AN ABSTRACT OF THE DISSERTATION OF

Jessica R. Curtis for the degree of Doctor of Philosophy in Radiation Health Physics presented on August 26, 2019.

Title: Special Nuclear Material Classification and Mass Content Estimation Using Temporal Gamma-Ray Spectroscopy and Machine Learning Methods

Abstract approved:

Steven R. Reese

Improving the methods used for the detection and estimation of fissile material mass content is essential to nuclear security and safeguards to ensure special nuclear material (SNM) accountability, control, safety and security. With the continued expansion of the nuclear industry and the need for safe management of spent fuel, improving existing nondestructive assay (NDA) techniques is imperative. In the present research, we are particularly interested in Pu-239 and U-235 content. Following thermal neutron-induced fission, temporal gamma-ray spectroscopy and machine learning methods were used to classify pure samples of Pu-239 and U-235. Further, regions of interest identified during data pre-processing were utilized for computing the relative mass content of the fissile materials. The temporal gamma-ray spectroscopy method takes advantage of the time-dependent decay of fission products. Without prior knowledge of peak locations or their associated energies, temporal patterns characteristic of radioactive decay were identified within the complex fission product gamma-ray spectra below 3 MeV. Following feature generation and feature selection, Welch's t-test was employed to tease out regions of interest used as "fingerprints" to create profiles of Pu-239 and U-235 to be used as input into four machine learning architectures: decision tree, random forest, neural network and Bayesian network. Classification accuracy ranged from 98-100% for all four classifiers. Initial results for determining relative mass content are promising as several regions of interest showed mass estimates within two standard deviations uncertainty for at least one of the fissile materials.

Special Nuclear Material Classification and Mass Content Estimation Using Temporal Gamma-Ray Spectroscopy and Machine Learning Methods

by

Jessica R. Curtis

A DISSERTATION

submitted to

Oregon State University

in partial fulfillment of

the requirements for the

degree of

Doctor of Philosophy

Presented August 26, 2019

Commencement June 2020

ACKNOWLEDGMENTS

I want to first thank God for blessing me with this opportunity and for giving me the strength and encouragement to see it through. I am extremely grateful to my advisor and mentor, Dr. Steve Reese, who has taught me more than I could ever give him credit for. I am beyond thankful for the endless hours he spent hashing out ideas, answering questions and discussing career plans. His door was always open. Thank you for guiding and supporting me over the past five years. I would like to thank Dr. Ophir Frieder for his insightful suggestions as well as his guidance in navigating the world of machine learning. I would like to express my heartfelt gratitude to Dr. Camille Palmer for her guidance and support both personally and professionally over the years. I am grateful to Dr. Sarah Emerson for her assistance and support throughout the research design and data analysis processes. My sincere thanks goes to Scott Menn for his help during data collection and Steve Smith for sharing his knowledge of the Fast Rabbit Pneumatic Transfer System. I will forever be thankful for the time both set aside to help me during my tenure at OSU. Finally, I am extremely grateful to my family and friends for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this dissertation. This accomplishment would not have been possible without them.

TABLE OF CONTENTS

TABLE OF CONTENTS (Continued)

TABLE OF CONTENTS (Continued)

LIST OF FIGURES

LIST OF FIGURES (Continued)

LIST OF TABLES

LIST OF TABLES (Continued)

DEDICATION

This dissertation is dedicated to my beloved husband and best friend, Doug Woods (1985 - 2019), who always believed in me. Through his endless love, support and encouragement I was able to make it through this journey. He showed me what unwavering faith looks like and I will forever be grateful for all that he taught me. While our time together was cut short, I am blessed to have been loved by such an amazing man. Your light continues to shine through the many lives you touched. Until we meet again...

1 Introduction

1.1 Motivation

Improving the methods used for detection and quantification of fissile material mass content is essential to nuclear security and safeguards as special nuclear material (SNM) accountability, control, safety and security remain to be of great importance. With the need for safe management of spent fuel, disarmament of nuclear weapons, and tracking of illicit nuclear material improving nondestructive assay (NDA) techniques is imperative. Knowledge of Pu-239 and U-235 is of particular interest.

U-235 and Pu-239 have similar fission yield curves with the majority of fission fragments having a mass number of 94 and 140 [52]. Figure 1 shows the fission product yields from the thermal neutron induced fission of U-235 and Pu-239, where plutonium is shifted toward higher masses. The similarities in their fission yield curves present a challenge when differentiating these materials in an unknown sample.

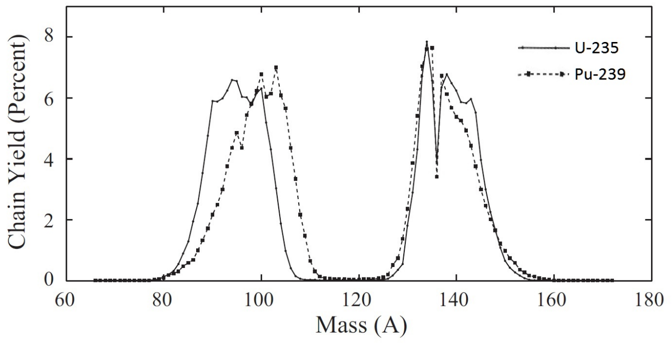


Figure 1: Thermal fission yield curves for Pu-239 (dashed line) and U-235 (solid line) [12, 17].

While passive measurements of special nuclear material have proven to be beneficial in certain situations, active interrogation methods such as thermal neutron induced fission followed by delayed gamma-ray detection have the ability to improve the detection and characterization of special nuclear material [35]. Traditionally, delayed gamma-ray detection has relied on peak identification and nuclear data to quantify fissile material. This process results in high uncertainties. Novel methods, such as temporal gamma-ray spectrometry, have been researched and suggest a reduction in the uncertainty associated with these traditional methods [64]

Williford's research in temporal gamma-ray spectrometry used the time-dependent decay characteristics of fission fragments to measure fissile material with improved accuracy and precision over traditional methods. Following thermal neutron induced fission, Williford evaluated the time and energy data of the resultant high energy beta-delayed gamma rays ($\geq$ 3 MeV). Many of these well-developed peaks were composed of gammas from more than one fission product. Each of these fission fragment decay products had an individual half-life, yield and potential for ingrowth; therefore, they could be evaluated temporally. Williford looked not only at key peaks but at how those peaks changed as a function of time resulting in the identification and quantification of fissile material based on the temporal characteristics of these peaks.

When performing active interrogation of SNM, focus is typically placed on identifying peaks above 3 MeV as they provide a unique signature for fissile material and are easily discerned from background. The delayed gamma-ray spectrum below

3 MeV is quite complex making it difficult to identify peaks for characterizing SNM. However, lower energy delayed gamma emissions are much more abundant.

Utilizing the temporal gamma-ray spectroscopy method developed by Williford, Curtis employed data mining techniques to identify channels which displayed differences in temporal behavior between Pu-239 and U-235 below 3 MeV. Spearman's correlation coefficient, a non-parametric, descriptive statistical method, was used to compare a training set of templates displaying radioactive decay behavior to channels displaying a pattern of counts over a given time interval. Spearman's correlation coefficient provided a statistical summary describing the underlying mechanism of radioactive decay for each channel. Computing the difference in correlation coefficients for each channel allowed for the discovery of differences between fissile material. The proof-of-concept work carried out by Curtis was promising for two reasons: 1) detector efficiency calibration was unnecessary as the temporal behavior was evaluated rather than known channels corresponding to specific energies and 2) additional regions of interest throughout the spectrum were identified and may be evaluated using Williford's method to further improve the accuracy and precision of quantifying mass content of fissile material.

1.2 Goal and Objectives

The goal of this research was to develop a novel machine learning process for the classification and potentially more accurate and precise quantification of fissile material content without the need for detector efficiency calibration or prior knowledge of isotopic peak energy information. Several objectives needed to be met to achieve this goal.

1.) Construct a fissile material feature set for input into machine learning classifiers that does not require prior knowledge of specific peak energies.

2.) Determine which classifier and feature set, out of those investigated, identifies the most accurate classification of fissile material.

3.) Identify regions of interest for determining the mass content of fissile material.

2 Literature Review

2.1 Nuclear Safeguards

In 1968, the Nuclear Non-Proliferation Treaty (NNPT) was introduced and since then, 190 countries have signed. Under the IAEA, a safeguards system was established by the NNPT with the following goals: to prevent the spread of nuclear weapons and weapons technology, to promote the peaceful use of nuclear energy and to further disarmament [28]. To meet these goals, the nuclear industry has focused on proper management and accountability of nuclear materials, ensuring that the diversion of special nuclear material is detected in a timely manner, and deterring proliferation activities by making it known that detection is a possibility [47]. Further, Physical Inventory Verification (PIV) tasks have been carried out by the IAEA to verify declared nuclear material inventories [47, 49]. Nondestructive assay techniques are commonly used for the management and accountability of nuclear materials. These methods have been in use for decades and have become an integral part of nuclear safeguards.

Nondestructive assay techniques are categorized as either passive or active depending on whether they are used to measure the spontaneous or induced radiation emitted by nuclear materials [49]. Destructive analysis techniques require sampling and chemical analysis of the source material; whereas nondestructive assay techniques do not require the source material to be altered, physically or chemically. Nondestructive assay techniques are potentially less time consuming and more cost effective than destructive analysis methods; however, they may be less accurate than chemical analysis [49]. To date, much research into the improvement and understanding of nondestructive assay techniques has been performed and the accuracy and precision of measurements has increased.

While the development of nondestructive assay methods occurred early on; it wasn't until the mid-sixties that the research and use of NDA techniques began to take off in the nuclear industry [21]. During this time the growth of the nuclear industry was at its highest and the need for more accurate measurements of nuclear materials for accountability and management was evident. Research into the use of passive and active nondestructive assay methods was performed at many of the national laboratories. Two of the main programs for active nondestructive assay were initiated under the Office of Safeguards and Material Management of the U.S. Atomic Energy Commission. In 1966, Los Alamos Scientific Laboratory researched using 14-MeV neutrons as a radiation source to perform active nondestructive assay [21]. In 1967, research on the use of high-energy gamma rays generated by a linear accelerator for active nondestructive assay was performed at the Linac Department of Gulf General Atomic [21]. Common applications of nondestructive assay include: accounting of nuclear material, searching for nuclear material as a safeguard against theft, verification of prior measurements and the quality control of nuclear materials [21]. Passive and active nondestructive assay techniques will be explored further in the next sections.

