SANDIA REPORT
SAND2020-10617
Printed September 2020

Sandia National Laboratories

Time Series Dimension Reduction for Surrogate Models of Port Scanning Cyber Emulations

Erin C.S. Acquesta, Laura P. Swiler, and Ali Pinar

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico
87185 and Livermore,
California 94550

Issued by Sandia National Laboratories, operated for the United States Department of Energy by National Technology & Engineering Solutions of Sandia, LLC.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@osti.gov
Online ordering: http://www.osti.gov/scitech

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5301 Shawnee Rd
Alexandria, VA 22312

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.gov
Online order: https://classic.ntis.gov/help/order-methods/

ABSTRACT

Surrogate model development is a key resource in the scientific modeling community for providing computational expedience when simulating complex systems without loss of great fidelity. The initial step to development of a surrogate model is identification of the primary governing components of the system. Principal component analysis (PCA) is a widely used data science technique that provides inspection of such driving factors, when the objective for modeling is to capture the greatest sources of variance inherent to a dataset. Although an efficient linear dimension reduction tool, PCA makes the fundamental assumption that the data is continuous and normally distributed. Thus, it provides ideal performance when these conditions are met.

In the case for which cyber emulations provide realizations of a port scanning scenario, the data to be modeled follows a discrete time series function comprised of monotonically increasing piece-wise constant steps. The sources of variance are related to the timing and magnitude of these steps. Therefore, we consider using XPCA, an extension to PCA for continuous and discrete random variates. This report provides the documentation of the trade-offs between the PCA and XPCA linear dimension reduction algorithms, for the intended purpose to identify key components of greatest variance in our time series data. These components will ultimately provide the basis for future surrogate models of port scanning cyber emulations.

ACKNOWLEDGEMENTS

This work was funded by the Laboratory Directed Research and Development program at Sandia National Laboratories. The authors gratefully acknowledge the support of the SECURE Grand Challenge project. We also acknowledge the help of our colleagues, including Thomas Tarman, Eric Vugrin, Christian Reedy, and Gerardo Cruz.

CONTENTS

1. Introduction 7
2. Dimension Reduction 8
   2.1. PCA: Principal Component Analysis 8
   2.2. XPCA 11
   2.3. Time Series Dimension Reduction 11
3. Cyber Emulations of Port Scanning 11
   3.1. Experiments 12
   3.2. Evaluation of Dimension Reduction 14
4. Conclusion 23
Appendix A. Reconstruction Visualizations 24

This page left blank

1 Introduction

In monitoring the behavior of physical or emulated computer experiments, one is often interested in the number of certain events over time. For example, quantities of interest may include the number of ports scanned, the number of times a website is pinged, the number of alarms or alerts, the number of packets sent per second, etc. While some quantities of interest may be real numbers, the ones listed and many others are discrete with binary or integer values.

In the progress of computer experiments, one records these quantities at some frequency (e.g., every second). This results in long vectors containing time series data characterized by discrete values such as the number of packets sent per second over a ten thousand second time interval. Computer experiments often exhibit an inherent stochastic behavior due to: randomness in timings of initializations, small changes in orderings of system calls, etc. Thus, if an experiment is repeated with all controllable conditions exactly the same, one run may result in a time series vector that differs significantly from another time series vector generated from an independent run with the same experimental configuration.

Our interest in this work is to understand how much of the inherent randomness observed in time series vectors of discrete quantities from computer experiments, or emulations, can be explained by a few underlying components. For example, if we run a cyber emulation 100 times and take the time series value at time, say, $t = 2374$ seconds, we may see realizations of this value at this particular time slice which differ significantly across the 100 replicates. Our goal is to determine if there is an underlying representation which can capture this variability and explain it with a relatively few vectors, called principal components. This process of explaining a large data set with a small number of components is common practice in the data sciences and is referred to as dimension reduction.

The most versatile and widely used method for dimension reduction is known as Principal Component Analysis (PCA). The statistical theory for PCA has been established for decades [[12] [5] [9]] but assumes that the vectors of data are all continuous valued. Recently, researchers at Sandia have developed a version of PCA called XPCA that can inherently handle discrete variables [1]. We present a brief overview of the theory for PCA and XPCA in Section 2. Then, we apply PCA and XPCA to several datasets that were generated as part of the SECURE Grand Challenge LDRD project. These datasets involve 1000 replicates generated by cyber emulations of various scanning scenarios, where the quantities of interest are realized as the number of closed, open, and filtered ports found. These quantities are all discrete values. A comparison of results of applying PCA versus XPCA are presented in Section 3. We provide a summary discussion and analysis of the differences in these two dimension reduction methods when applied

to cyber emulation experiment data in Section 4.

2 Dimension Reduction

The field of study related to dimension reduction is quite extensive. The key to determining when one method is more appropriate than an alternative method relies on both an understanding of the data and the intended purpose for which the latent space (i.e. lower-dimensional transformation) will be used. In our case, the data is a time series of discrete values that are monotonically increasing, replicating the behavior of a piece-wise constant step function. For the use case of a latent space representation, we need to first summarize the cyber emulation under consideration. Although we are focused on emulating an attacker scanning for vulnerable ports, this is only one part of a larger attack chain in which the defender tries to thwart the attacker with various defensive strategies. Thus, the target exemplar that is the focus of our report, is only a small component of the overall modeling challenge that can be quite computationally expensive to implement. Finding efficiencies here, without loss of fidelity, will buy us computational gains as we continue to layer in more complexities. Given the redundancy in a piece-wise constant step function, we know that we can reduce the time series to the points for which a step occurs. The timing of these switches and the size of a jump at each step is subject to the random nature of the order for scanning the ports, probability of packets dropping, and additional external emulation conditions. The details for these sources of variance will be provided in Section 3. Here we want to emphasize the goal of modeling the components of the time series that contribute the greatest variation in our realizations. Identifying a linear dimension reduction method that captures the greatest variance in the experiments and inherently manages discrete realizations is ideal.

The discreteness of data led us to exploring the use of XPCA. As an extension of PCA, we thought it would be best to baseline our understanding and use of XPCA in comparison to the widely used traditional PCA method. In this section, we provide the fundamental principles of PCA and the extensions that XPCA provides for discrete realizations, with a brief summary for how these methods are applied to time series data.

2.1 PCA: Principal Component Analysis

Principal Component Analysis is a dimension reduction technique that has been commonly used for exploratory data analysis for decades. The idea is to reduce the dimensionality of the data while retaining as much of its variability as possible. It is credited to Karl Pearson [2] and Harold Hotelling [4]. Two excellent books on the topic are *Principal Component Analysis* [5] and *Functional Data Analysis* [9]. The review article by Jolliffe and Cadima [6]

provides an overview of PCA summarizing the method and some of its recent applications in data science.

The idea in PCA is to construct a linear representation of variable values to highlight the variance present in the data. The exposition used in this section follows that of Ramsay and Silverman [9]. We assume the data are in a matrix $\mathbf{X}$ with N rows corresponding to N sample observations and L columns corresponding to L variables of interest. To find the first principal component, define f_i as a linear combination of variable values:

$$f_i := \sum_{j=1}^{L} \beta_j x_{i,j} = X_{(i,:)}\beta, \text{ for } i = 1 \dots N. \tag{1}$$

The goal is to find the set of $L \times 1$ coefficients β_1 such that the variance of f_i across all rows is maximized.

The optimal β_1 when applied to all observations will result in a vector Y_1 of dimension $N \times 1$:

$$Y_1 = \mathbf{X}\beta_1. \tag{2}$$

This vector identifies the strongest and most important mode of variation. [9] The process is repeated with subsequent steps, to determine the sets of coefficients β_2, β_3, ... that define the second, third, etc. most important modes of variation. The coefficient vectors β are called *loadings*, and the elements of the Y vector are called *factor scores*. The *principal components* are given by $\mathbf{X}\beta_1$, $\mathbf{X}\beta_2$, etc. [6]. In summary, the principal components can be represented by the system:

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta}. \tag{3}$$

To provide more details and a common notation, we outline the typical use of PCA as it was applied in this report, noting that steps 2–4 are inherent to the Python implementation of PCA provided by the sklearn.decomposition library [8]:

1. Format the data and create the data matrix, $\mathbf{X}$. We assume the data come from a number of samples N generated from a process. Typically, we might have N samples from physical experiments, observational studies, or computational simulations. In the situation described in this report, the samples come from a computer emulation called *minimega*.

   Each sample is a vector; there is a sequence of data for each sample. This is sometimes called functional data analysis [9], especially when the data represent functional values over time, also called time series data. If each sample contains a data vector of length L, the matrix dimension of $\mathbf{X}$ is $N \times L$. The objective of our dimension reduction analysis is targeted on the variable dimensions, L, where the

population statistics for each variable dimension is inherent in the N observations of each.

2. Center the data. This involves taking the mean of each column of $\mathbf{X}$ and subtracting it from that column as shown in Equation 4, where $\mathbf{X}_c$ denotes the centered matrix, and $X_{i,j}$ denotes the data matrix value at row i and column j.

$$\mathbf{X}_c = \begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ X_{i,1} - \overline{X}_{:,1} & X_{i,2} - \overline{X}_{:,2} & \dots & X_{i,L} - \overline{X}_{:,L} \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \tag{4}$$

3. Generate the covariance matrix of the centered data matrix $\mathbf{X}_c$ and take the eigendecomposition of it. The covariance matrix of the normalized data is calculated as:

$$V = \frac{{\mathbf{X}_c}^T \mathbf{X}_c}{N} \tag{5}$$

   One then performs a singular value decomposition of V to generate the eigenvectors $\boldsymbol{\beta}$. [5] Typically, the process is only carried out until the M principal components are identified that account for a certain percentage of the variance. The first eigenvector (with the largest eigenvalue) of V corresponds to β_1, the second eigenvector of V corresponds to β_2, etc. Note that the full $\boldsymbol{\beta}$ can be of size $L \times L$ but typically is smaller, $L \times M$.

   To determine M, typically one examines the eigenvalues of the covariance. [6] The trace (sum of the diagonal elements) of the covariance is the sum of the variances of the original L variables. The percent total variance explained by principal component k is given by $\lambda_k / tr(V)$, where λ_k is the k^{th} eigenvalue corresponding to principal component k. By summing the first M eigenvalues and dividing by the trace of the covariance matrix, one can obtain the total variance in the data explained by the first M principal components. Often, a few principal components can explain 95% or more of the variance, which demonstrates the dimension reduction from L variables to M components.

4. To perform a prediction, we can simply multiply the factor scores by the loadings and add back the mean vector.

$$\mathbf{X}_{pred} = \mathbf{Y}{\boldsymbol{\beta}_M}^T + \overline{\mathbf{X}} \tag{6}$$

A final implementation note: the singular value decomposition is usually not performed on the covariance matrix but on the centered data matrix. Some care must be taken in this case to ensure the proper normalization of the eigenvalues and variance information.

