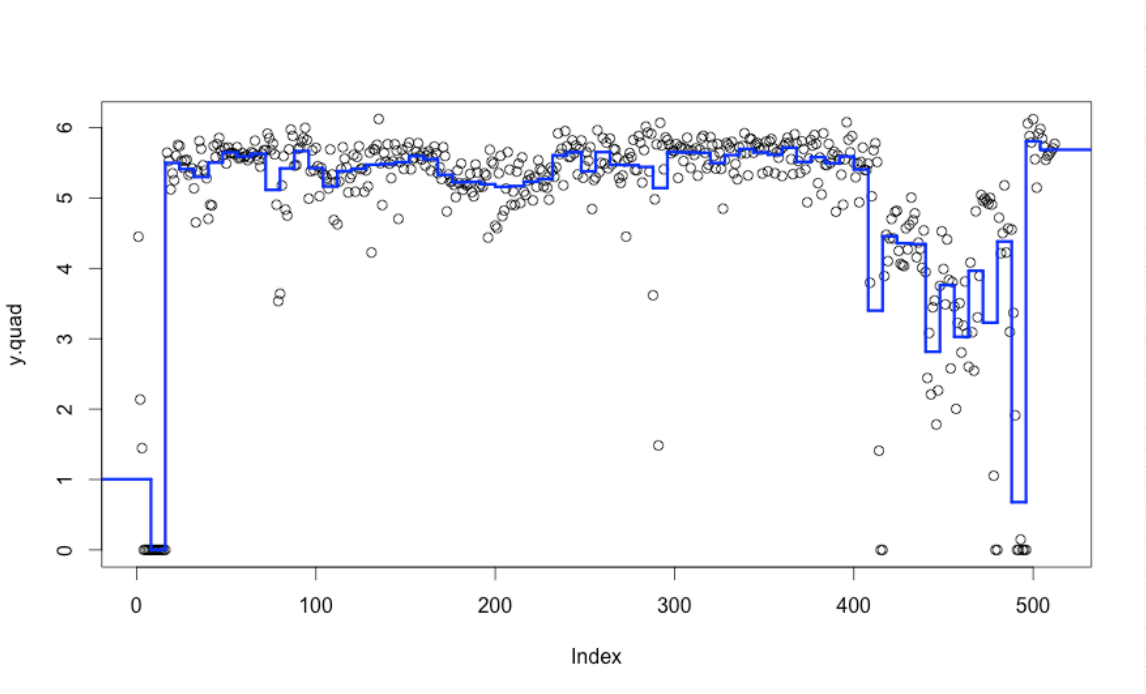


Wavelet-Based Functional Approximations



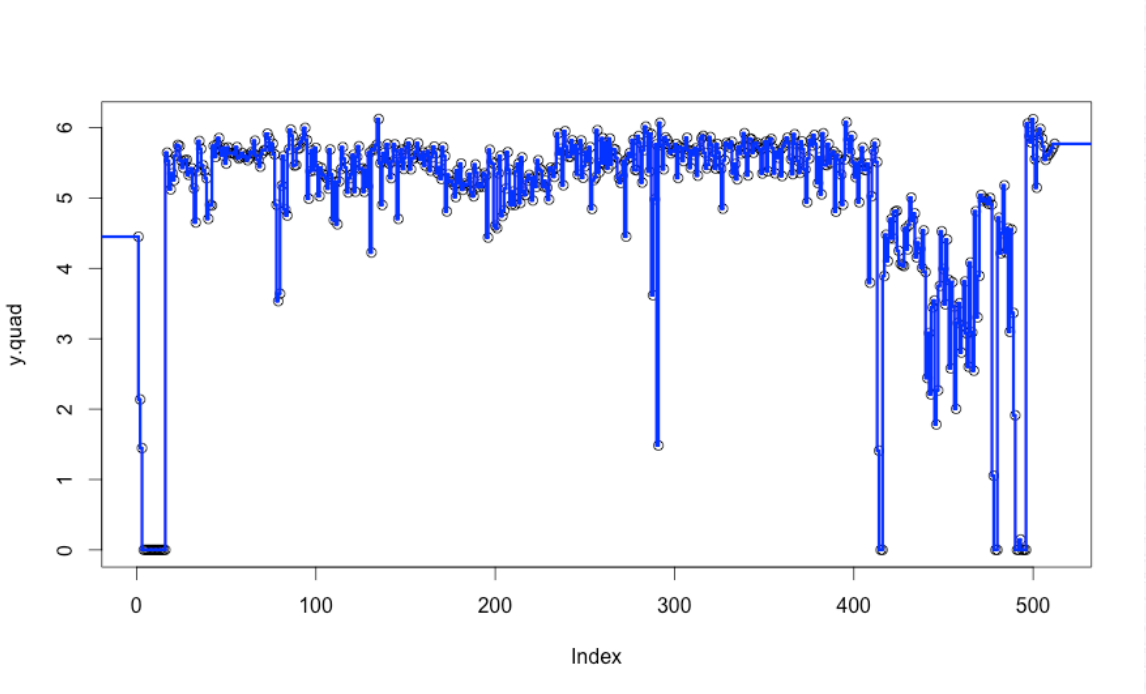
Entropy Time Series projected onto Haar father wavelet space with 2^3 segments