2.1.1 Passive Interrogation

Passive interrogation involves the detection of gamma rays emitted intrinsically from nuclear materials. Uranium and plutonium isotopes as well as their decay products emit alpha and/or beta radiation along with their associated gamma rays as a part of their natural decay process. In addition, both uranium and plutonium isotopes can spontaneously fission producing fission fragments along with their respective neutrons and gamma rays. The energies of the gamma rays emitted are characteristic of the nuclide from which they originate and detection of these gamma rays can be used to differentiate nuclear material from that of background [49].

Gamma rays emitted naturally from U-235 and Pu-239 are typically low in energy, around the 500 keV range. For U-235, passive measurements include those at or below 186 keV and for Pu-239, those at or below 413 keV [6]. In most situations where uranium and plutonium need to be distinguished from one another, these lower energies are difficult to detect due high background levels or to shielding. In addition, emissions from spontaneous fission are much lower than those from induced fission. Implementing active interrogation methods can overcome these challenges and results in a more accurate measurement of fissile material.

Passive interrogation is typically used to measure scrap, waste and residue; however, passive interrogation methods have their limits [21]. For instance, the emission of lower energy gamma rays from nuclear material such as uranium and plutonium are easily shielded making passive interrogation nearly impossible. Or, in the case of measuring plutonium and uranium in an spent nuclear fuel (SNF) assembly, a complex overlapping background exists from the gamma rays emitted by the buildup of fission products during irradiation and those emitted by uranium and plutonium; making measuring these materials quite challenging using passive methods [49].

2.1.2 Active Interrogation

The primary purpose for researching active NDA methods was for nuclear safeguards; however, improved processes for quality control also resulted ([22]. Active interrogation involves the use of penetrating radiation, typically a photon or neutron source, to induce fission on fissile material resulting in fission fragments and their associated neutrons and gamma rays. Typically, a neutron source results in the prompt fission production of two to three neutrons and about eight gamma rays [21]. Following irradiation, about six or seven delayed gamma rays are emitted and about 0.01 to 0.02 delayed neutrons per fission [21].

Active interrogation techniques may be used to detect either the prompt or delayed emissions. While neutron induced fission results in strong prompt gamma-ray signatures, detecting the prompt emissions can be challenging due to irradiation and detection occurring simultaneously. Doing so requires implementing one of many methods to accurately differentiate the prompt fission emissions from the interrogating source [21]. Delayed emission detection on the other hand measures the decay of fission products after irradiation has ended. Inducing fission on U-235 and Pu-239 results in fission products that decay via beta decay resulting in the emission of subsequent gamma rays. Delayed gamma rays are about 500 to 700 times more intense than delayed neutrons from fission and therefore the counting of delayed gamma rays is a more sensitive method [21].

Many authors have reported the use of active interrogation methods for detection and quantification of shielded SNM [43, 56, 23, 40]. In addition, significant research has been performed using active interrogation for the management and accountability of Pu-239 and U-235 in spent fuel [59, 51, 64]. Various active interrogation methods are being explored; however, one of the more common techniques utilizes the delayed gamma method.

2.1.3 Delayed Gamma-Ray Detection

Early research showed that differentiating U-235 from Pu-239 could be performed by using active interrogation and evaluating gamma-ray energy peaks greater than 800 keV by looking at their peak intensity ratios and comparing these values to calculated theoretical values [6]. The intensity ratios were developed from the multiple isotopes comprising the chosen peaks and were independent of the number of fissions induced. While the authors confirmed that their method could be used to identify fissile material, they stated that a large number of intensity ratios needed to be evaluated to ensure accurate identification.

Firestone et al. extended this research and looked at the prompt and delayed gamma-rays in the 3-4 MeV range to determine the concentration and enrichment of U-235. They found that the high-energy gamma rays above 3 MeV emitted from the decay of short-lived fission products provide a unique signature of fissile material and can be used to quantify uranium. Further, by using ratios of the gamma-ray intensities, the concentration of fission isotopes could be determined [18].

Both of these approaches relied on nuclear data which carries fairly high uncertainty, especially for short-lived fission products. However, this research led to the identification of specific gamma-ray line pairs which dominated the gamma-ray spectrum from 1 minute to 14 hours following fission [39]. Marrs et al. focused on identifying gamma-ray line pairs that were insensitive to neutron energy. They were able to identify line pairs and groups with intensity ratios that were different for U-235 and Pu-239 [39]. From here, research of a novel temporal gamma-ray spectroscopy method was explored empirically by Chivers [12].

2.1.4 Temporal Gamma-Ray Spectrometry

In an effort to reduce the high uncertainties associated with traditional methods, Chivers et al. suggested to solely look at the temporal response from the beta-delayed gamma emissions of fissile materials. To do this, the peaks in the spectrum needed to be normalized so that only temporal differences were present. This was done by taking the counts in a 3 keV energy bin over a 2.5 second time frame and normalizing this by the total number of counts over a 10 second time frame (the entire counting interval) in the same 3 keV bin [12]. Using these ratios, characterization of fissile material could be performed. Further, material could be quantified by taking the difference of these ratios. Chivers' work was promising but had not been carried out experimentally.

Building on their research, Williford exploited the time-dependent decay characteristics of fission fragments and evaluated not only the peaks but he also looked at how they changed as a function of time. To do this, Williford fission-normalized the measurements by using peak responses at a reference time and creating a continuous temporal spectrum. Several factors, such as the number of radioisotopes composing the peak, their respective half-life as well as whether any of the radioisotopes experience ingrowth, influence the temporal response of peaks [64]. The temporal method takes advantage of using the more pronounced, well-developed peaks composed of multiple fission fragments thus minimizing systemic biases. Through the evaluation of these high energy peaks ($\geq$ 3 MeV) within the temporal spectrum, Williford showed that he could quantify fissile material with a 2 % uncertainty; a significant improvement from the existing 10 % uncertainty for SNF assemblies using traditional delayed gamma methods [64, 11].

2.1.5 Discovering Meaningful Patterns Below 3 MeV Using Spearman's Correlation Coefficient

Often focus is on higher energy delayed gamma rays that are easier to discern from the very complex spectrum of fission product delayed gamma emissions below 3 MeV. Discovering important relationships within complex data sets is challenging. Data needs to be reduced to key features to obtain accurate, efficient results when implementing machine learning models. Curtis was able to show that evaluating the spectrum below 3 MeV using temporal methods paired with data mining leads to a data-driven process that can provide additional information for fissile material classification and quantification.

Through the measure of association between two variables, less relevant attributes may be overlooked effectively filtering the data and highlighting features that are most significant. Constructing a new feature from count frequency data reduces dimensionality. Given the unique time-dependent decay characteristics of fission products, known patterns of decay and ingrowth are expected throughout the spectrum. This a priori knowledge was used to generate a training set of data which used patterns of decay and ingrowth to evaluate the fission product spectra of U-235 and Pu-239. Specifically, gamma-ray event data was transformed from vectors of count values per time of occurrence to radioactive decay patterns and correlated with decay and ingrowth templates using Spearman's correlation coefficient. Employing these methods, Curtis discovered meaningful patterns within a large complex spectrum of gamma-ray emissions.

2.2 Data Mining

Data mining is performed to support discovery of meaningful patterns within data. A combination of tools and methods from machine learning, statistics and other data analysis technologies are utilized in the process [26]. Data preparation is an important step in data mining. Raw data needs to be pre-processed to effectively reduce the dimensionality of a data set or transform the data prior to implementing machine learning algorithms. For example, statistical methods may be used to transform the data, developing new features, also referred to as attributes, that may be used as input for machine learning algorithms. These new features, may provide a better representation of the data, enabling the user to use fewer parameters in their analysis. Further, statistical methods such as attribute selection and visualization are important data analysis techniques used during data preparation [65]. Attribute selection is particularly important. Ideally, the user would like to reduce the number of features used to train a given classifier without losing discriminatory power for class identification [58].

Further, in data pre-processing the user analyzes data to extract structural patterns that may be used for nontrivial prediction on new data [65]. Determining whether a pattern is significant requires understanding the underlying mechanism being described and should be carried out by experts with the required knowledge

[26]. Often, internal models are constructed during data preparation prior to input into machine learning models. Data preparation and model development are iterative in nature. Many insights learned from the developed model lead to new ways to pre-process the data resulting in a more accurate final model.

2.2.1 Descriptive Statistics

Measures of correlation are descriptive statistical methods used to show the extent of relationship between two variables. Correlation measures are not used for inferential purposes; however, once a measure of correlation has been calculated, inferential statistics may be used to evaluate a hypothesis concerning the correlation [55]. In addition, a correlation between two variables does not imply that a change in one variable causes a change in the other. If two variables have a strong correlation, one can not conclude that one variable caused a change in the other variable; rather it can be noted that there is an association present between the two variables [41]. Two of the most common measures of correlation are the Pearson product-moment correlation coefficient (PPMCC) and the Spearman's rank-order correlation coefficient [41]. The Pearson product-moment correlation coefficient was developed by Karl Pearson in 1900 and is a measure of linear correlation between two values. Spearman's rank-order correlation coefficient was developed in 1904 by Charles Spearman and is a special case of the PPMCC.

2.2.2 The Pearson Product-Moment Correlation Coefficient

The Pearson Product-Moment Correlation Coefficient (PPMCC) is a bivariate, parametric measure of association/correlation. This method is used with interval/ratio data to evaluate whether a linear relationship exists between two variables.