2.2 XPCA: Extending PCA to Combinations of Discrete and Continuous Variables

The standard PCA assumes the data has a normal multivariate distribution. That is the matrix are assumed to be continuous and each column has a normal marginal distribution. The work of Clifford Anderson-Bergman, Tamara Kolda, and Kina Kincher-Winota [1] extends PCA to account for heterogenous variables including continuous, binary, and ordinal or discrete. They do this by modifying the semiparametric copula-based principal component analysis (COCA) method [3], which combines a Gaussian copula with nonparametric marginals. A copula is a way to specify dependence amongst random variables. [7]

In [1], the authors propose a new extended PCA (XPCA) method that uses a Gaussian copula and nonparametric marginals and accounts for discrete variables in the likelihood calculation. The authors use empirical distribution functions for the nonparametric marginals and derive the likelihood function and a fitting algorithm which can be applied to estimate the eigenvectors and eigenvalues. The XPCA approach is similar to PCA in that it can be used to find latent structure in the data and perform dimensionality reduction. In the application of interest in this report, all variables are discrete, not a mix of continuous and discrete variables.

2.3 Time Series Dimension Reduction

In time series data, the variables for each observed sample point are simply the points in that time series. There can be issues with discretization of time series data. Chapter 8 of [9] is titled "Principal components analysis for functional data." It has a nice presentation of how to interpret the principal components for time series as well as many practical numerical implementation details.

The objective in port scanning cyber emulation experiments is to track an attacker's knowledge at a particular time, resulting in long vectors containing time series data characterized by discrete values. Therefore, not only do we need to exercise caution with PCA methods used on discrete realizations, but considerations of functional data analysis are also fundamental to our approach.

3 Cyber Emulations of Port Scanning

In emulation of cyber attacks, port scanning is a key phase in the development of an attacker's knowledge relating to vulnerabilities (e.g., open ports) in a cyber network. Once an attacker gains such knowledge they will then continue with the execution of their attack by distributing payloads to the vulnerabilities they have just learned there is access to. Through this in-

spection, each key component of an attacker threat chain is critical to the validation of cyber threat modeling. Details for the validation methods used by additional team members on a parallel effort can be found in [11]. It is the focus of our efforts outlined in this report to explore methods for obtaining computational efficiency for modeling the emulation of port scanning.

In this section, we provide a brief description of the emulation experimental design for port scanning. For simplicity and control of additional sources of variance, these experiments are independent from the intrusion detection system (IDS) emulations. Taking the resulting realizations generated by the scanning experiments, we then provide a detailed comparison between traditional PCA and XPCA in providing the top components that capture the greatest variance of the system while maintaining structural fidelity in the latent space representations.

3.1 Experiments

Port scanning is a pivotal stage of the attack chains in cyber security. For this reason, we focused our analysis on the results of the emulation experiments that generate realizations for two distinct port scanning strategies. The first is considered a "Slow & Stealthy" strategy where the attacker prioritizes avoiding detection and forgoes any risks related to the time it takes to learn each new vulnerability. In the slow and stealthy approach, the attacker would choose a host-group size of 4 parallel ports probed simultaneously and a delay of 10 seconds. Alternatively, the second strategy is referred to as the "Fast & Loud" approach. This approach, the attacker prioritizes expediency at the risk of early detection and consequently misses the opportunity for learning more vulnerabilities in the network. In the fast and loud strategy, the attacker would instead choose to use a host-group size of 6 and a scanning delay of 5 seconds.

We provide Table 3.1 that maps the scanning scenario type to the long name we use for archiving each of the experiments. Table 2 provides the summary of parameters for each aspect of the experiments.

Experiment Type	**Reference Name**
Slow & Stealthy	20190923_c189_parallel_eab_demo
Fast & Loud	20190925_c24_parallel_eab_demo

Table 1: **Experimental Run Reference Names:** The experimental type allows us to reference quickly the type of experiment that was run for the port scanning problem. The long name captures the date of the experiment, the node it was run on, whether is was run in parallel, and the targeted objectives for running the experiments.

As described by Vugrin *et. al.*, the scanning experiments are performed in a virtual test bed, run on the open source virtual machine management

software *minimega* [11]. For automated experiment scenario management, *ScOrch* is a Sandia developed python package that is integrated with minimega to provide efficient instrumentation and data extraction. For emulating a scanning scenario, the popular open source utility *Nmap* is used to scan hosts and learn what services are available on each [10]. With these tools, we are able to emulate a port scanning scenario with 24 Remote Terminal Units (RTUs) represented by: 4 open, 8 closed, and 12 filtered. In this experiment the objective is to simulate a CRASHOVERRIDE attack. In this scenario we assume that the attacker already has remote access on an Industrial Control System (ICS). With control over a Supervisory Control and Data Acquisition (SCADA) machine (e.g., an engineering workstation), the attacker scans for open ports for which they can then deploy a system override protocol. Probing ports at random, the attacker is able to learn what type of port they have targeted by the ssh response that is return (open or closed) or the lack of a response (filtered or dropped).

In a real-world scenario, the response from a scanning probe can be affected by external interference (e.g., background traffic). To emulate the effects of external interference resulting in a packet loss, we assume a 10% chance that a response to a scanning probe will be dropped. Therefore, the realization that no response is sent back from a probe can not be assumed to be a filtered port; there is a possibility it was either an open or closed response that was lost when a packet was dropped. To mitigate the loss of potentially valuable information, an attacker will consider resending the probe. At the onset of the scanning, a decision will be made for the number of times to resend a probe. Assuming the packet loss is an independent event for each probe, using just one resend lowers the attacker's likelihood of missing an open port from a 10% chance to a 1% chance. However, this comes with an increased computational cost for the attacker as well as an increased probability that the attacker will be detected.

The second source of random behavior is emulated by permuting the order of the ports to be scanned. The order that an attacker will scan the ports will determine if an open port is found on the first probe, or maybe not until the 10^{th} or the 18^{th} probe. Therefore, we run 1000 (or more) emulations generating a time series of realizations representing the cumulative number of each type of port found at time t. Since a probe can result in no response, which can occur from sequential packet losses, we represent an attacker's knowledge of a system at time t to be the number of known open, closed, and *inconclusive* ports.

For quick reference, we provide Table 2 listing the key parameters for the emulytics scanning experiments.

For simplicity, we have reported only the pertinent details of the emulytics experiments relative to the port scanning scenarios in support of our analysis on dimension reduction. For those that are interested in learning more about the the cyber threat modeling we again direct the reader to [11].

Network	Parameter	Value
	Packet loss	10%
	Nodes	24 RTU: 4 open, 8 closed, 12 filtered
Nmap	Parameter	Value
	Number of resends	1
	Probe timeout	0.5 seconds
Slow, Stealthy	Parameter	Value
	Host-group	4
	Scan-delay	10 seconds
Fast, Loud	Parameter	Value
	Host-group	6
	Scan-delay	5 seconds

Table 2: **Scanning Emulation Parameters**

3.2 Evaluation of Dimension Reduction

To initiate the discussion for the dimension reduction analysis, we provide a further description of data resulting from the cyber emulytics scanning experiments. As mentioned in the previous section, the realizations generated by each of the 1000 runs of the experiment are a time series tracking the knowledge an attacker has of the network at each time step. It is important to note that although we will be analyzing a time series with a refined uniform mesh, this is not the format of the results tracked by *ScOrch*. Instead, during the run for the scanning scenario, the *pcap* file generated by *ScOrch* captures the timestamp and response of each port that is probed. Although this is sufficient for representing the knowledge obtained by an attacker at each time t, it does not provide the appropriate data structure to validate against a mathematical model for scanning. Therefore, the decision was made to generate the results of scanning to align with the same time discretization that is utilized by a mathematical model[1]. The results are a discrete time series of piece-wise constant step functions, where each uniform time step captures information in the *pcap* file documenting when knowledge about a port is obtained. For convenience, we provide Figures 1 - 3 for examples. These are all pulled from the fast and loud scanning strategy. Each of the port types are plotted independently from each other, since we will focusing the dimension reduction analysis on each port type as an independent analysis.

These example plots were selected from a handful of realizations to demonstrate different behaviors. The time series in example 1 in Figure 1

[1]For more information regarding the mathematical model of scanning and the validation of the cyber emulations, the reader can again reference Vugrin *et. al.* [11]

provides an illustration of a simple example that directly produces the behavior that is expected from the port scanning situation. Next, example 2 in Figure 2 provides a nice example for a closed port results in two sequential packet drops. In this situation, the knowledge obtained at the end of the scanning process is 4 open, 7 closed, and 13 inconclusive ports (note the correct behavior is supposed to be 4 open, 8 closed, and 12 inconclusive ports). Another interesting observation is that the attacker (lucky in this realization!) would learn of 2 open ports (50% of the vulnerabilities) from their first probe, noting that 6 hosts are used in this situation. Finally, example 3 in Figure 3 illustrates a realization for which the attacker will not learn about a single open port until about a third of the way into the scenario. At that time, they learn of three (out of four) open vulnerabilities.

Now consider the 1000 experimental runs that were used to generate 1000 time series realization for each of the three types of ports. The length of the time series will depend on the scanning strategy. For the slow and stealthy strategy, the attacker uses 4 hosts and a delay of 10 seconds. In this case, it will take approximately 175 seconds to scan the 24 RTU network. Alternatively, when the attacker uses a fast and loud strategy, the 6 hosts and 5 second delay will get the job done in approximately 60 seconds. Therefore, the original time series data dimensions are strategy specific. The data structure is a matrix where each row represents each of the 1000 realizations from the 1000 emulytic runs for the experiment. The columns of our datasets capture the original time series dimensions. Depending on the strategy, this will be either 60 seconds or 175 seconds, where each dimension represents one second in that time series.