Wavelet-Based Functional Approximations



Entropy Time Series projected onto Haar father wavelet space with 2^6 segments

Wavelet-Based Functional Approximations



Entropy Time Series projected onto Haar father wavelet space with 2^J segments

Wavelet Energy Spectrum: Formula

The Wavelet Energy Spectrum summarizes the "detail" or "variation" available at various resolution levels. The energy spectrum is computed as a function of the mother wavelet coefficients, d_{jk}. In particular, the "energy", E_j, of the time series at the jth resolution level is defined by:

$$E_j = \sum_{k=1}^{2^j - 1} (d_{jk})^2$$

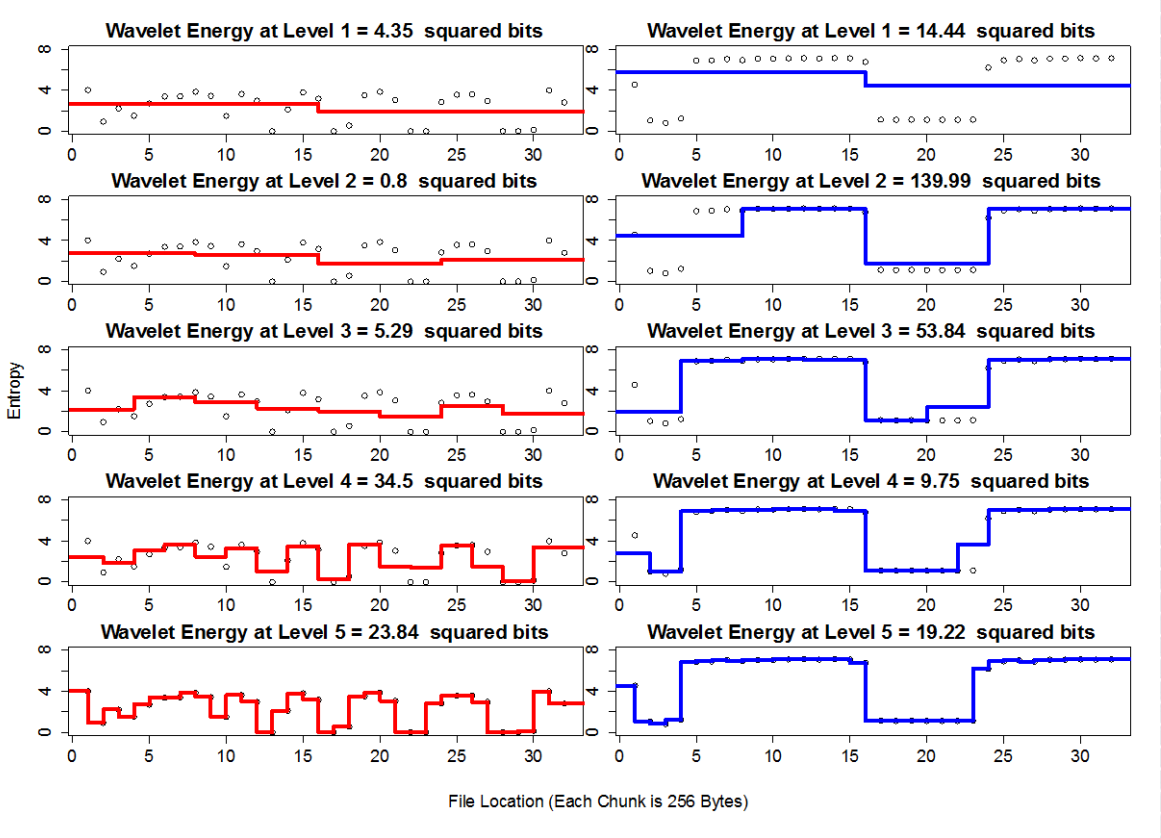
Wavelet Energy Spectrum: Summary

- We perform a **multi-resolution analysis**, revealing how entropic changes in the computer program are distributed across small, intermediate, and large scales.
- The **wavelet coefficients** summarize the amount of "detail" or "change" exhibited within a time series at various locations over various levels of resolution.
- Using these wavelet coefficients, we summarize the overall amount of entropic detail in a time series at various levels of resolution. The total amount of detail at a particular level of resolution is known as its **energy.**
- The distribution of energy across various levels of resolution is known as an **energy spectrum.**

Wavelet Energy Spectrum as Features

- There is strong precedent, especially in biology, for using the energy spectrum of signals as features within automatic classification systems.
- For instance, using wavelet coefficients, researchers have been able to
  - Automatically classify lung sounds into categories (crackles, wheezes, striders, squawks, etc.)
  - Determine whether brain EEG scans originated from healthy patients, patients with epilepsy, or patients who were in the middle of having a seizure.
  - Determine whether EMG signals collected from the bicep originated from patients who were healthy, suffering from myopathy, or suffering from neurogenic disease

Pilot Study on n=1,599 files of size J=5



These are (Haar) wavelet-based functional approximations to a representative good and bad file.

Pilot Study on n=1,599 files of size J=5

Resolution			Statistical Model For File Size J = 5		