The correlation coefficient computed by PPMCC is represented by the greek letter rho, ρ. Rho can have a value between 1 and -1. As the absolute value of ρ approaches 1, the strength of the linear relationship between the two variables increases. As the absolute value of ρ approaches 0, the strength of the linear relationship between the two variables decreases. The sign of ρ indicates the direction of the linear relationship where, a positive sign indicates a direct linear relationship and a negative sign indicates a indirect (inverse) linear relationship [55]. Rho is calculated by using the following formula,

$$\rho = \frac{\sum XY - \frac{(\sum X)(\sum Y)}{n}}{\sqrt{\left[\sum X^2 - \frac{(\sum X)^2}{n}\right]\left[\sum Y^2 - \frac{(\sum Y)^2}{n}\right]}} \tag{1}$$

where, X and Y represent the values of the two data sets, and n represents the number of measurements in the data set.

The PPMCC assumes a linear relationship best represents the data. The value of ρ may not indicate the extent of the relationship between the two variables if a curvilinear relationship better describes the data [55]. Spearman's rank-order correlation coefficient, used for curvilinear relationships, will be discussed next.

2.2.3 Spearman's Rank-Order Correlation Coefficient

Spearman's rank-order correlation coefficient is a nonparametric, descriptive statistical method used to investigate the strength of association between non-linear sets of rank-order data. Given that the Spearman's method assumes nothing about the distribution and is used for non-linear associations, it is an ideal method for use in this research. Spearman's correlation coefficient is a special case of the Pearson product-moment correlation coefficient therefore it is also represented by the greek symbol rho, ρ, and is referred to as the rho score or Spearman's rho. Spearman's rho will be denoted as ρ_s. To calculate Spearman's correlation coefficient, the data sets are ranked separately in either ascending or descending order as long as the ranking is performed the same for both data sets. Table 1 shows an example of how two data sets would be ranked.

Table 1: Example ranking of two data sets. The first ranked data set is channel 231 from a single Pu irradiation cycle and is to be compared with the ranked data of template 1, the second data set.

Channel 231 Pu	**Rank Pu**	**Template 1**	**Rank Template 1**
264	10	659	1
267	9	435	2
283	7	287	3
275	8	190	4
285	6	125	5
295	4	83	6
287	5	54	7
306	2	36	8
301	3	24	9
313	1	16	10

The difference in ranks is calculated and the resulting value squared. The following formula is used to calculate Spearman's rho score for untied ranks:

$$\rho_s = 1 - \frac{6 \sum d_i^2}{n(n^2) - 1} \tag{2}$$

where, n represents the number of measurements in the data set and d_i represents the difference in ranks. While Equation 2 is for untied ranks, it has been noted in the literature that if only a few ties exist using Equation 2 is still quite reliable [8].

In R, a tie-corrected formula is used to compute Spearman's rho. Equations 3-6 lead to the final formula utilized in R seen here as Equation 7:

$$T_x = \sum_{i=1}^{s} (t^3_{(x)} - t_{i_{(x)}}) \tag{3}$$

$$T_y = \sum_{i=1}^{s} (t^3_{(y)} - t_{i_{(y)}}) \tag{4}$$

The notations in Equations 3 and 4, indicate that for each variable, the number of tied ranks is subtracted from the number of tied ranks cubed and then summed.

$$\sum x^2 = \frac{n^3 - n - T_x}{12} \tag{5}$$

$$\sum y^2 = \frac{n^3 - n - T_y}{12} \tag{6}$$

where, n in Equations 5 and 6 represent the number of observations in the data set.

$$\rho_{s_c} = \frac{\sum x^2 + \sum y^2 - \sum d^2}{2\sqrt{\sum x^2 \sum y^2}} \tag{7}$$

To verify that the same ρ_s score will be computed when using the tie-corrected formula on data that are untied, let's look at a simple example where the computed Spearman's rho is a value of one. Consider the sum of the difference in ranks to be zero and the number of observations in the data set to be ten. Substituting these values into Equation 2 computes the following ρ_s score:

$$\rho_s = 1 - \frac{6(0)}{10(10^2) - 1} = 1 \tag{8}$$

Now lets take a look at the tie-corrected formula used in R. Equations 3 and 4 result in a value of zero since there are no ties present in the data set. Substituting in ten for n and Equation 3 into Equation 5 we get,

$$\sum x^2 = \frac{10^3 - 10 - 0}{12} = 82.5 \tag{9}$$

Substituting in ten for n and Equation 4 into Equation 6 we get,

$$\sum y^2 = \frac{10^3 - 10 - 0}{12} = 82.5 \tag{10}$$

Finally, substituting in zero for the difference in ranks and Equations 5 and 6 into equation 7 we get,

$$\rho_{s_c} = \frac{82.5 + 82.5 - 0}{2\sqrt{(82.5)x(82.5)}} = 1 \tag{11}$$

Equation 11 results in a Spearman's rho value of 1, the same result seen using Equation 8.

As with PPMCC, the Spearman's rho score produces a value between 1 and -1 indicating either a strong positive or a strong negative association between the paired data sets. As mentioned previously, an association between data sets does not imply causation. The Spearman rank-order correlation coefficient method is a descriptive statistical method, and for our purposes is used for pattern recognition, not to predict behavior. The Spearman's method is ideal for data sets that do not have a normal distribution, have a potential for outliers and have non-linear relationships [55]. Radioactive decay is a stochastic, non-linear process; making Spearman's rank-order correlation coefficient an appropriate method to implement into a machine learning process for the identification of the differences in the spectra of plutonium and uranium.

2.3 Machine Learning

Machine learning is a subfield of computer science that originally was used for artificial intelligence. Through the use of algorithmic methods, machine learning is able to address complex data tasks. Breiman has written extensively about the two cultures of statistical modeling. Scientists either assume that data was created from a particular model or they assume an unknown model and apply algorithmic methods to the data to reach conclusions [10]. The latter ensures a data driven process and supports an innovative way of solving complex data problems.

King states, "...the production of humanly comprehensible rules from a machine learning system allows the rules to be checked for consistency with existing knowledge and opens the possibility that rules may provide fresh insight." [32] King et al. used machine learning techniques to derive a set of rules that could be used to predict relative activities for two drugs. Decision trees have been used to detect breast masses [42] while neural networks (NN) have been used for general spectral analysis in the fields of chemistry [34, 66] as well as for low resolution x-ray fluorescence spectra analysis [37].

Machine learning techniques typically fall into one of two categories; supervised or unsupervised methods. In supervised machine learning, the use of background knowledge is important. A set of known inputs, class labeled data, is used as a training set. These inputs are compared with the desired outputs given by a "teacher" with the goal of learning rules that map inputs to outputs. Unsupervised learning on the other hand, uses unlabeled data for training [25]. This research utilizes supervised learning methods to classify special nuclear material.

2.3.1 Machine Learning Applied to Gamma-Ray Spectroscopy

Machine learning techniques are used across disciplines and the area of radiation detection is no exception. For example, neural networks, support vector machines, and Bayesian classifiers have been used to analyze and interpret gamma-ray spectrometry data [67, 16, 29, 63, 27, 61, 57, 1]. Application of machine learning techniques, specifically for gamma-ray spectroscopy analysis, have been most prevalent in the areas of Homeland security, non-proliferation, and environmental assessment and restoration.

Olmos et al. used a neural network (NN) to identify a single isotope of interest present in a mixture of elements [45]. Their method evaluates the entire spectrum rather than individual peaks by comparing its shape with patterns located in a reference spectra produced by NaI(Tl) and Ge(Li) detectors. A set of training isotopes were used for training the NN. However, Olmos et al. state that their method is most useful when precise and accurate activity determination is not necessary [46]. In addition to their work in isotope identification using NN, Olmos et al. analyzed gamma-ray spectra using a linear associative memory neural network algorithm to successfully address drift problems in nuclear instrumentation [44].

Kangas et al. used artificial neural networks for assigning quality factors to alpha spectra used for air quality monitoring, specifically for the detection of plutonium contamination in nuclear processing and storage facilities [30]. Quality factors were assigned to spectra by experts and used for training the NN. Once trained, the NN was used to provide real-time spectra inference providing early warning based on spectral quality. Similar to Olmos et al., their method relied on using the shape of the spectrum to train the NN.

Sullivan et al. used a Bayesian approach to address poor isotope identification using handheld radio-isotope identifiers (RIIDs) for Homeland security and nuclear safety applications [57]. Their method uses a spectral library containing peak energies and branching ratios combined with detector efficiency. A wavelet-based algorithm is used for peak identification followed by the implementation of a Naïve Bayes classifier to determine the probability of a particular isotopes presence. They accurately identified isotopes in unknown shielding scenarios despite appreciable calibration drift. However, their method relies on a priori knowledge of peak energies, branching ratios and requires efficiency calibration. Use of branching ratios contributes to uncertainty, an issue addressed by the use of temporal spectroscopy methods. Bayesian approaches have also been used for the analysis of radiation portal data [14], for application to probabilistic risk assessment of nuclear waste disposal [33] and for monitoring the movement of weak radiation sources (ie: tracking shipments) while discriminating "true" sources from "false" sources [13].

Data from portal monitors have also been analyzed using artificial neural networks (ANNs) [29]. Kangas et al. employed an ANN to successfully discriminate normally occurring radioactive materials (NORM) from special nuclear material effectively reducing the probability of false alarms. Further, Sharma et al. implemented machine learning techniques to reduce false alarm rates when using gamma-ray spectrometers for the identification of persons concealing radioactive materials [54].

Vigneron et al. have used neural networks to analyze gamma spectra from uranium-enrichment measurements for the determination of U-235 and U-238 content [62]. Relying on passive detection, focus was placed on analyzing the $K_\alpha X$ region of the spectrum using infinitely thick samples to avoid matrix effects and calibration procedures. Further, Aitkenhead et al. evaluated the spectra of shielded plutonium using ANNs to detect the presence or absence of plutonium, estimate Pu-239 content as well as distinguish material age of shielded plutonium [2]. However, they were unable to achieve accurate material age estimation. In addition, Dragovic et al. used an ANN to model uncertainty in gamma-ray spectrometry [16]. Once trained, their NN could predict measurement times needed to obtain a particular level of statistical accuracy in the analysis of environmental samples.

Support vector machine algorithms have also been used for the classification of waste drums using NaI (Tl) detectors [27] and Varley et al. have utilized principal component analysis and neural network algorithms to estimate the activity and depth of Ra-226 contamination using spectra from Sodium Iodide and Lanthium Bromide detectors[61, 60]. Lastly, Keller et al. have looked into the application of ANNs for real-time data processing for environmental restoration and waste management at the Hanford site [31].