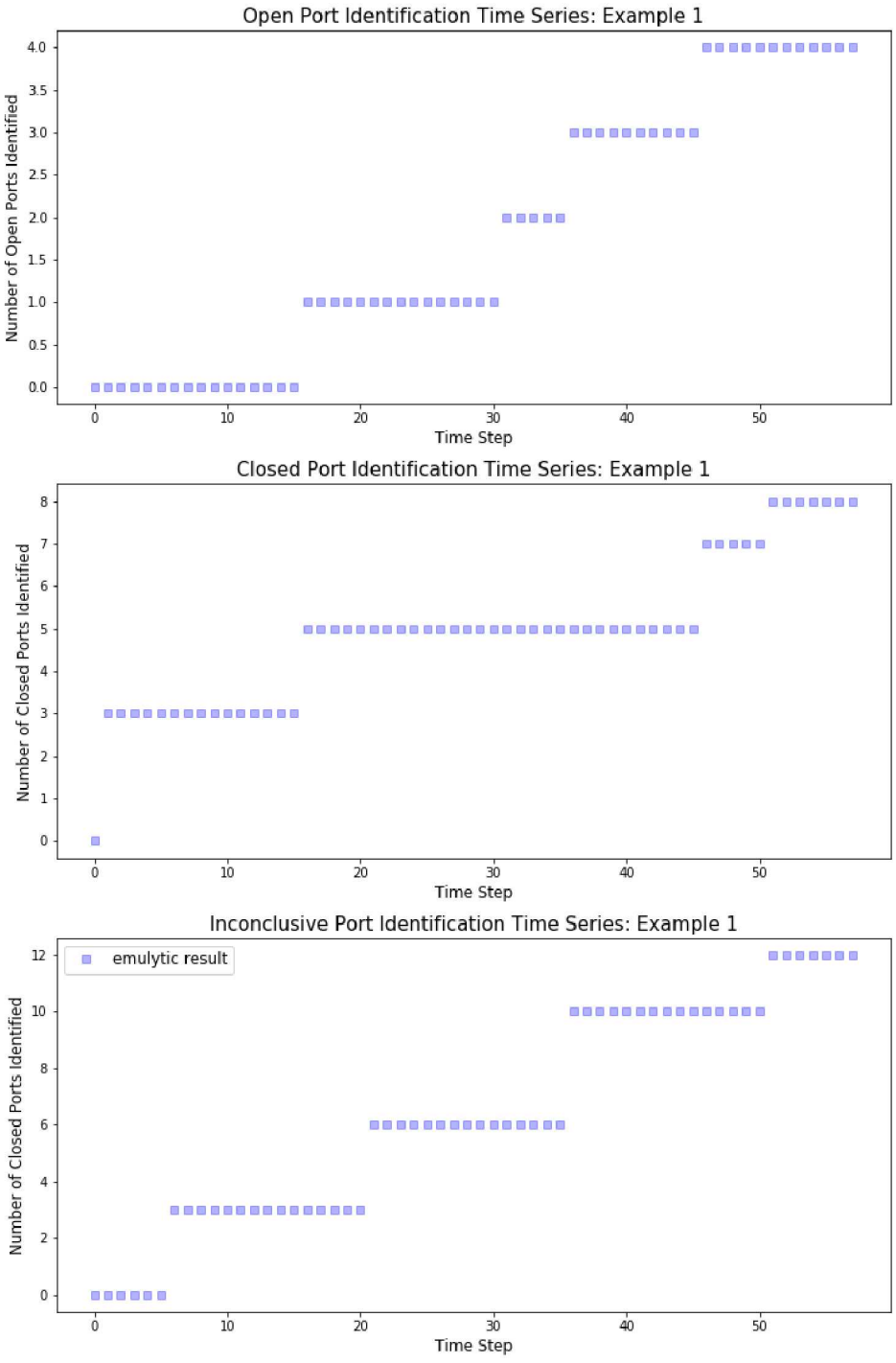

Figure 1: **Port Scanning Cyber Emulytics: Example 1**

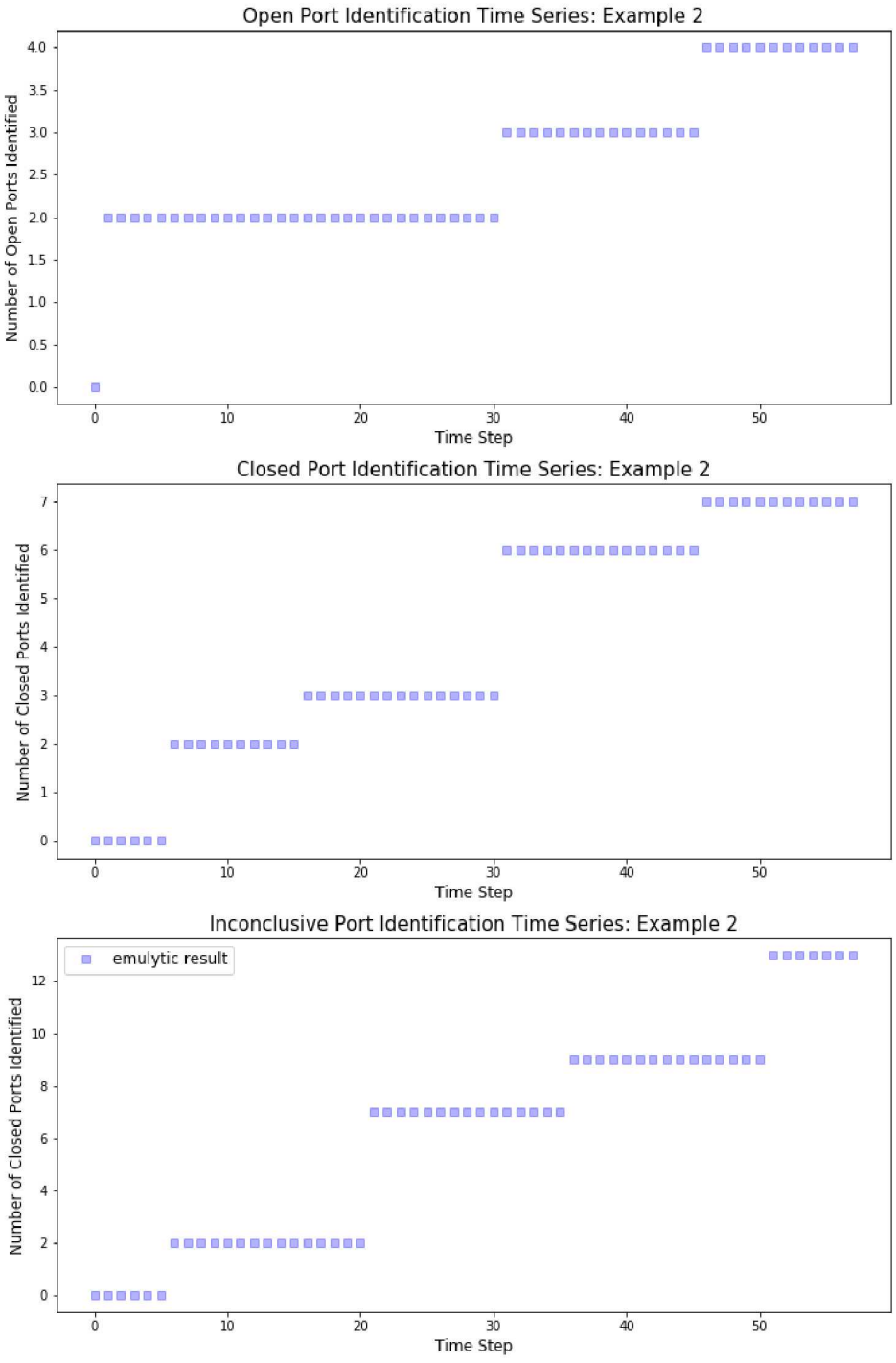

Figure 2: **Port Scanning Cyber Emulytics: Example 2**

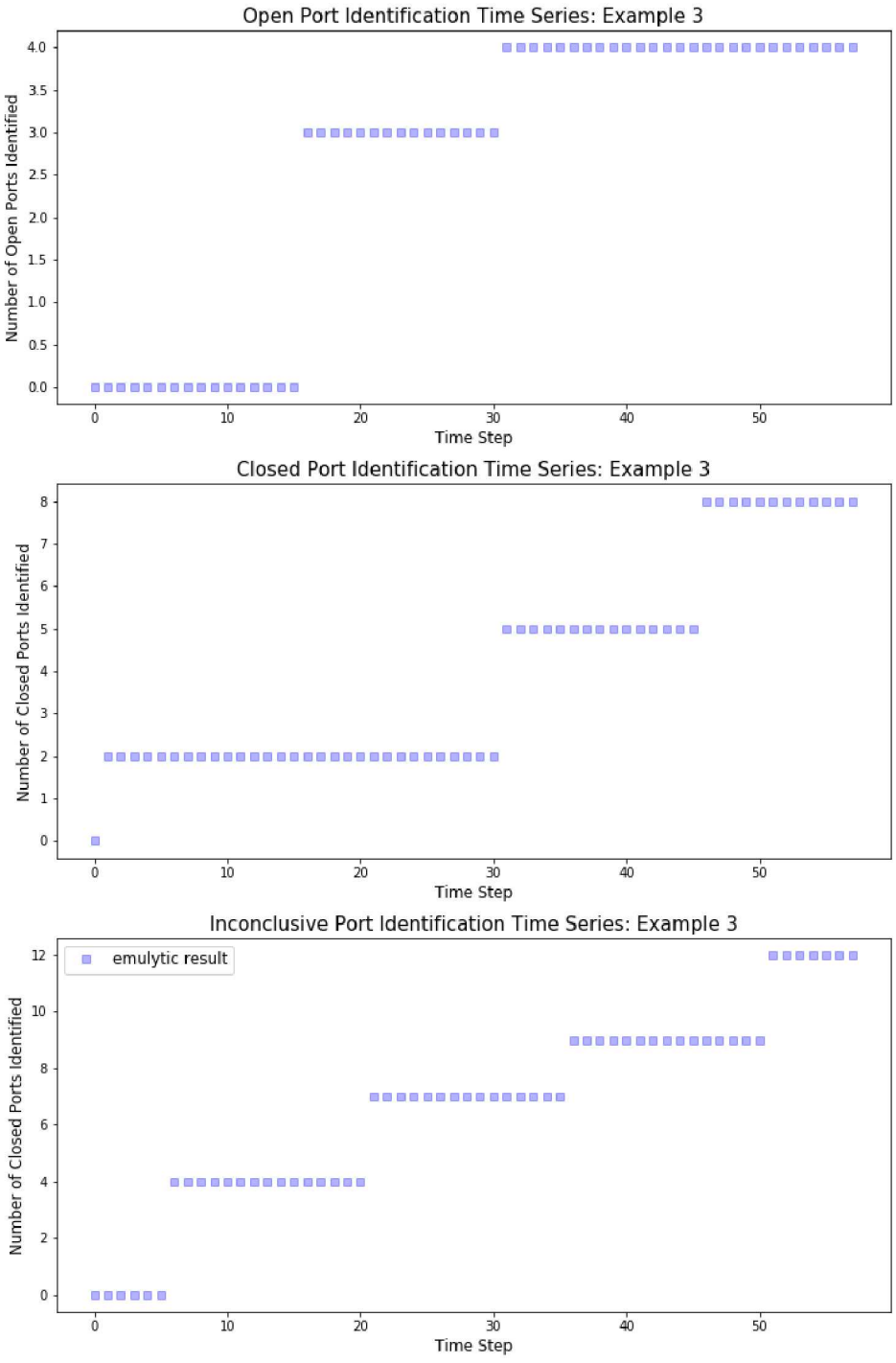

Figure 3: **Port Scanning Cyber Emulytics: Example 3**

Given the redundancy in the time series datasets that inherently capture such large ranges of variability, we now focus on the evaluation of a latent space representation that captures the majority of the variance with a minimal number of components.

As we mentioned earlier, we focused our exploratory data analysis on using XPCA in comparison to traditional PCA. This analysis involves the linear dimension reduction methods (1. PCA, and 2. XPCA),two different scanning scenarios (1. slow and stealthy, and 2. fast and loud), with three classes of time series realizations relating to each of the three port types (1. open, 2. closed, and 3. inconclusive). For each combination, we calculate the variance explained for the first principal component and the contributions from each of the subsequent components to capture the total variance explained up to the tenth component. As a primary objective, we focused on the comparison in total variance explained between PCA and XPCA. Figures 4 and 5 provide these results, where the shapes in the plots are used to indicate the port type and the colors distinguish PCA versus XPCA. The strategies are plotted separately given the distinction in their original dimension representations. In both strategies, we see that the variance explained in the first principal component for XPCA is drastically lower than that of that for PCA. This was an unexpected result and one that requires more research and attention to determining the discrepancy in the two methods that results in such distinction.

In Figure 4 we plot the total variance explained for the slow and stealthy strategy (175 original dimensions). This plot shows that the PCA components will converge close to 100% variance explained within the first 8-10 components. Alternatively, since we truncate the analysis at 10 components, we have yet to see this convergence in the XPCA components.