Level	# Bins	Bin Size	Value of β_j	$P-value$	Malware Sensitivity
1	2	16	0.448	*****	+56.5%
2	4	8	0.174	*	+19.0%
3	8	4	0.847	*****	+133.2%
4	16	2	-0.106	n.s.	-10.0%
5	32	1	-0.240	**	-21.4%

- "Value of β_j": the estimated beta weight in a logistic regression, based on a corpus of n=1,599 files of size J=5, fitting file maliciousness to the (normalized) wavelet energy values at five levels of resolution
- "P-value" codes: $* = p < .05$, $** = p < .01$, $*** = p < .001$, $**** = p < .0001$, $***** = p < .00001$.
- "Malware Sensitivity": the estimated change in the odds that a file is malware associated with an increase of one standard deviation in the corresponding feature. It is calculated by $(e^{\beta} - 1) \times 100\%$.

A problem in application to malware analysis

- The biological studies above were all **controlled observational situations** which produced time series samples (e.g. lung sounds, brain EEGs) of fixed length.
- Thus, these samples all produce the same number of wavelet energy features, J
- Variation in these J features were then immediately associated with a classification variable (categories of lung sounds, presence/absence of epilepsy, etc.) by setting the input layer of the neural network to have J activation notes.
- Yet executable files **out in the "wild"** have different lengths, and so J will vary from sample to sample.

Wavelet Energy Spectrum Suspiciousness: Definition

- To solve this problem, we create an algorithm which converts the wavelet energy spectrum into a malware "propensity score" or "suspiciousness score" within [0,1] describing the "propensity" of that executable file to be malware based on the wavelet energy spectrum alone.
- We call the resulting feature **Wavelet Energy Spectrum Suspiciousness**
- This feature can be compared from sample to sample. the input layer of the neural network to have J activation notes.

Data

Data

- Data is a corpus of Windows Portable Executable (PE) files.
- We formed a balanced data set of 39,968 files; of these, 19,991 (50.01%) were malware, whereas 19,997 (49.98%) were clean computer programs.
- The "malware" category contained different types of malicious software (e.g., computer viruses, Trojan horses and spyware -- but not adware).