Much of the work to date has focused on evaluating the passive spectra from mid and low resolution detectors. However, some research in the application of machine learning algorithms for classification of radioisotopes using HPGe detectors has been conducted. For example, an ANN algorithm used for pattern recognition has been employed for gamma-ray spectral analysis from Ge detectors [67]. Typically, ANNs are rarely used for gamma-ray spectrometry analysis due to the large spectral size [67]. To overcome this, Yoshida et al. used a peak search procedure to reduce gamma-ray spectra to peak energy data to be used as input into a NN [67]. They created a subset of data consisting of peak energy information as the input data and used patterns of emitted gamma-ray energies from individual nuclides for training. They were able to successfully use ANN for pattern recognition and identify radioisotopes within the gamma-ray spectra [67]. However, they were unable to process the raw spectral data and were limited to a data set consisting of gamma-ray peak energies. Neural networks have also been used for time-of-flight discrimination between neutrons and gamma-rays in HPGe detectors, correctly classifying gamma-ray and neutron events [3].

In addition, Pilato et al. successively used NNs for quantitative analysis of gamma spectra produced by a HPGe detector [48]. Their method overcomes the traditional obstacles associated with determining the activity of radionuclides such as overlapping peaks which software deconvolution algorithms have difficulty with. Relying on peak deconvolution results in a solution that is dependent on the complexity of the spectra. They use “global information contained in the spectrum” rather than separating peaks by deconvolution [48]. In addition, their method limits the need for user intervention and expert knowledge once the NN has been trained.

Through the review of the literature, it is evident that the application of machine learning techniques to gamma spectroscopy continues to be of great interest. The limitations that many of these authors encountered will be overcome with the novel research presented here. This research uses active interrogation methods rather than passive methods used by many of these researchers. Further, this research does not require detector efficiency calibration, nuclear data, nor a library of specific peak energies a priori. Lastly, while the entire spectrum will be analyzed, its shape will not be used in the training of the machine learning algorithms. Rather, the global spectral information will drive the learning process enabling the user to identify areas of interest for fissile mass content evaluation that to date have been difficult to analyze.

2.4 Waikato Environment Knowledge Analysis

Waikato Environment Knowledge Analysis (WEKA) was developed at the University of Waikato in New Zealand and is open source software written in JAVA. The WEKA workbench provides a convenient method for implementing machine learning algorithms. WEKA resources are abundant and the graphical user interface (GUI) enables the user to visualize the development of several algorithms such as the decision tree and Bayesian network architectures [24]. Further, with the many tools available for data pre-processing, WEKA provides the user a way to efficiently tackle any data mining problem. As such, WEKA is a widely used tool across disciplines for data mining [24].

2.5 Classifiers

The process of developing a model which predicts class labels for unlabeled data is known as classification [65]. Giacometti et al. state, "A classifier is a global model which not only describes the whole training database achieving some level of accuracy, but also is used to predict the class label for data objects that are unlabeled." [20] Classifiers are constructed through the analysis of training data, data for which class labels are known [20]. The goal is to assign an input vector X to one of K discrete classes, denoted C_k, where k=1,....,K. Each input is assigned to only a single class; therefore, the classes are said to be disjoint [7].

To ensure a given classifier achieves the desired level accuracy and does not overfit the data, two phases are employed; train and test. Data need to be partitioned to perform the two phases. There are many strategies for partitioning the data. One technique, the holdout method, involves assigning a certain amount of data to the testing phase and using the rest of the data for training. Typically, one-third of the data are assigned to the testing phase and the remaining two-thirds are assigned to the training phase [65]. However, bias may occur when samples of the data chosen for the testing phase are not representative of the data chosen for the training phase. It is important to mitigate any potential bias which may occur during this process.

Cross-validation is often used for reducing bias. K-Fold cross validation involves selecting a fixed number of folds, partitions, of the data. Each fold is used for training and the remainder used for testing [65]. As and example, if a user were to perform 10-fold cross-validation, data would be randomly divided into ten partitions. Each partition is used for testing and the remaining nine-tenths are used for training. The error rate is calculated on the data set used for testing. This learning is performed ten times producing ten error estimates. The average error estimate is used to evaluate the overall error of the classifier.

Once the classifier has been trained and tested, the model is used to classify an unknown, unlabeled data set for validation. There exist numerous algorithms for addressing classification problems. In the following sections we outline three that have been chosen for this project; decision tree, neural network, and Naïve Bayes architectures. Implementation of and evaluation of all three architectures will yield a thorough evaluation of the most accurate, precise and efficient model of those selected to use for the classification of Pu-239 and U-235.

2.5.1 Decision Tree

Perhaps the simplest of all classifiers, decision trees are models which follow a tree-like structure. Classification is performed through recursive and stepwise partitioning of a target field, which contains all instances of interest, based on unique features associated with one or more attributes [15]. Attributes are often referred to as input fields. Input fields are chosen based on their ability to explain the variability between the target values. Determining the "best" input field is somewhat subjective as the process can be both user and computationally driven. The inherent top-down structure of decision trees allow the user to visualize the tree and choose strong inputs based on known theoretical connections of model components [15]. However, decision trees also enable a variety of computational approaches to be implemented. For example, the C4.5 (J48) algorithm located in the WEKA, offers various pruning algorithms for selecting attributes to split on. The default algorithm uses information gain, gain ratio and entropy computations in its selection process [42]. Often, inputs are selected based on combining expert knowledge with computational outputs of the partitioning strength provided by the software.

Nodes represent tests on a given attribute/input field, the branches describe the outcome of the tests, and the tree leaves represent the classes or class probabilities [65]. The top most node is termed the root node and contains all of the information regarding the target field. Instances begin at the root node and travel down the tree based on the attribute values at each node. When a leaf is reached, a class is assigned based on the classification of the respective leaf. Nodes may be nominal, categorical or interval measurements. When the root node contains categorical values, the tree is referred to as a classification tree. Decision trees are known for being powerful yet easy to implement and interpret [15].

Let's take a look at a simple example and walk through how the classification process works. In the following example we are going to classify whether we should head to the beach or not. The rows of the table display the instances for the data set. The columns represent the various attributes we can evaluate to determine whether or not we should visit the beach. The final column is the outcome (label) associated with each instance.

Table 2: Beach data.

Weather Outlook	Temperature	Windy	Beach
Sunny	50	Yes	No
Overcast	75	No	Yes
Sunny	69	No	Yes
Overcast	50	Yes	No
Sunny	85	Yes	Yes
Sunny	57	No	Yes
Overcast	62	No	No
Sunny	70	Yes	No

Figure 2 shows a decision tree developed based on using the temperature attribute as the root node. The subsequent nodes are based on the remaining attributes, outlook and windy. Evaluation at each node leads to the classification of beach or no beach. The tree displays a set of rules which accurately classify each of the instances in Table 2.

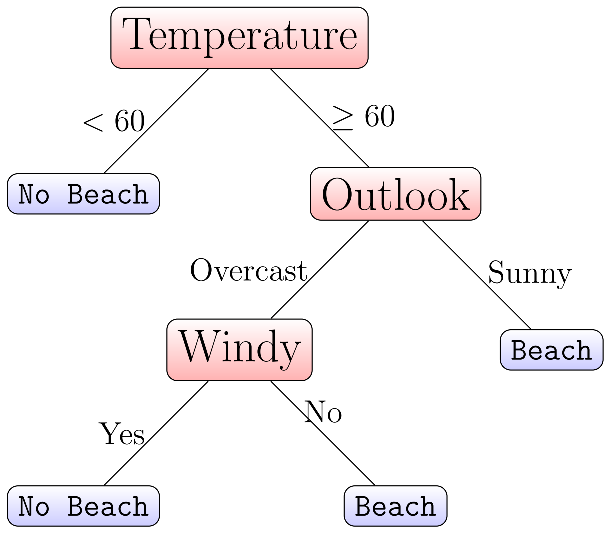


Figure 2: Decision tree.

2.5.2 Random Forest

Breiman proposed the random forest algorithm in 2001 [19]. Several randomized decision trees are grown, without pruning, using randomly selected inputs or combinations of inputs at each node [9]. Given a particular instance, each tree provides a class prediction of equal weight. A majority vote from all of the decision trees generated results in the final class prediction. This method has led to improved classification accuracy. Further, due to the Strong Law of Large Numbers, random forests always converge and therefore do not overfit [9]. Figures 3 and 4 are visual representations of how the random forest algorithm works.

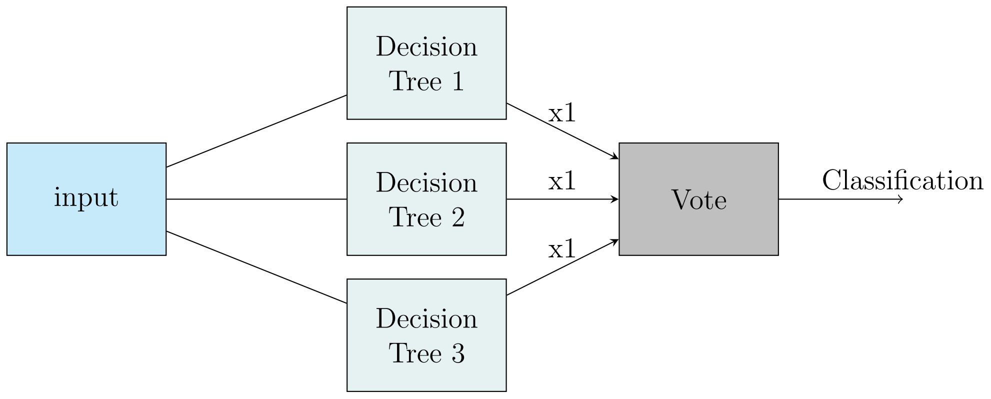


Figure 3: Visualization of the random forest architecture showing that equal weights are given to each trees vote. [36].