In consideration for the more aggressive scanning strategy, Figure 5 shows that both linear dimension reductions converge close to 100% variance explained within the first 10 components. Again, we see the convergence to full variance explained occurring more quickly with traditional PCA components. If capturing variance explained in the fewest number of components was our only objective, then it appears that traditional PCA outperforms XPCA.

To assess the fidelity of the latent space representation, we performed dimension reduction for each method at 6-dimensions and 10-dimensions then evaluated the reconstruction error back in the original time dimension. To measure the reconstruction error, we take the vector difference between the reconstructed time series with the original time series and evaluate the 2-norm of that difference. The results of this analysis have been summarized in Tables 3 - 6. Aligned with the corresponding variance explained, we provide the minimum, maximum, and mean reconstruction error across all 24 targeted analysis. For the 12 scenarios for which we reconstructed the original time series from n XPCA components, we are able to obtain

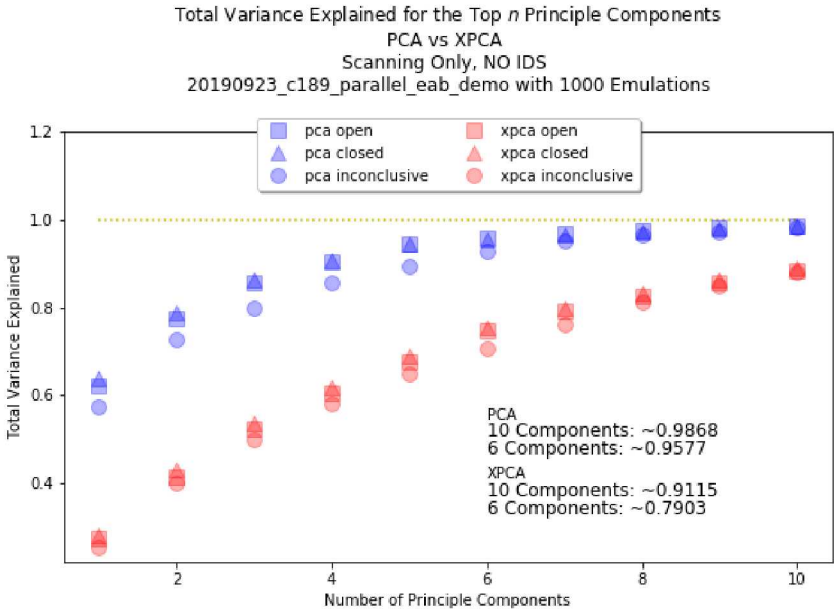

Figure 4: **Slow & Stealthy Variance Explained**

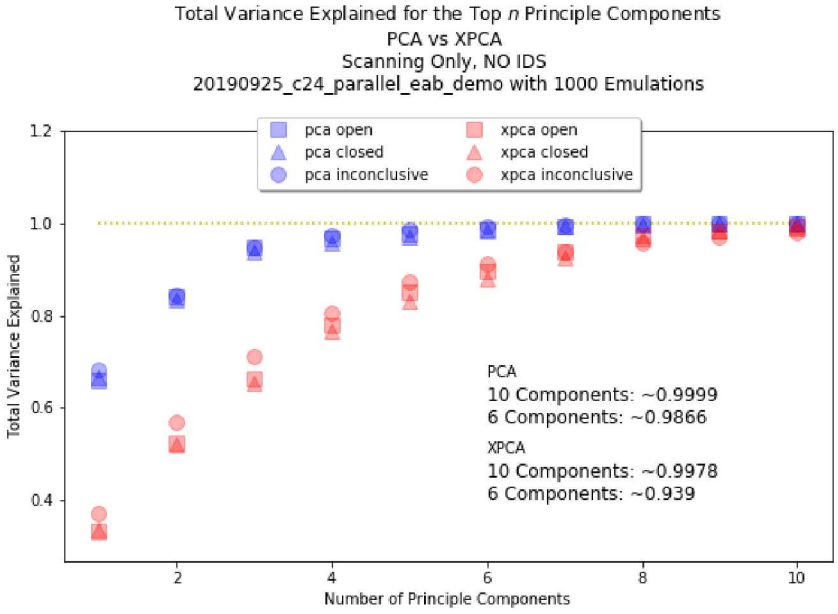

Figure 5: **Fast & Loud Variance Explained**

perfect reconstruction. In comparison to traditional PCA, the maximum reconstruction error from XPCA is approximately half that of the maximum

reconstruction error from PCA.

For a more detailed summary of all of the experiments and the corresponding reconstructions, we provide comparison plots in Appendix A.

Slow & Stealthy	Open	Closed	Inconclusive
Variance Explained	95.77%	95.63%	95.77%
Min	0.5964	0.7741	0.4948
Mean	1.7552	2.6293	2.5299
Max	7.1306	9.7563	13.039

Fast & Loud	Open	Closed	Inconclusive
Variance Explained	98.66%	98.22%	98.66%
Min	0.0235	0.1085	0.0105
Mean	0.4517	0.7616	0.1458
Max	3.5327	5.5633	7.3703

Table 3: **6-Dimension PCA Summary:** variance explained and reconstruction error (min, mean, max), for each experimental run (low-slow versus high-fast) accounting for all three categories of ports (open, closed, and inconclusive)

Slow & Stealthy	Open	Closed	Inconclusive
Variance Explained	74.83%	75.44%	70.56%
Min	0	0	0
Mean	0.18396	1.3698	1.9203
Max	6.3246	10.9545	13.0767

Fast & Loud	Open	Closed	Inconclusive
Variance Explained	89.58%	87.96%	91.35%
Min	0	0	0
Mean	0.0424	0.05	0.1
Max	2.2361	5.1962	7.8102

Table 4: **6-Dimension XPCA Summary:** variance explained and reconstruction error (min, mean, max), for each experimental run (low-slow versus high-fast) accounting for all three categories of ports (open, closed, and inconclusive)

Slow & Stealthy	Open	Closed	Inconclusive
Variance Explained	98.68%	98.47%	98.68%
Min	0.0617	0.1214	0.0377
Mean	0.7725	1.3402	1.0475
Max	6.2919	8.1558	8.1489

Fast & Loud	Open	Closed	Inconclusive
Variance Explained	99.99%	99.97%	99.99%
Min	0.0005	0.001	0.0021
Mean	0.0088	0.021	0.0202
Max	1.4464	1.7595	2.5931

Table 5: **10-Dimension PCA Summary:** variance explained and reconstruction error (min, mean, max), for each experimental run (low-slow versus high-fast) accounting for all three categories of ports (open, closed, and inconclusive)

Slow & Stealthy	Open	Closed	Inconclusive
Variance Explained	88.48%	89.03%	87.97%
Min	0	0	0
Mean	0.0455	0.1258	0.3048
Max	4.4721	7.746	8.9443

Fast & Loud	Open	Closed	Inconclusive
Variance Explained	99.34%	99.07%	98.07%
Min	0	0	0
Mean	0.002	0.009	0.003
Max	2.2361	5.1962	7.8102

Table 6: **10-Dimension XPCA Summary:** variance explained and reconstruction error (min, mean, max), for each experimental run (low-slow versus high-fast) accounting for all three categories of ports (open, closed, and inconclusive)

4 Conclusion

In this report, we compare PCA with XPCA on discrete value time series data from port scanning emulation experiments. The objective for this analysis is to derive a lower dimensional representation for a discrete time series function composed of piece-wise constant steps. The PCA or XCPA can find the key principal components that capture the greatest variance in the stochastic experimental realizations. With these components, we can then design a surrogate model to capture the random behavior that is the driving factor for that variance.

Our main finding of this work is that PCA performs better with respect to variance explained but worse with respect to reconstruction error on these discrete time series data sets. Our original goal was to target a lower dimensional representation that captures a majority of the variance and drops a marginal amount of variance based on a predefined threshold. It seems that traditional PCA can do this job with fewer components, but does so at the risk of introducing loss of fidelity when we examine the reconstruction error. Our assessment is that XPCA requires more components but then is able to do a better job than PCA because there is not "noise" from the sum of continuous components which do not quite add up to the discrete values in the PCA reconstructions as shown in Figures 6-17 in the Appendix. Therefore, the XCPA is better suited for obtaining acccurate reconstructions of time series data to target a surrogate model that efficiently captures the salient random behavior with only marginal loss of fidelity.

The next steps will be to look into the computational complexity tradeoff between the algorithms, relative to the two objectives of reconstruction error and variance explained.

A Reconstruction Visualizations

In addition to understanding the variance explained, we also emphasize the reconstruction error comparison between PCA and XPCA. Although a table with summary statistics is a useful reference for seeing these distinctions at a high level, we have provided here the direct comparisons between the emulytics experiment results compared to the reconstructions derived from each of the PCA and XPCA latent spaces. These tell a more comprehensive visual story for how the attack strategy for scanning, as well as the choice of dimensions, results in differences between PCA and XPCA.