Data: Training and test Set

- Before feature extraction, 31,676 (80%) of the samples were randomly selected to belong to the "training set," and the other 7,916 (20%) of the samples were classified in the "test set."
- The training set was used to
  1. train the feature weights in the final classifier
  2. determine the sparsity of the final classifier
  3. determine weights for constructing the "propensity features"

Method

Feature Overview

I. WHOLE FILE FEATURES (20)

A. BASIC (2)	B. SUMMARY ENTROPY (7)	C. CHANGE POINT * (9)	D. WAVELETS *+ (1)	E. DFA * (1)
length	mean	maxJumpSize	WESSS	α
length^2	sd	minJumpSize		
	STN	maxResidual		
	max	meanLongestRun		
	sneakyHigh	maxLongestRunPct		
	lotsHigh	meanRunLength		
	pct0	entropyRuns		
		realFileMean		
		stableProp		

II. SECTION SPECIFIC FEATURES (12)

A. BASIC (6)	B. SECTION SUSPICIOUSNESS + (6)
dataExists	dataSuspiciousness
rsrcExists	rsrcSuspiciousness
relocExists	relocSuspiciousness
textExists	textSuspiciousness
headerExists	headerSuspiciousness
certificateExists	certificateSuspiciousness

III. INTERACTION FEATURES (488)

All structural entropy features used for classification.

KEY: * = Model-based features, + = Propensity Features

Model Building: Lasso Logistic Regression

- We model the binary classification variable (malware or not) as a function of the 520 features related to structural entropy described above.
- We apply **lasso logistic regression**, a principled statistical method for model fitting while simultaneously performing feature selection.
- The logistic lasso produces an entire set of models, indexed by the sparsity parameter, λ. As the sparsity parameter increases, the size of the model decreases (by discarding features).
- The value of the sparsity parameter can be optimized. Thus, the lasso discovers the optimal proportion of features useful for classification

Model Building: Lasso Logistic Regression

In particular, the vector of feature weights, β, which solve the lasso logistic regression function, is given by :

$$\min_{\beta} \left[f(\beta) = \sum_{i=1}^{N} log\left(1 + e^{-y_i \beta^T x_i}\right) + \lambda ||\beta||_1 \right]$$

where λ is the "lasso parameter" or "regularization parameter" that determines the sparsity of the solution.

Model Building: Lasso Logistic Regression

In particular, the vector of feature weights, β, which solve the lasso logistic regression function, is given by :

$$\min_{\beta} \left[\underbrace{f(\beta) = \sum_{i=1}^{N} log\left(1 + e^{-y_i \beta^T x_i}\right)}_{\text{Error in Fit}} + \underbrace{\lambda ||\beta||_1}_{\text{Penalty}} \right]$$

where λ is the "lasso parameter" or "regularization parameter" that determines the sparsity of the solution.