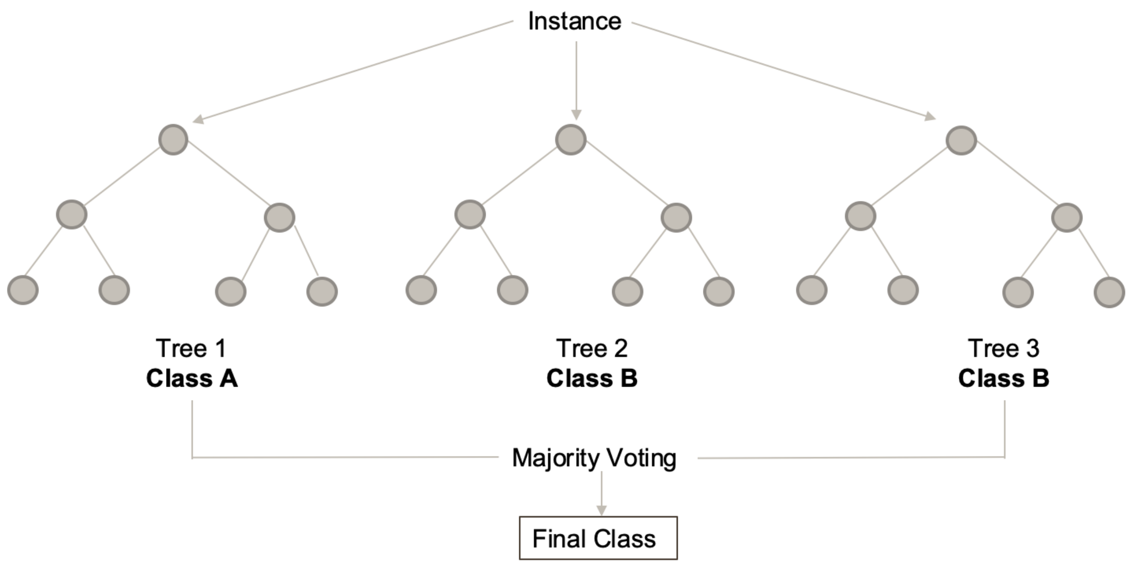


Figure 4: Three individual decision trees, each with their own vote for classification. The majority vote results in the final classification of a given instance.

2.5.3 Neural Network

Predicting class labels can also be performed by using an algorithm to separate different classes by a hyperplane. Data which can be separated into two groups via a hyperplane are known to be linearly separable [65]. The perceptron learning rule is a simple algorithm which can determine a separating hyperplane. The equation for a hyperplane is

$$w_0a_0 + w_1a_1 + w_2a_2 + ... + w_ka_k = x \tag{12}$$

where $a_1, a_2, ...a_k$ are the attribute values and $w_1, w_2...w_k$ are the weights which define the hyperplane [65]. The attribute a_0 is assumed to always have the value of 1 to represent the bias. If the sum is greater than 1, the first class is predicted otherwise the second class is predicted. The goal is to find weights which define a hyperplane that accurately classifies the data. Determining the weights is an iterative process and all instances are evaluated each iteration. To begin, all weights are defined to be zero. Weights are adjusted by determining if an instance has been correctly classified. If it has not, and the instance belongs to the first class, its attribute values are added to the weight vector. Otherwise, they are subtracted from the weight vector. Weights are adjusted until all the training data has been classified correctly. If the data are linearly separable, the hyperplane will converge. The hyperplane is known as a perceptron and is graphically represented in Figure 5.

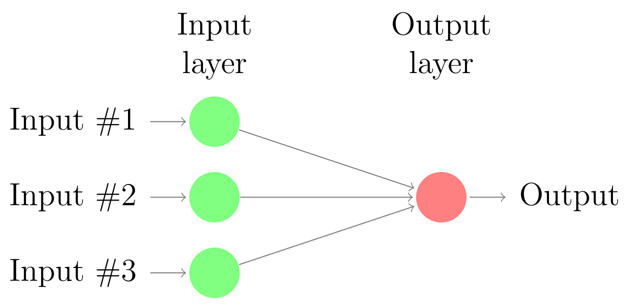


Figure 5: Simple perceptron.

As seen in Figure 5, the structure appears as a network of neurons. There are nodes (neurons) located in each layer and weights (seen as lines) representing the connection pathways between the neurons. In fact, this model was termed 'neural network' for its similarity to the biological neural process we are familiar with. The nodes in the input layer correspond to the attributes of the data set and the output layer defines the class. If the data are not linearly separable, this method can still be implemented. However, it is represented by a nonlinear model called a multilayer neural network [65]. An example of a multilayer perceptron (MLP) model is seen in Figure 6.

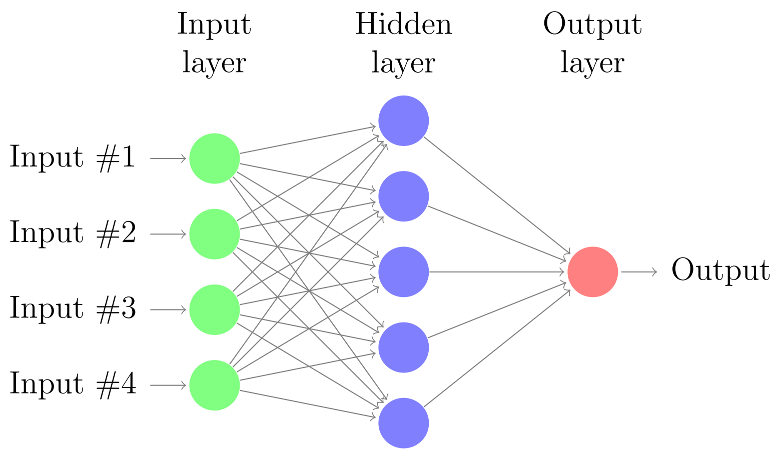


Figure 6: Multilayer perceptron.

A MLP includes one or more hidden layers. Hidden layers are responsible for applying an activation function or combination of functions to the data and are often not visualized by the user and as such are known as ‘hidden’. The number of hidden layers and number of nodes within each layer determine the network structure. Learning the structure of a neural network is often based on experimentation combined with some level of expert knowledge [65] and is driven by the desired accuracy. Determining the weights for a MLP is a bit more complex and requires a method known as backpropagation to be implemented if there is more than one hidden layer.

The weight values connecting the input layer to the hidden layer are modified based on the strength of individual unit contributions from the hidden layer to the output layer [65]. The gradient descent algorithm is used to modify the weights. The gradient descent procedure is an iterative optimization algorithm based on taking derivatives [65]. The sigmoid function and the squared-error loss function are most commonly used. The sigmoid function converts the weighted sums from the inputs into a range of values between 0 and 1. Figure 7 displays the sigmoid function, f(x), for a given input value x.

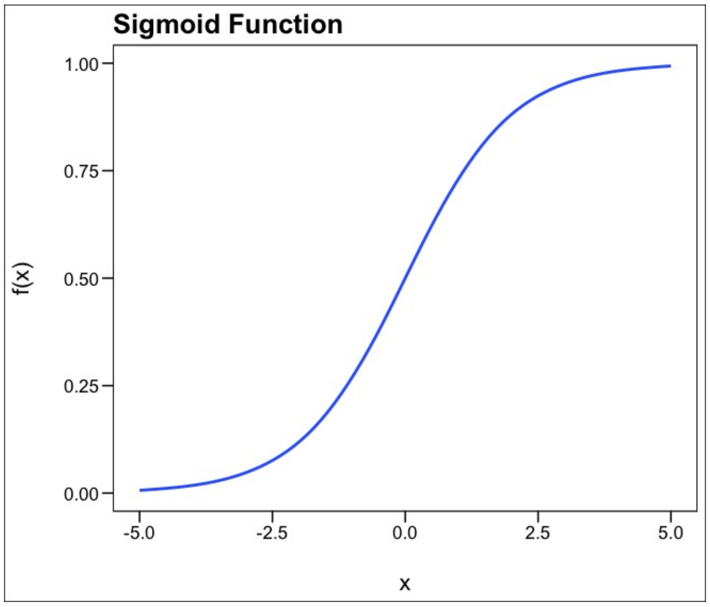


Figure 7: Sigmoid function, f(x) for a given input value x.

Equation 13 defines the sigmoid function seen in figure 7

$$f(x) = \frac{1}{1 + e^{-x}} \tag{13}$$

where x is the the hyperplane function defined in equation 12. Equation 14 defines the squared-error loss function

$$E = \frac{1}{2}(y - f(x))^2 \tag{14}$$

where $f(x)$ is the output prediction and y is the instance's class label [65]. Specifically, the gradient descent procedure utilizes information gained from taking the derivative of the error function to adjust the weights and minimize the overall error on the training data [65]. The value of the derivative is multiplied by the learning rate, a small constant. The result is then subtracted from the current weight value. This process is repeated until a minimum weight value is reached.