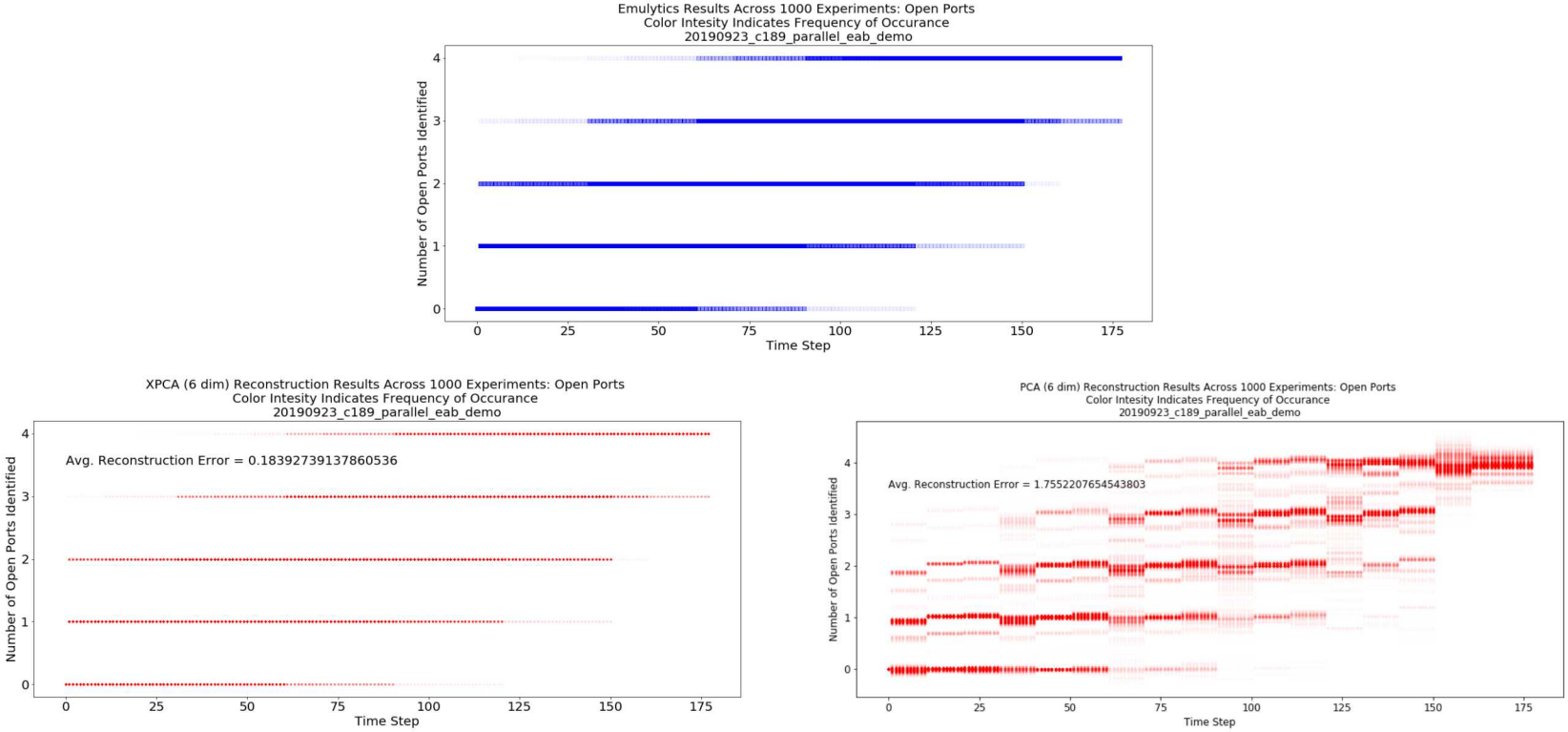

Figure 6: **Slow & Stealthy 6D Open Ports Reconstruction Error**

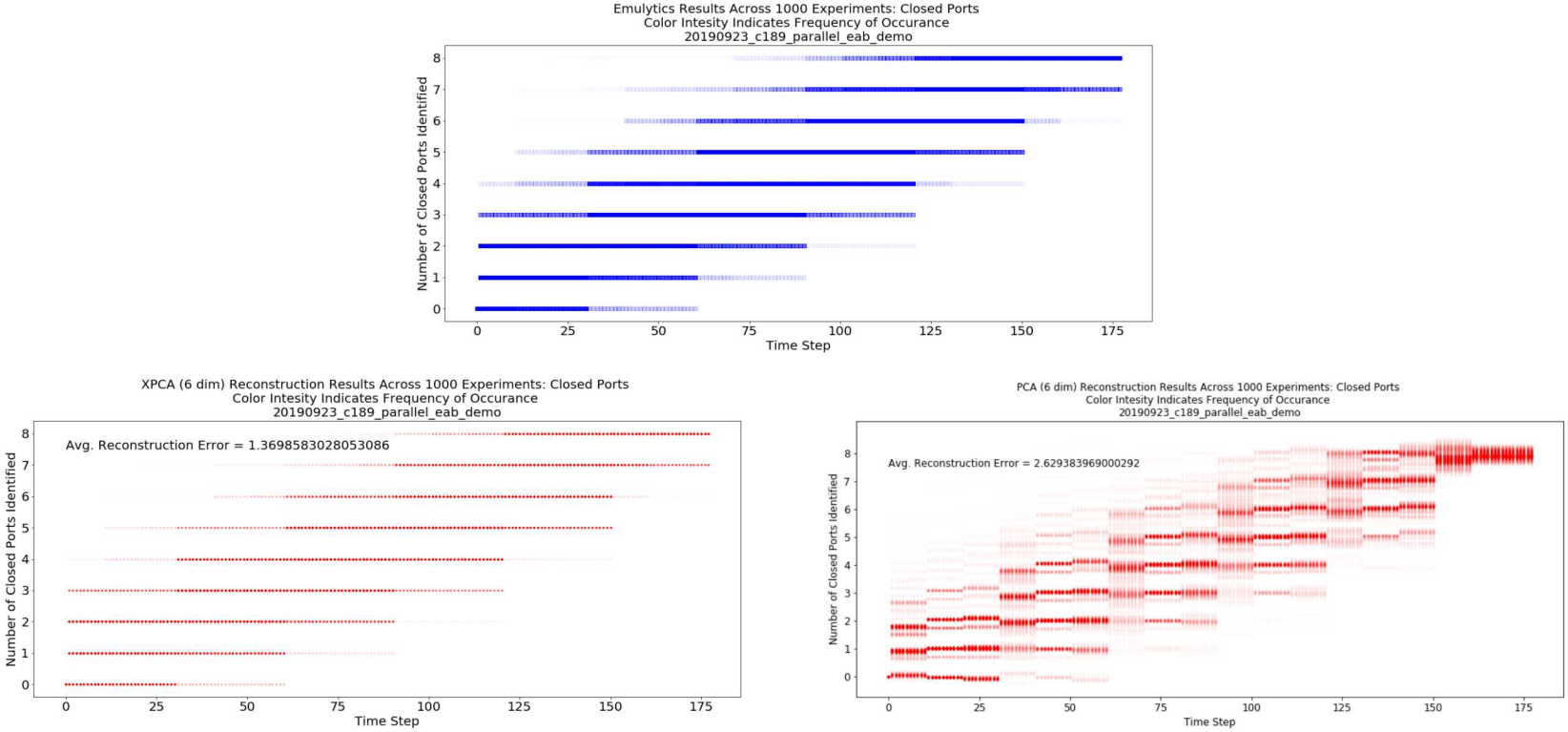

Figure 7: **Slow & Stealthy 6D Closed Ports Reconstruction Error**

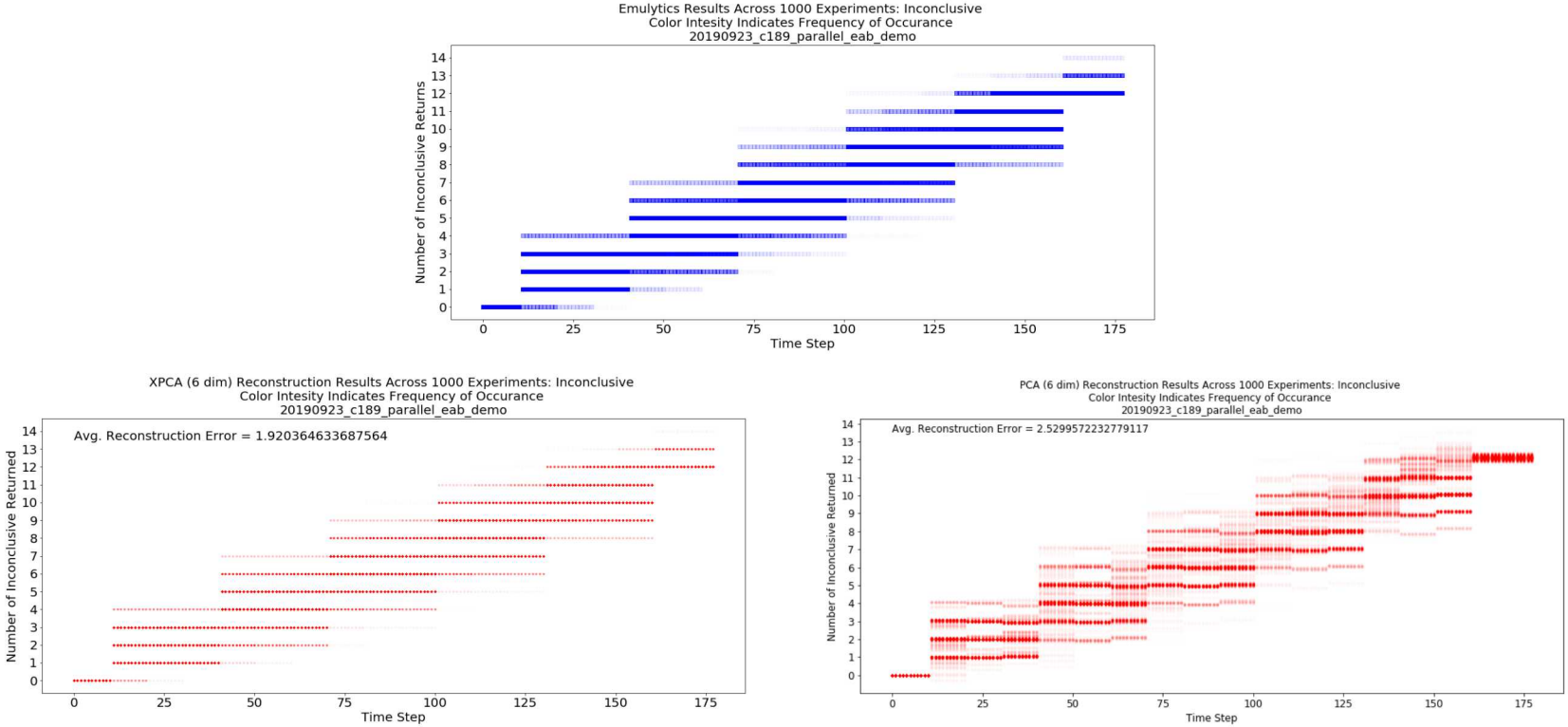

Figure 8: **Slow & Stealthy 6D Inconclusive Ports Reconstruction Error**

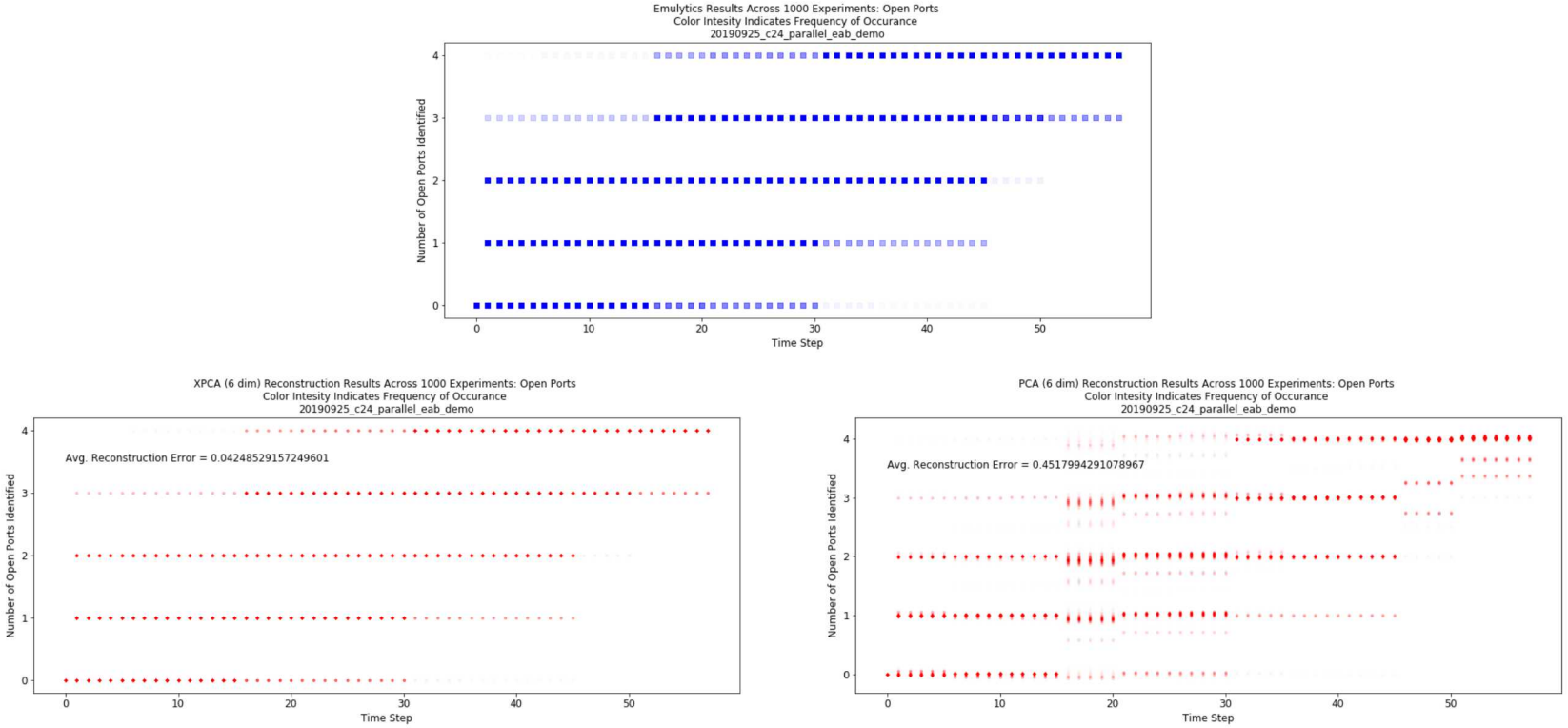

Figure 9: **Fast & Loud 6D Open Ports Reconstruction Error**

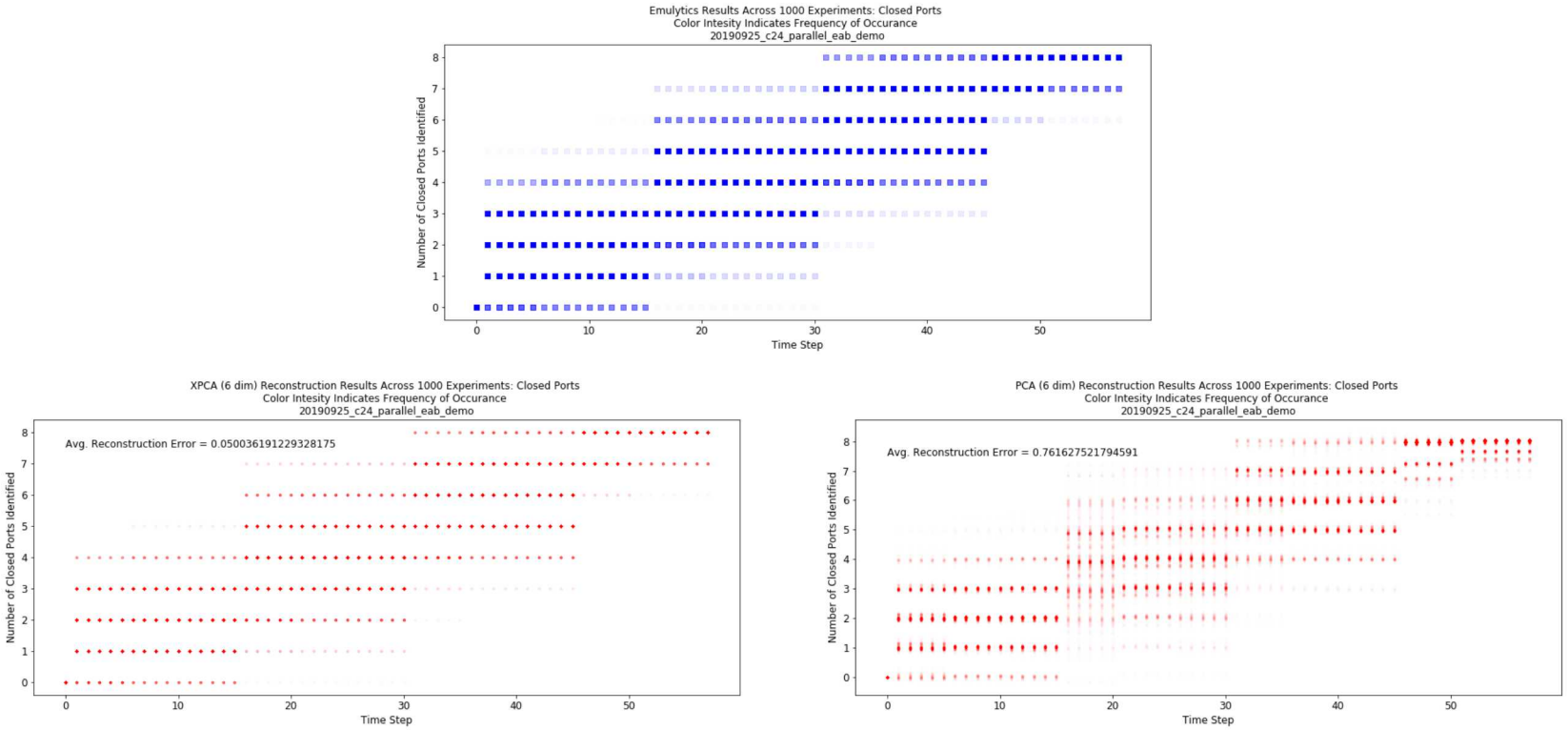

Figure 10: **Fast & Loud 6D Closed Ports Reconstruction Error**

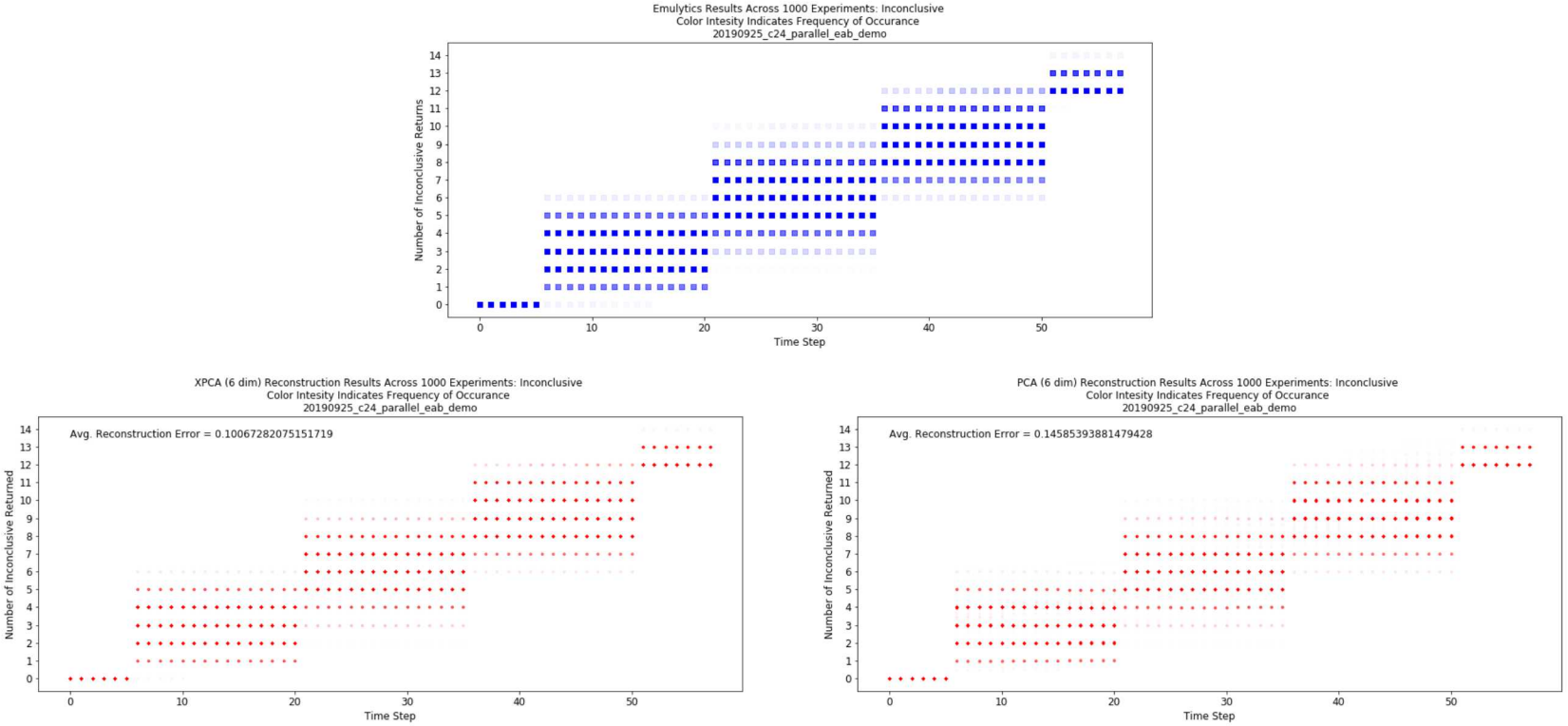

Figure 11: **Fast & Loud 6D Inconclusive Ports Reconstruction Error**

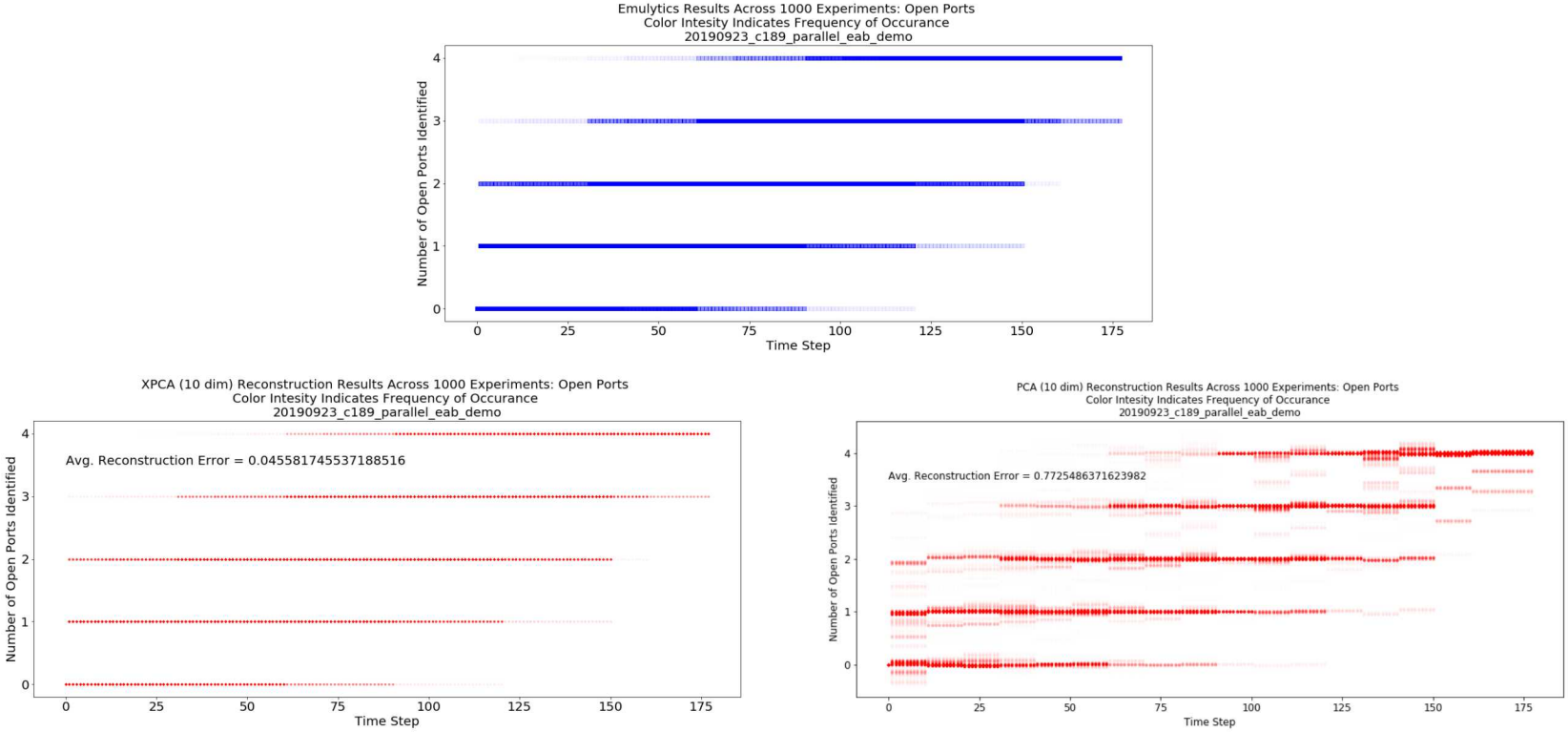

Figure 12: **Slow & Stealthy 10D Open Ports Reconstruction Error**

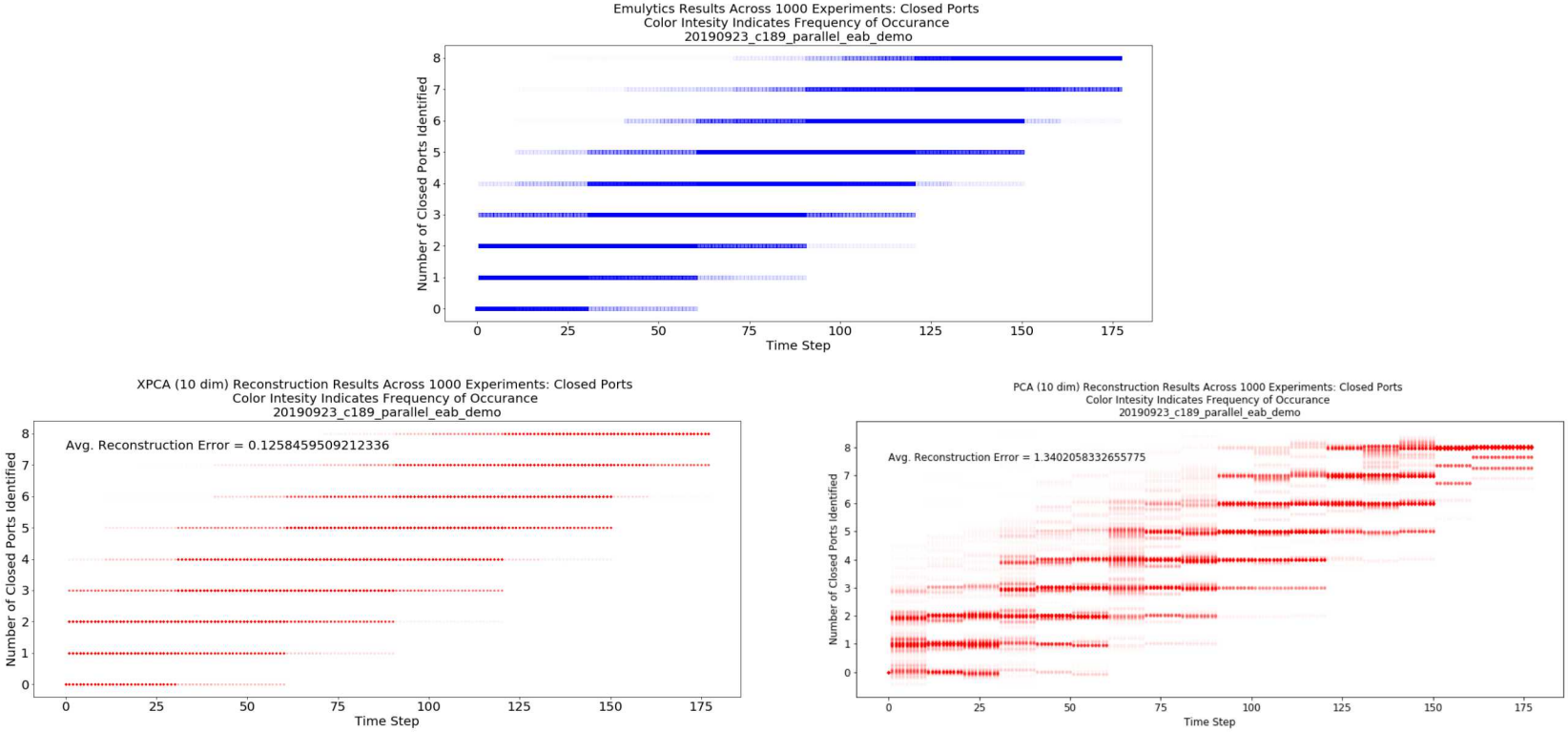

Figure 13: **Slow & Stealthy 10D Closed Ports Reconstruction Error**