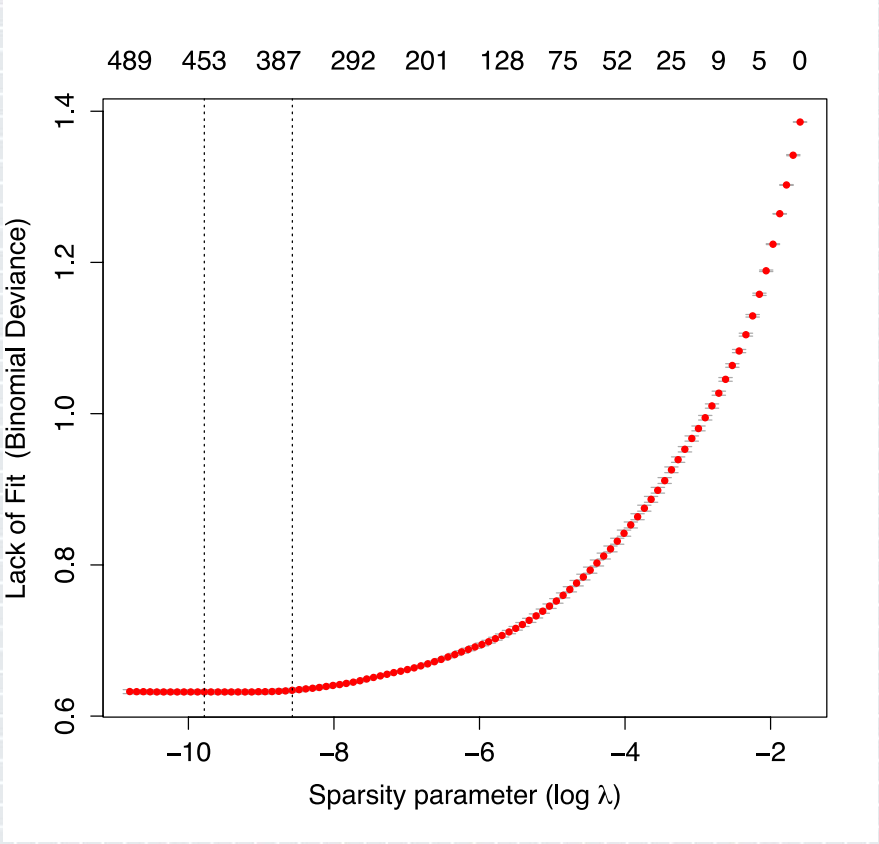
The optimal value of λ is chosen by 10-fold cross-validation.

Results

Optimal sparsity of model

Our feature construction method was rather liberal. For example, our original dataset included 488 interaction terms (bivariate products of simpler features), and these had no a priori justifications. Could we have used far fewer than 520 features in our set?

Optimal sparsity of model: Keep 387 Features



Finding useful features in the dataset. The graph depicts the relationship between model size (governed by the sparsity parameter, and plotted on the x-axis) and the fit of model (assessed by the deviance measure, and plotted on the y-axis) to validation data.

The number of features retained by model is completely determined by the sparsity parameter and is shown in the top row above the plot.

Overall model performance

Predicted	Actual Clean	Actual Malware
Clean	3413	443
Malware	569	3491
Total Accuracy	87.21%	

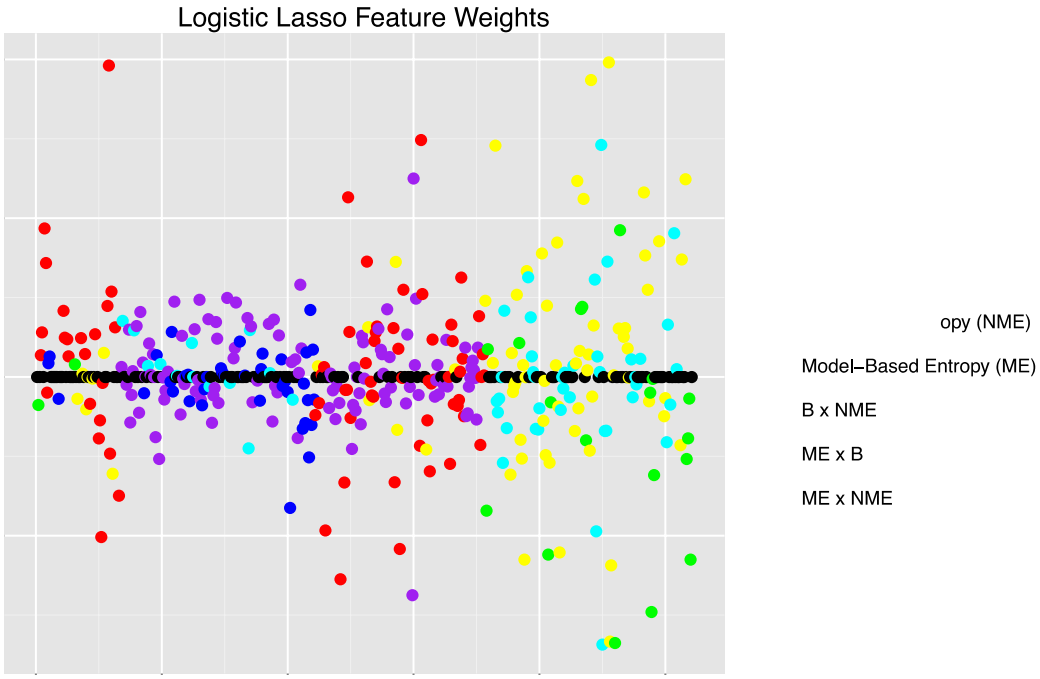
Test set performance of structural entropy classifier on a set of 7,916 newly seen files.