For a single perceptron, no hidden layer present, differentiating the error function with respect to a particular weight w_i yields

$$\frac{dE}{dw_i} = (f(x) - y)\frac{f(x)}{dw_i} \tag{15}$$

where $f(x)$ is the perceptron's output and x is the weighted sum of the inputs [65]. To compute $\frac{f(x)}{dw_i}$, we need to take the derivative of the sigmoid function, denoted as $f'(x)$, which is

$$\frac{df(x)}{dx} = f(x)(1 - f(x)) \tag{16}$$

However, we are interested in the derivative with respect to w_i rather than x. Noting that x is defined as

$$x = \sum_i w_i a_i \tag{17}$$

we can write the derivative of $f(x)$ with respect to w_i as

$$\frac{df(x)}{dw_i} = f^{'}(x)a_i \tag{18}$$

Substituting equation 18 into the derivative of the error function, equation 15, yields the expression required to calculate the change in weight, w_i and is seen here

$$\frac{dE}{dw_i} = (f(x) - y)f^{'}(x)a_i \tag{19}$$

Arriving at an expression which allows computation of the change of weights for networks with hidden layers is more complicated. Assuming there is one hidden layer, a_i from Equation 19 is replaced with the output of the i^{th} hidden unit. The result is

$$\frac{dE}{dw_i} = (f(x) - y)f^{'}(x)f(x_i) \tag{20}$$

The weights we are attempting to change are represented by w_{ij} and are those from the j^{th} input to the i^{th} hidden unit. As before, the corresponding derivatives need to be taken

$$\frac{dE}{dw_{ij}} = \frac{dE}{dx}\frac{dx}{dw_{ij}} = (f(x) - y)f^{'}(x)\frac{dx}{dw_{ij}} \tag{21}$$

To compute $\frac{dx}{dw_{ij}}$ recall that

$$x = \sum_i w_i f(x_i) \tag{22}$$

and

$$\frac{dx}{dw_{ij}} = w_i \frac{df(x_i)}{dw_{ij}} \tag{23}$$

Further,

$$x_i = \sum_j w_{ij} a_j \tag{24}$$

so

$$\frac{df(x_i)}{dw_{ij}} = f^{'}(x_i) \frac{dx_i}{dw_{ij}} = f^{'}(x_i) a_j \tag{25}$$

The final expression used to determine the change of weights, w_{ij} is

$$\frac{dE}{dw_{ij}} = (f(x) - y) f^{'}(x) w_i f^{'}(x_i) a_j \tag{26}$$

Networks such as those seen in Figures 5 and 6 are known as feed-forward networks, the most common of neural networks. Feed-forward networks process the data in one direction from the input layer to the output layer, known as forward propagation. In a single perceptron network, there are no connections between input neurons; all neurons are directly connected to the output neuron. For a MLP with one hidden layer, forward propagation continues through additional layers and analyzes a combination of inputs. However, if a multilayer network consists of more than one hidden layer the above derivation for an MLP is repeated using the weights correlated with the inputs of the previous layer and the error is propagated from the output unit backwards through the network to the first layer. The generic gradient descent procedure used for a single hidden layer is then termed backpropagation [65].

2.5.4 Naïve Bayes

The Naïve Bayes classifier is based off of simple probabilistic modeling. Specifically, Bayes rule of conditional probability. Baye's rule is:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{27}$$

where $P(A|B)$ is the liklihood of event A occurring given event B is true, $P(B|A)$ is the liklihood of event B occurring given event A is true, and $P(A)$ and $P(B)$ are the probabilities of observing events A and B independently. As a general example, given three pieces of evidence and assuming they are independent and each have equal contribution to the outcome given their class, we obtain their combined probability by multiplying their individual probabilities as seen here

$$P(yes|E) = \frac{P(E_1|yes) * P(E_2|yes) * P(E_3|yes) * P(yes)}{P(E)} \tag{28}$$

Let's take a look at a more specific example, the beach data we used for our decision tree. Table 3 shows the counts and probabilities for the various attributes.

Table 3: Beach data with counts and probabilities.

Weather Outlook			**Temperature**			**Windy**			**Beach**	
	Yes	**No**		**Yes**	**No**		**Yes**	**No**	**Yes**	**No**
Sunny	3	2	$>$ **60**	3	2	**True**	1	3	5	3
Overcast	1	2	$<$ **60**	1	2	**False**	3	1		
Sunny	3/8	2/8	$>$ **60**	3/8	2/8	**True**	1/8	3/8	5/8	3/8
Overcast	1/8	2/8	$<$ **60**	1/8	2/8	**False**	3/8	1/8		

Given a new instance, one that is unlabeled, we would like to predict whether we should head to the beach. Table 4 describes the attributes of our new instance.

Table 4: New, unlabeled instance.

Weather Outlook	**Temperature**	**Windy**	**Beach**
Overcast	75	No	?

We treat each attribute equally and independently, computing the likelihood that the attribute 'beach' would be yes or no. By multiplying the corresponding fractions the outcome of yes is

$$Likelihood\ yes = \frac{1}{8} * \frac{1}{8} * \frac{3}{8} * \frac{5}{8} = 0.004 \tag{29}$$

and the outcome of no is

$$Likelihood\ no = \frac{2}{8} * \frac{2}{8} * \frac{1}{8} * \frac{3}{8} = 0.003 \tag{30}$$

we see that given the new instance we are more likely to head to the beach. Normalizing the likelihoods to one yields the associated probabilities

$$Probability\ yes = \frac{0.004}{0.004 + 0.003} = 0.571\ or\ 57.1\% \tag{31}$$

$$Probability\ no = \frac{0.003}{0.004 + 0.003} = 0.429\ or\ 42.9\% \tag{32}$$

The combination of using Baye's rule and that the classifier 'naively' assumes independence, give this method the name Naïve Bayes. While this method is unrealistic, in practice it has shown to work well and produces a strong classifier [65]. Naïve Bayes classifiers yield probability estimates of a given class rather than a binary class prediction. The conditional probability of class values is estimated given the values of all other attributes [65]. Bayesian networks provide a concise way to describe probability distributions for a given classification problem. It is represented as network of nodes each containing the probabilities for the various outcomes possible for its respective attribute. Graphically, this looks like

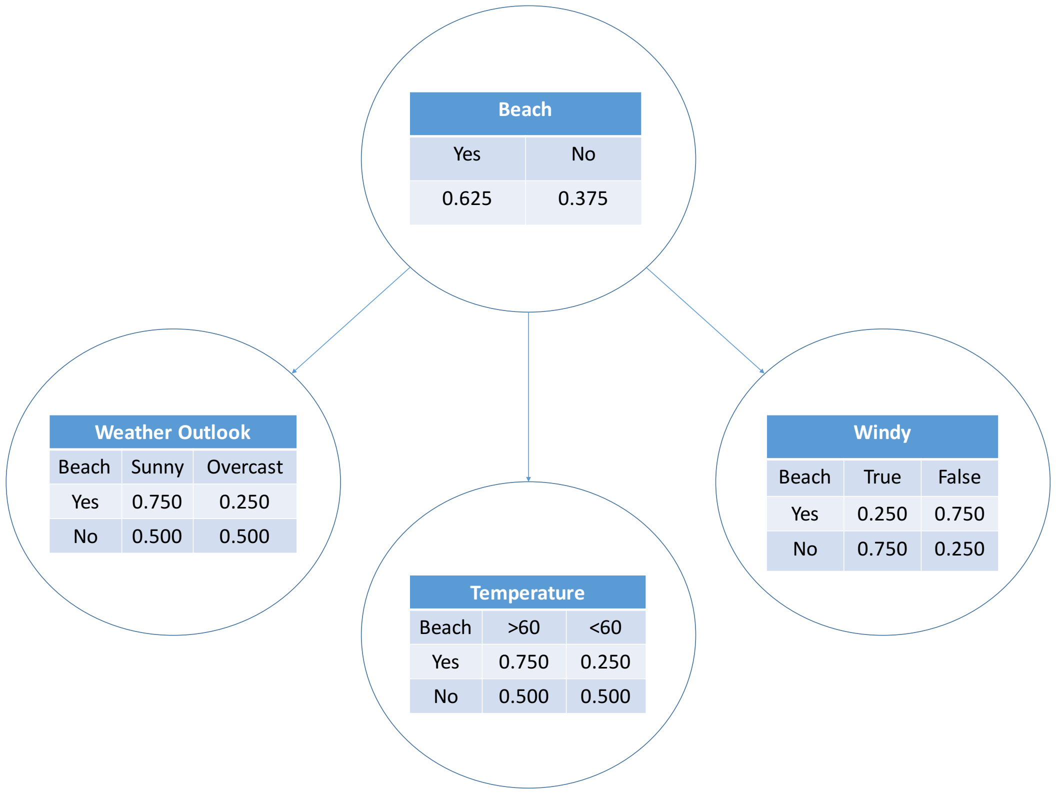


Figure 8: Bayesian network using beach data.

The top node is the parent node containing the probability distribution for the class attribute beach. The other nodes are called daughter nodes and contain probability distributions for their corresponding attributes. The outcome probabilities are calculated the same way as in our previous example. This multiplication process is carried out recursively between a single attribute and the rest of the attributes via the chain rule. The chain rule is expressed as [65]

$$P(A_1, A_2, ..., A_n) = P(A_1) \prod_{i=1}^{n-1} P(A_{i+1}|A_{i-1}, ..., A_1) \tag{33}$$

Equation 33 is a decomposition of the joint probability of n attributes A_i and is the underlying mechanism for computing the probabilities in Bayesian networks..

3 Temporal Spectroscopy Theory

Temporal spectroscopy is an active interrogation technique which produces a gamma-ray spectrum following thermal neutron induced fission. Fission fragments produced from the fission of SNM, will beta decay and produce gamma-ray emissions. The buildup and decay of the radionuclides yield information regarding the temporal behavior of the spectrum. Regions of interest are identified and their temporal response is analyzed. Through their temporal ratios, the mass percentage of a material may be determined.

As mentioned previously, the initial temporal method was developed by Chivers et al. His discrete method evaluated integrated count rate spectra, as seen in Figure 9, over a specified time interval. To estimate the percentage of fissile material, a static ratio between integrated counts at two different times was used. Referring to Figure 9, F^{mix} is the estimated fissile content, $\int_0^{t_x} N^{mix}(t)dt$ and $\int_0^{t_{ref}} N^{mix}(t)dt$ are the integrals representing the integrated counts for the given time range. Rather than using discrete gamma-ray lines, Chivers' method uses wider regions of interest yielding improved statistics over shorter counting periods.