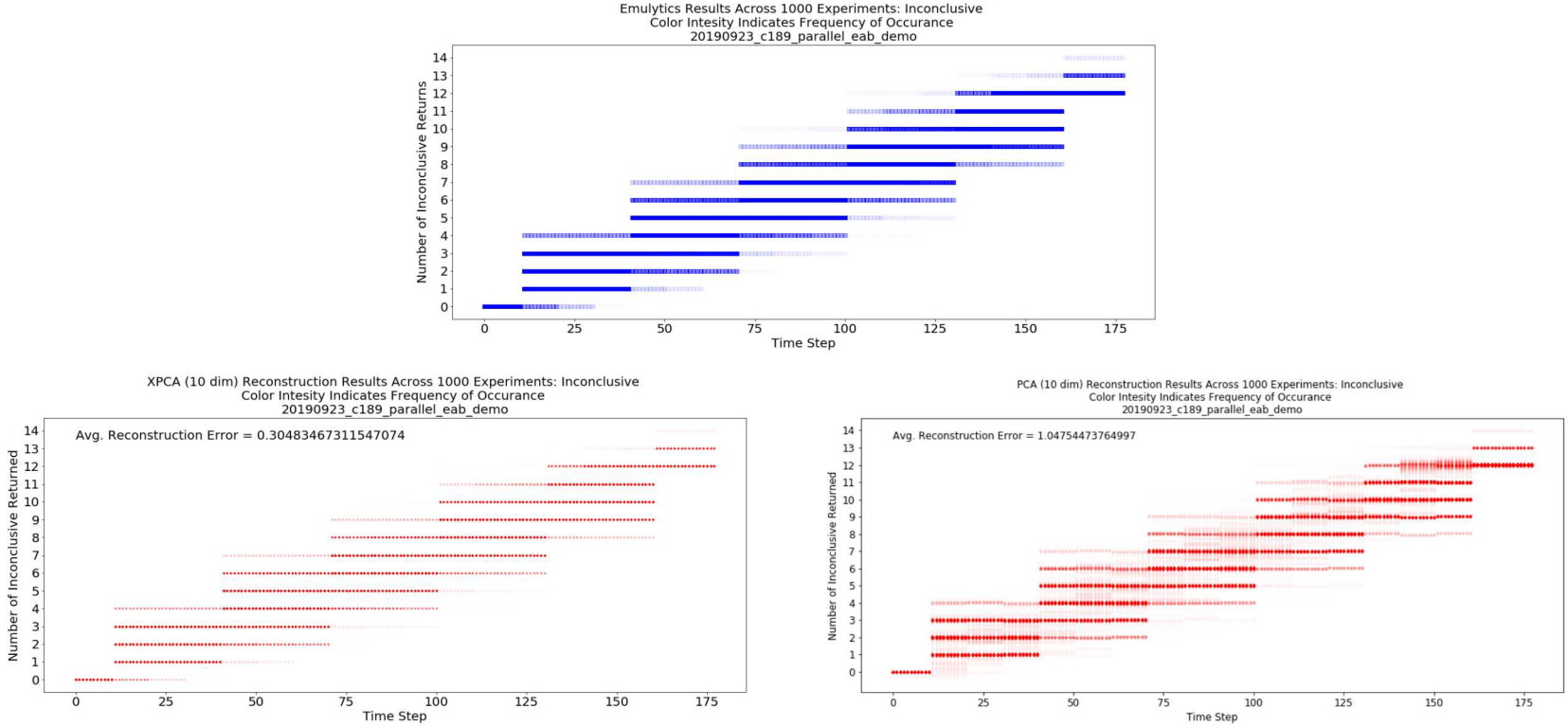

Figure 14: **Slow & Stealthy 10D Inconclusive Ports Reconstruction Error**

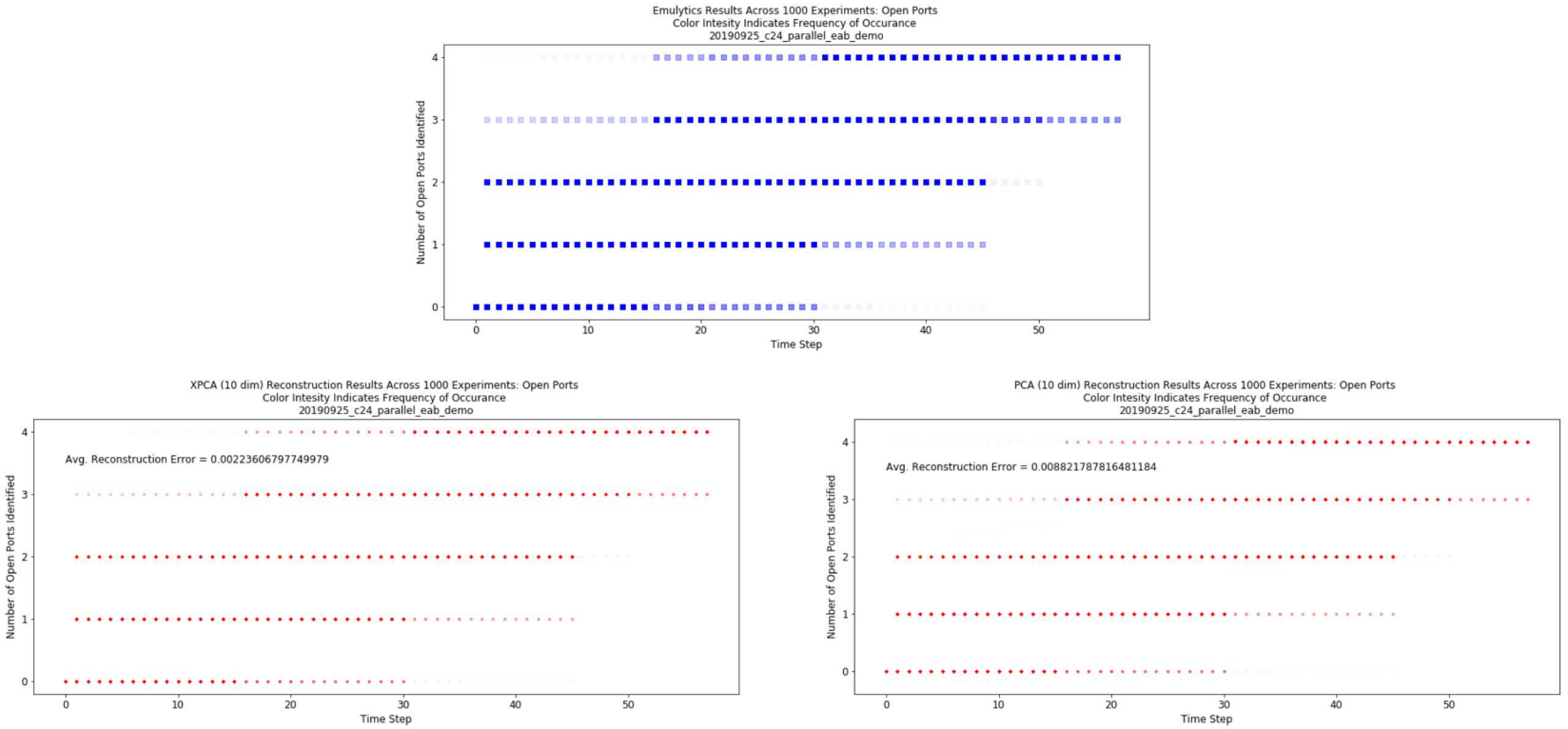

Figure 15: **Fast & Loud 10D Open Ports Reconstruction Error**

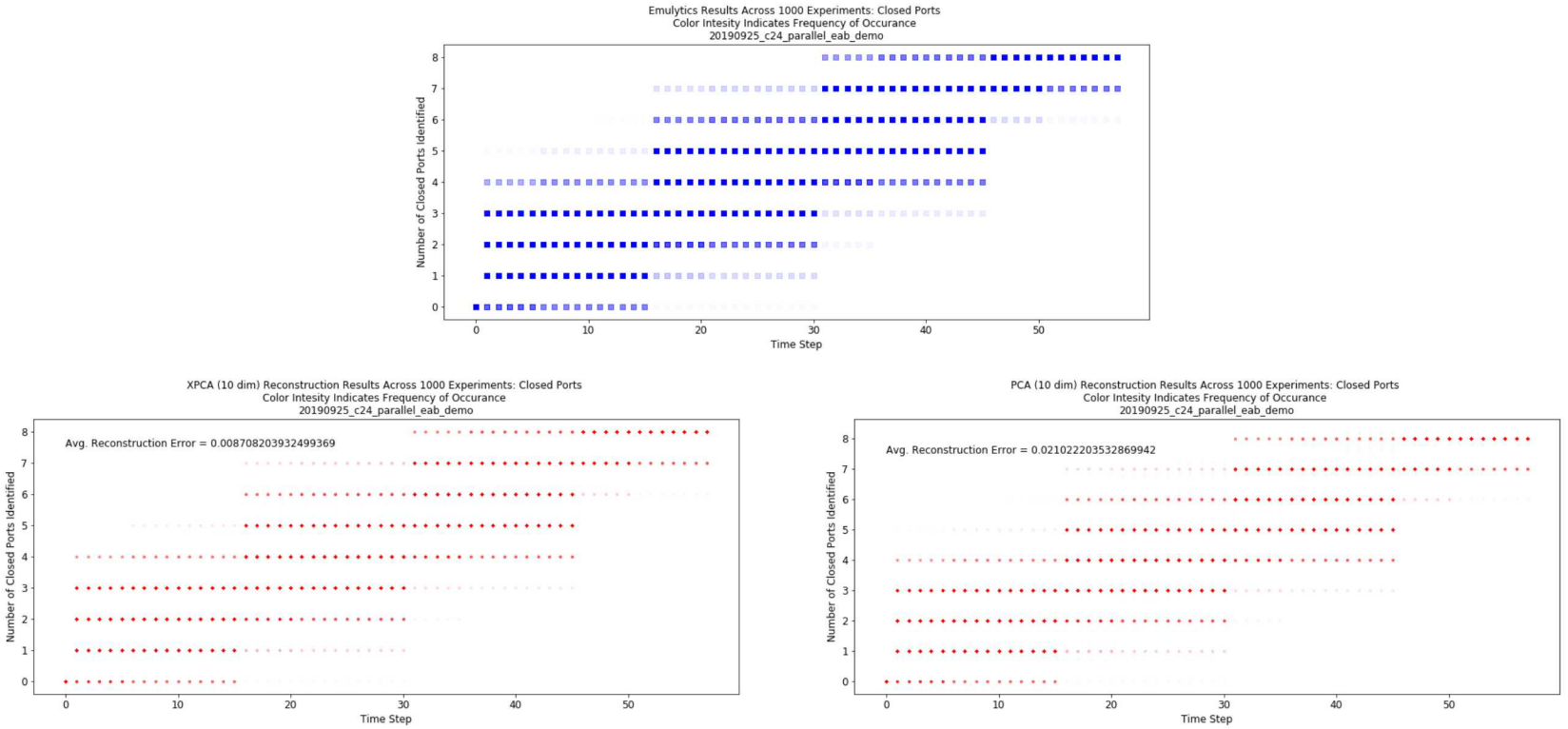

Figure 16: **Fast & Loud 10D Closed Ports Reconstruction Error**

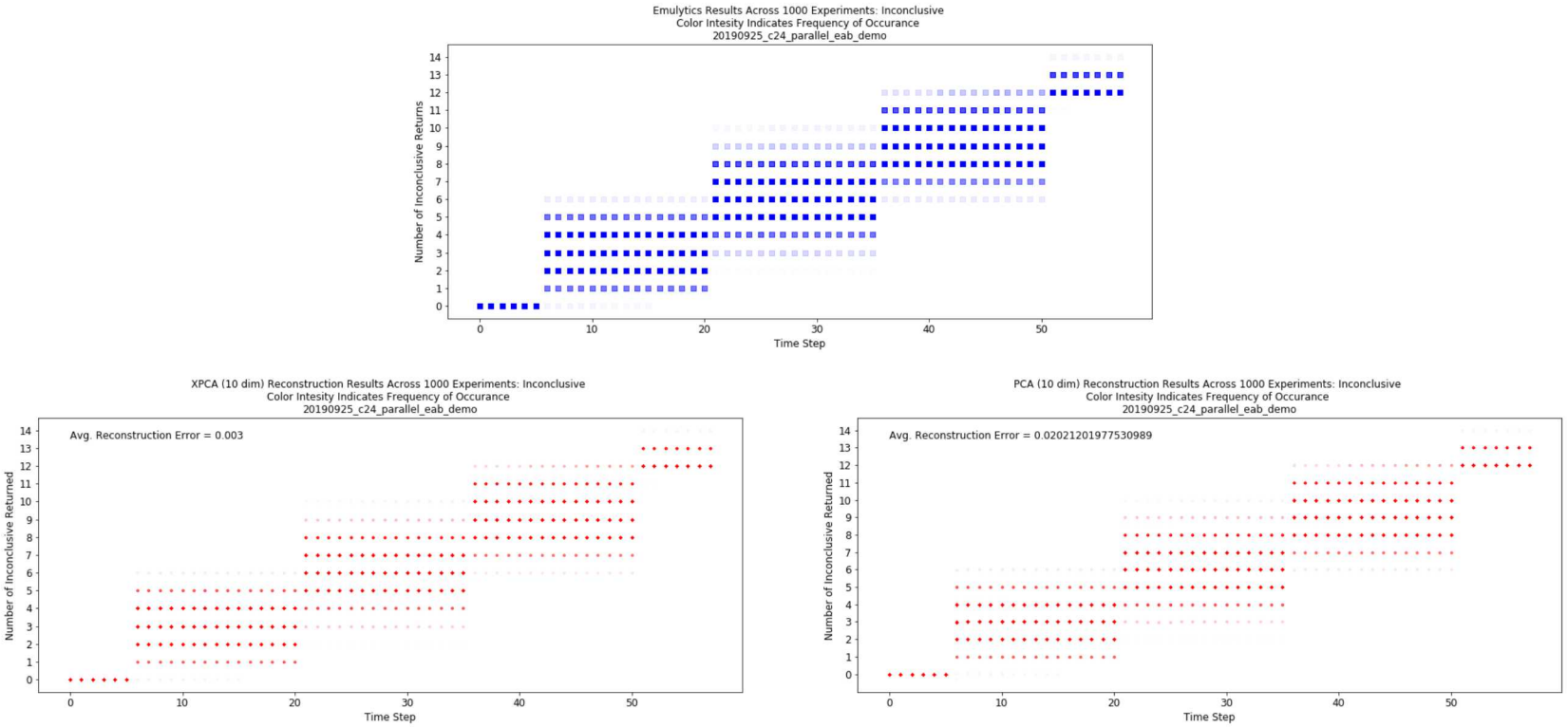

Figure 17: **Fast & Loud 10D Inconclusive Ports Reconstruction Error**

References

[1] Clifford Anderson-Bergman, Tamara G Kolda, and Kina Kincher-Winoto. Xpca: Extending pca for a combination of discrete and continuous variables. *SAND2018-8212O; arXiv preprint arXiv:1808.07510*, 2018.

[2] Karl Pearson F.R.S. Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[3] Fang Han and Han Liu. Semiparametric principal component analysis. In *Advances in Neural Information Processing Systems*, pages 171–179, 2012.