Contribution of model-based features

Feature Subsets	Test Set Performance
Alphas Only	53.09%
Mean Change Point Only	64.16%
Wavelet Features Only	67.42%
All 3 (Main Effects Only)	71.70%
All 3 (with Interactions)	73.44%
Full Model	87.21%

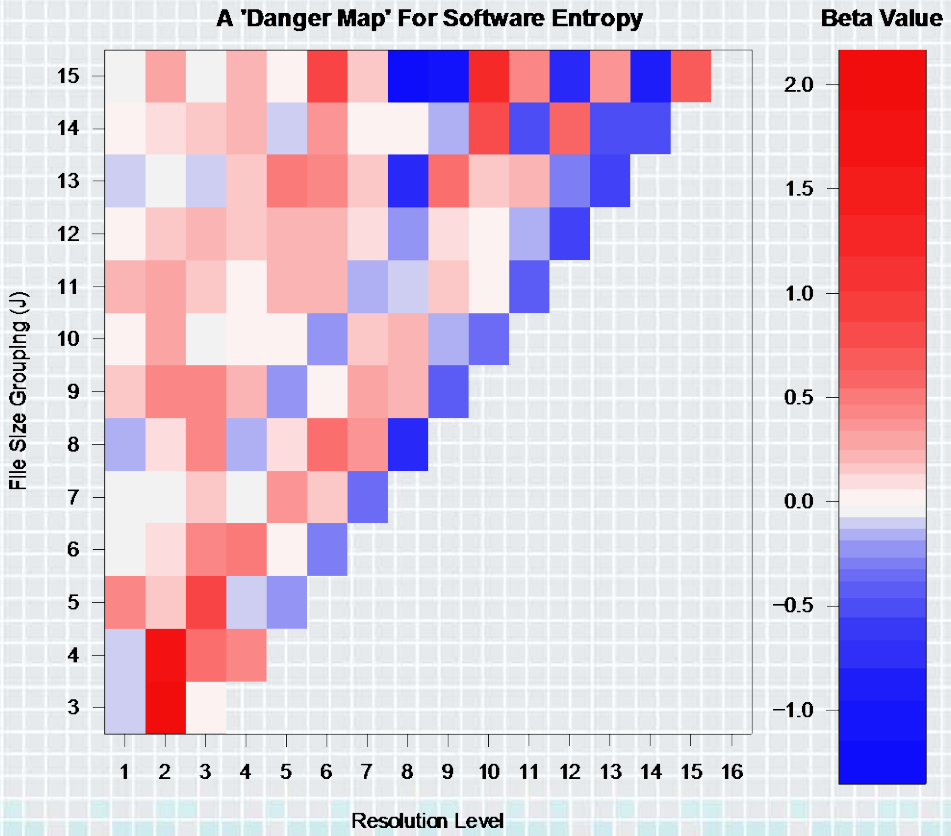
Test set performance (n=7,916) of classifiers using various subsets of the model-based entropy features