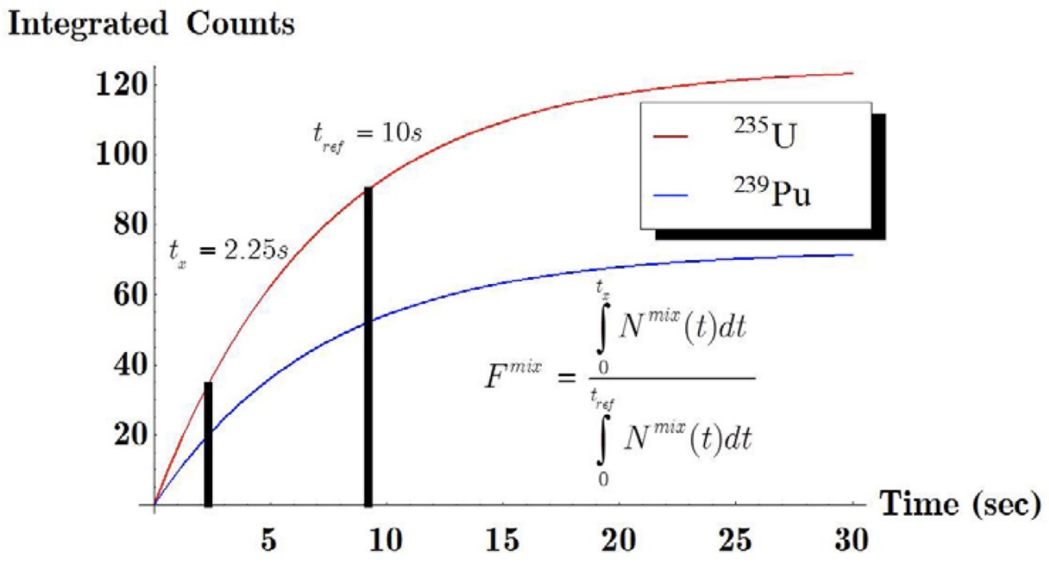
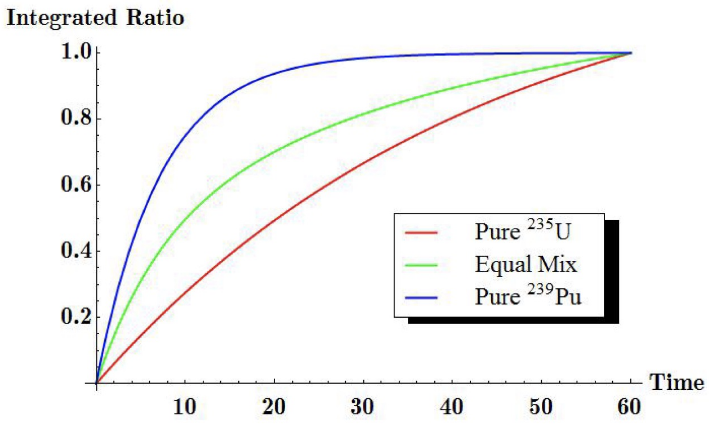
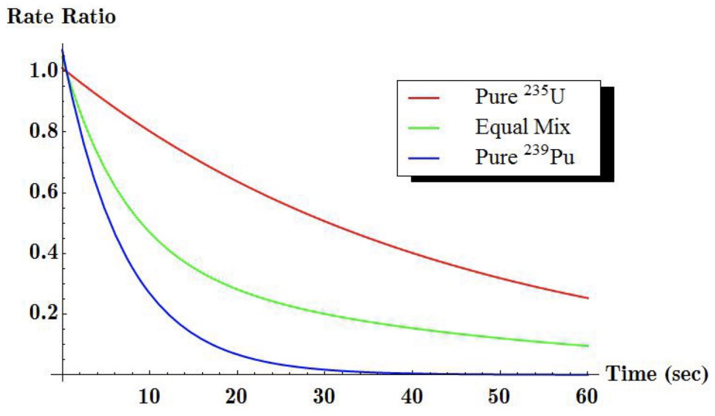


Figure 9: Temporal method [64].

Williford uses Chivers' theoretical method as the foundation for his temporal spectroscopy model. Rather than using discrete static measurements, Williford's method uses continuous temporal measurements. Fissile mass estimation may be performed using either integrated counts or count rate spectra of a given region. Using integrated counts increases the fidelity of the measurement while using count rates provides an additional method for determining the fissile mass estimate. Figures 10a and 10b display the normalized integrated count and count rate spectra, respectively.



(a)



(b)

Figure 10: Plots showing (a) integrated count rate ratio and (b) count rate ratio for pure and an equal mix of fissile materials [64].

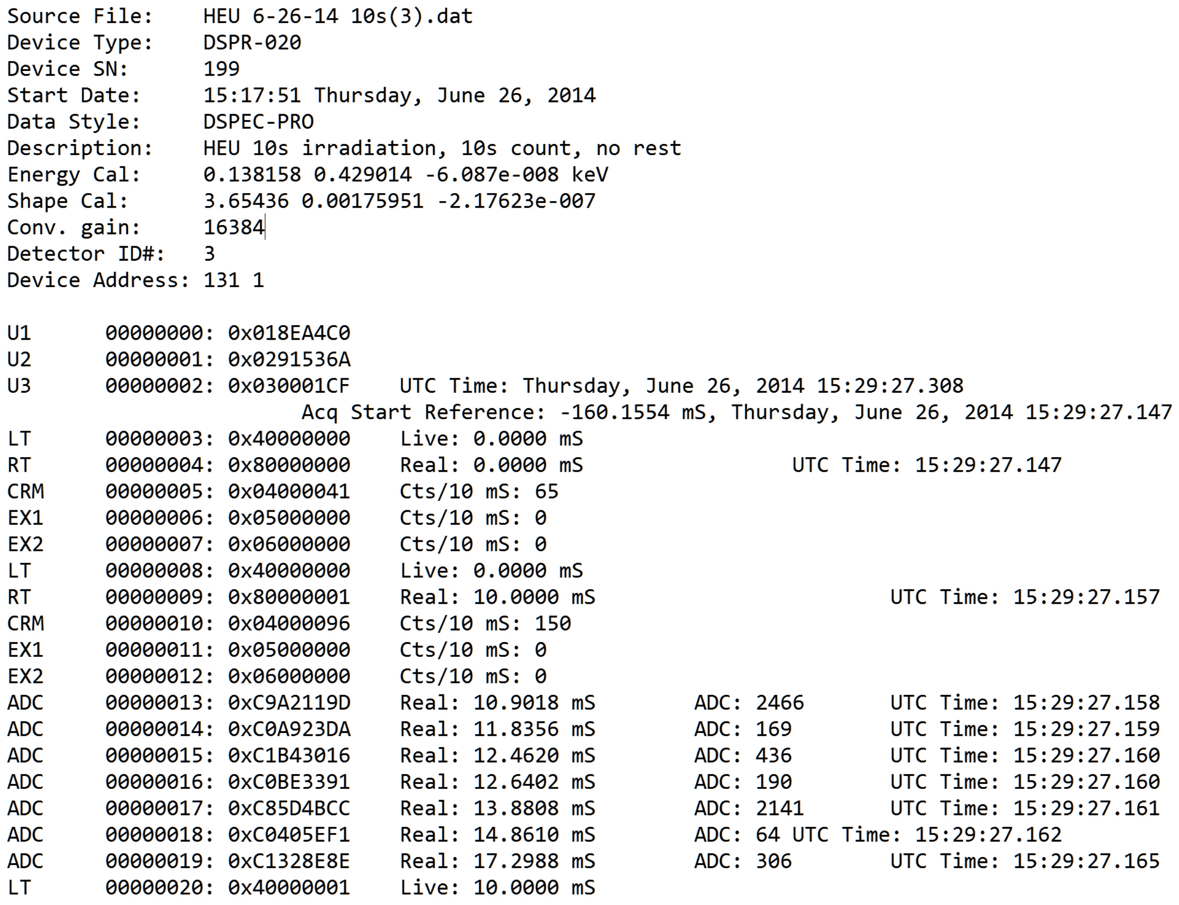
The continuous temporal ratios are developed by dividing the counts at each time interval by the counts at a reference time. The reference time used is based on the interval with the highest number of counts. Evaluating the temporal responses in the specified regions of interest yields the amount of SNM present in a given sample.

3.1 Collection of Temporal Spectroscopy Data

Temporal data are collected using a DSPEC pro digital multi-channel analyzer. Operating the DSPEC pro in List Mode, information such as the time of detection, energy channel, as well as live and real time data are collected and streamed to the computer. This data file can then be converted into a usable text file. For this research, the DSPEC Pro was operated in List Mode rather using traditional spectroscopy methods which result in a spectrum output for the user.

ListPRO is a program developed by ORTEC and is used to retrieve the List Mode data which is stored in an internal memory cache [64] every ten milliseconds. Following data retrieval, the cache is cleared and all data are dumped. The result is a large data file (.dat) of each detected count, its respective time of occurrence (in increments of 200 nanoseconds), the channel in which the count occurred (ADC), running live and real time (in increments of ten milliseconds), header information as well as external port readings (in increments of ten milliseconds) that must be converted to a text file by the ORTEC ListDump program for further use [64].

The ORTEC ListDump program reads the .dat file and converts it to a tab delimited text file which displays a list of sequential events along with header information. Figure 11 shows an excerpt of the List Mode text file produced.



```
Source File:    HEU 6-26-14 10s(3).dat
Device Type:    DSPR-020
Device SN:      199
Start Date:     15:17:51 Thursday, June 26, 2014
Data Style:     DSPEC-PRO
Description:    HEU 10s irradiation, 10s count, no rest
Energy Cal:     0.138158 0.429014 -6.087e-008 keV
Shape Cal:      3.65436 0.00175951 -2.17623e-007
Conv. gain:     16384
Detector ID#:   3
Device Address: 131 1

U1      00000000: 0x018EA4C0
U2      00000001: 0x0291536A
U3      00000002: 0x030001CF    UTC Time: Thursday, June 26, 2014 15:29:27.308
                        Acq Start Reference: -160.1554 mS, Thursday, June 26, 2014 15:29:27.147
LT      00000003: 0x40000000    Live: 0.0000 mS
RT      00000004: 0x80000000    Real: 0.0000 mS                 UTC Time: 15:29:27.147
CRM     00000005: 0x04000041    Cts/10 mS: 65
EX1     00000006: 0x05000000    Cts/10 mS: 0
EX2     00000007: 0x06000000    Cts/10 mS: 0
LT      00000008: 0x40000000    Live: 0.0000 mS
RT      00000009: 0x80000001    Real: 10.0000 mS                        UTC Time: 15:29:27.157
CRM     00000010: 0x04000096    Cts/10 mS: 150
EX1     00000011: 0x05000000    Cts/10 mS: 0
EX2     00000012: 0x06000000    Cts/10 mS: 0
ADC     00000013: 0xC9A2119D    Real: 10.9018 mS        ADC: 2466       UTC Time: 15:29:27.158
ADC     00000014: 0xC0A923DA    Real: 11.8356 mS        ADC: 169        UTC Time: 15:29:27.159
ADC     00000015: 0xC1B43016    Real: 12.4620 mS        ADC: 436        UTC Time: 15:29:27.160
ADC     00000016: 0xC0BE3391    Real: 12.6402 mS        ADC: 190        UTC Time: 15:29:27.160
ADC     00000017: 0xC85D4BCC    Real: 13.8808 mS        ADC: 2141       UTC Time: 15:29:27.161
ADC     00000018: 0xC0405EF1    Real: 14.8610 mS        ADC: 64 UTC Time: 15:29:27.162
ADC     00000019: 0xC1328E8E    Real: 17.2988 mS        ADC: 306        UTC Time: 15:29:27.165
LT      00000020: 0x40000001    Live: 10.0000 mS
```

Figure 11: List mode text file output.