[4] H Hotelling. Analysis of a complex of statistical variables into principal components. *J. Educ. Psychol.*, 24:417441, 498520, 1933.

[5] Ian T Jolliffe. *Principal component analysis, second edition.* Springer Series in Statistics, 2002.

[6] Ian T Jolliffe and Jorge Cadima. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202, 2016.

[7] Roger B Nelsen. *An introduction to copulas.* Springer Science & Business Media, 2007.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[9] James O Ramsay and Bernard W Silverman. *Applied functional data analysis: methods and case studies.* Springer, 2007.

[10] Jorge Savaglia and Ping Wang. Cybersecurity vulnerability analysis via virtualization. *Issues in Information Systems*, 18(4), 2017.

[11] Eric D. Vugrin, Jerry Cruz, Christian Reedy, Tom Tarman, and Ali Pinar. Cyber threat modeling and validation: Port scanning and detection. In *Proceedings of the 7th Annual Symposium on Hot Topics in the Science of Security (in prep.)*, HotSoS 2020, New York, NY, USA, 2020. Association for Computing Machinery.

[12] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

DISTRIBUTION

Email—Internal

Name	Org.	Sandia Email Address
Dornburg, Courtney S.	5954	ccdornb@sandia.gov
Tarman, Thomas D.	5682	tdtarma@sandia.gov
Cruz, Gerardo	5682	gcruz@sandia.gov
Vugrin, Eric D.	5621	edvugri@sandia.gov
Hart, Derek H.	5621	derhart@sandia.gov
Gearhart, Jared L.	8721	jlgearh@sandia.gov
Turner, Daniel Z	1463	dzturne@sandia.gov
Thayer, Gayle E.	8762	gthayer@sandia.gov
Technical Library	01977	sanddocs@sandia.gov

This page left blank

This page left blank

Sandia National Laboratories

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.