Contribution of model-based features



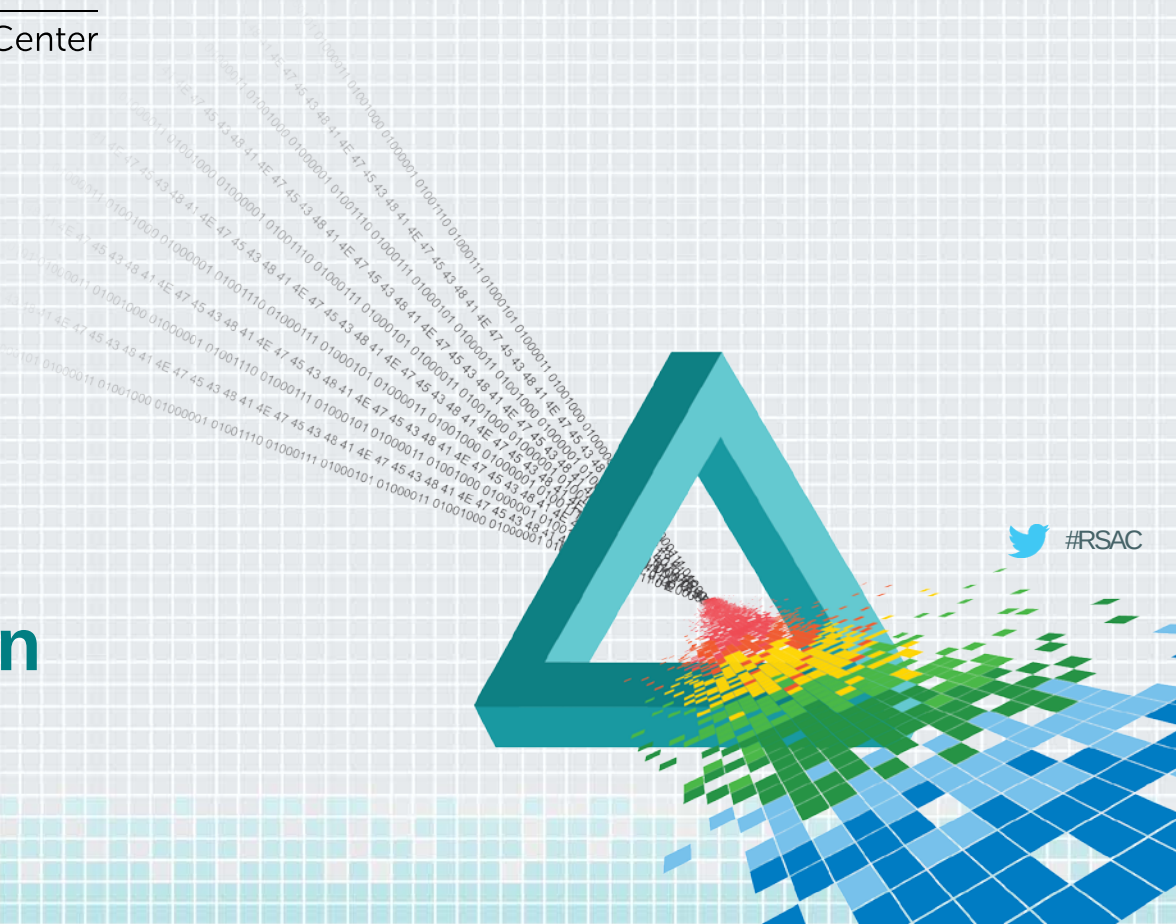
Feature Importance Plot: Blue, purple, and cyan are prominent throughout the display, which reflects that the model-based features are quite substantially involved in the overall predictive performance of the automated malware classifier.

Wavelet Energy Patterns: A “Danger Map”



The plot shows logistic regression beta coefficients for determining the probability that a portable executable file is malware based upon the magnitude of file's entropic energy at various levels of resolution within the code.

- Positive betas mean that higher "entropic energy" at that resolution level is associated with a greater probability of being malware.
- Negative betas (mean that higher "entropic energy" at that resolution level is associated with a lower probability of being malware.



Discussion

Take Home Points

- Machine learning automatic malware classification systems could benefit from incorporating "structural entropy" features.
- We extracted structural entropy features via three primary techniques: detrended fluctuation analysis, mean change point modeling, and wavelet decompositions.
- These features, especially considered in tandem, allow for the structural entropy classifier to construct a fairly complex relationship between machine code and a judgment of "entropic suspiciousness."
- This relationship would be much harder for an adversary to control than, say, mean entropy levels.
- Moreover, the strong predictive performance of these structural entropy features suggest that they may be a powerful contributor to a larger machine learning system for automatically classifying malware.

Questions and Answers

Contacts:

- Mike Wojnowicz mwojnowicz@cylance.com
- Glenn Chisholm gchisholm@cylance.com