U1, U2 and U3 separate the Coordinated Universal Time (UTC) timestamps, the spectrometer's count-rate meter (CRM) records the time since the last cache dump, EX1 and EX2 indicate the value of the signal into external ports 1 and 2 respectively, ADC represents the channel in which the count was detected, LT represents the live time and RT represents the real time.

List Mode data are collected continuously during irradiation cycles whether the sample is in front of the detector or being irradiated. The List Mode data files need to be parsed in order to evaluate the data when the sample is in front of the detector. A parse code developed in Python by Williford is used. As mentioned earlier, in the List Mode text file the value of EX 1 represents where the sample is located. EX 1 has a value of 0 during sample irradiation and then switches to a value of 1 when the sample is at the detector. The parse code is written to look for the the switch from 0 to 1. Parsing of the raw data results in forty individual one-minute time stamped data files, twenty files corresponding to the twenty cycles of irradiation/counting performed for each material. These files contain the time at which a count is detected and the channel (ADC) in which it occurs.

3.2 Multidimensional Analysis for Mass Estimation

Once the List Mode data are collected, parsed, significant regions of interest are identified and the spectrum is normalized by the temporal response at some reference time, the multidimensional analysis may be performed. It is important that reference spectra and sample spectra are collected using the same geometry and cycle protocols to properly conduct a multidimensional analysis.

To estimate the relative content of fissile materials, the mixture of two materials can be represented as a linear combination:

$$a = \sum_{i=0}^{2} c_i \times \omega_i \tag{34}$$

where, a is the temporal response spectra of a combination of materials, c_i is the i^{th} temporal response value and ω_i is the i^{th} relative weight for i materials. As an example, given ten time bins and two materials the resultant matrix, a, is:

$$a = \begin{pmatrix} c_{11}\omega_{11} \\ c_{12}\omega_{12} \\ c_{13}\omega_{13} \\ \vdots \\ c_{110}\omega_{110} \end{pmatrix} + \begin{pmatrix} c_{21}\omega_{21} \\ c_{22}\omega_{22} \\ c_{23}\omega_{23} \\ \vdots \\ c_{210}\omega_{210} \end{pmatrix} = \begin{pmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_{10} \end{pmatrix} \tag{35}$$

where, the sum of $c_{11}\omega_{11} + c_{21}\omega_{21}$ results in a single value located as the first vector value in a and so forth. As mentioned previously, the temporal spectra can either be integrated counts or the count rates for a given region of interest. The multidimensional analysis technique is developed from expanding the above relationship into a system of linear equations. It is used to estimate the weight values of the materials of interest given the temporal data of pure samples and the mixture.

Reference spectra are defined as the temporal spectra of samples with known concentrations. In this research, samples contain highly enriched uranium (HEU) and Pu-239. The reference spectra, $\vec{s_i}$, are normalized and assembled into a reference matrix, S. For example, the following is an S matrix for a single region of interest. We see $\vec{s}$ of ten time intervals for materials of interest.

$$S = \left(\begin{pmatrix} s_{U1} \\ \vdots \\ s_{U10} \end{pmatrix} \begin{pmatrix} s_{Pu2} \\ \vdots \\ s_{Pu10} \end{pmatrix} \cdots \begin{pmatrix} s_{i1} \\ \vdots \\ s_{i10} \end{pmatrix} \right) \tag{36}$$

The reference matrix, S, is used to generate a basis set of vectors by performing a singular value decomposition (SVD). Three matrices are formed after performing an SVD on a matrix. Matrix U is a left singular orthogonal matrix containing the eigen vectors, Σ is a matrix of the square root of the eigen values and V is the right singular orthogonal matrix. Using matrix U, a basis matrix of the reference spectra is formed. The SVD of matrix S is written as:

$$S = U \times \Sigma \times V^T \tag{37}$$

Using the left singular orthogonal matrix U, the basis matrix A of the reference spectra is created:

$$A = U^T \times S \tag{38}$$

Next, a composition matrix, W, containing the weights of the reference spectra for each isotope $\vec{\omega}_i$ is created. Seen here is an example of matrix W for two reference spectra with 100% pure HEU and Pu-239. The relative weights for each isotope are located in the rows.

$$W = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{39}$$

Assuming all observed temporal spectra are a linear combination of the independent isotopic spectra allows the matrix A to be written as the weight matrix multiplied by an unknown transformation matrix, C. The transformation matrix determines how the isotopic weights contribute to the temporal spectrum.

$$A = C \times W \tag{40}$$

The transformation matrix is found by taking the inverse of matrix W. However, the reference matrix W may not be square. This poses challenges when finding the inverse matrix as it may not always be possible. Therefore, a pseudo-inverse matrix can be found by taking the SVD of matrix W [Strang, 2006]. The SVD of W results in:

$$W = u \times \sigma \times v^T \tag{41}$$

allowing the pseudo-inverse W^+ to be constructed by

$$W^+ = v \times \sigma^{-1} \times u^T \tag{42}$$

and allowing C to be constructed from

$$C = A \times W^{+} \tag{43}$$

Similarly, an observed temporal spectrum of unknown composition $\vec{o}$ can be transformed into the vector space defined by the orthogonal matrix A. The transformed spectrum $\vec{\alpha}$ of the unknown sample is written as:

$$\vec{\alpha} = U^{T} \times \vec{o} \tag{44}$$

Writing $\vec{\alpha}$ as a linear combination of the transformation matrix C and the relative weights from each fissionable isotope, we get the following:

$$\vec{\alpha} = C \times \vec{\omega} \tag{45}$$

Estimating fissile material content requires solving for $\vec{\omega}$. Similar to how W^{+} was found, the pseudo-inverse of C needs to be found. The result is:

$$\vec{\omega} = C^{+} \times \vec{\alpha} \tag{46}$$

3.3 Computing Relative Mass Content

The multidimensional analysis results in a weight vector, $\vec{\omega}$, consisting of normalized weight values. To estimate the actual weight values representing the associated mass contribution of fissile materials, algebraic manipulation is required. The normalized temporal data are defined as:

$$\alpha(t) = c_1(t) \times \omega_1 + c_2(t) \times \omega_2 \tag{47}$$

where $\alpha(t)$ is the temporal spectrum of the unknown sample, $c_i(t)$ is the temporal spectra of the known constituents and ω_i is the relative temporal weights of the constituent spectra. Recalling that the temporal spectrum of the unknown sample $\alpha(t)$ is composed by normalizing the standard spectrum by the response at a reference time t_r, we can write the normalized unknown temporal spectra as:

$$\alpha(t) = \frac{a(t)}{a(t_r)} \tag{48}$$

where, $a(t)$ is the standard spectrum of the unknown sample at time t and $a(t_r)$ is the standard spectrum at a reference time t_r. The standard spectra are represented as a linear combination of the un-normalized spectra n. However, the weights are not normalized and therefore differ resulting in the following formulas:

$$a(t) = n_1(t) \times y_1 + n_2(t) \times y_2 \tag{49}$$

$$a(t_r) = n_1(t_r) \times y_1 + n_2(t_r) \times y_2 \tag{50}$$

Note: given that the weights differ, they are represented by the variable y_i rather than by ω_i. Setting the equations equal to one another in terms of a single material allows solving for the un-normalized weights y_i in terms of the normalized weights ω_i for the material of interest.

$$C_1(t) \times \omega_1 = \frac{a_1(t)}{a_1(t_r)} = \frac{n_1(t) \times y_1}{n_1(t_r) \times y_1 + n_2(t_r) \times y_2} \tag{51}$$

Recognizing that $y_2 = 1 - y_1$ and that the normalized response $c_1(t_r)$ is equal to 1 when the equation is analyzed at $t = t_r$, allows equation (51) to be simplified to:

$$\omega_1 = \frac{n_1(t) \times y_1}{n_1(t_r) \times y_1 + n_2(t_r) \times (1 - y_1)} \tag{52}$$

To obtain the true relative weight of the material of interest in the unknown sample, we solve for y_1:

$$y_1 = \frac{\omega_1 \times n_2(t_r)}{\omega_1 \times n_2(t_r) + \omega_2 \times n_1(t_r)} \tag{53}$$

4 Knowledge Gap

To achieve the desired accuracy during classification, machine learning algorithms require pre-processing of data. To date, methods have included peak energy and peak search procedures. Specifically, Sullivan et al. created a library of specific peak energies and branching ratios while Yoshida et al. performed a peak search procedure based on peak energies of interest [57, 67]. In this research, specific peak energies which correlate to known isotopes of interest will not be utilized for classifier training. Instead, the user implements a novel method generating new features during data pre-processing. The user transforms the data using statistical methods such as mean, median and Speraman's correlation coefficient creating new features which describe meaningful patterns present in the spectra. Rather than letting known isotope energies drive the data mining process, the data speaks for itself.

Further, machine learning has not been applied to temporal spectroscopy methods. To date, temporal spectroscopy methods used for mass estimation have been performed on higher energy portions of the spectrum. Combining temporal spectroscopy with machine learning methods allows the entire spectrum to be utilized in the quantification of mass content.

5 Advantages of Combining Machine Learning and Temporal Spectroscopy Methods

This research adds to the body of knowledge by expanding the techniques used for the classification and mass estimation of fissile material. Introducing novel data pre-processing methods and combining machine learning algorithms with temporal spectroscopy methods has the potential to provide improved precision and accuracy of the classification and mass estimation of fissile material without the need for efficiency calibration. Compared to previous temporal spectroscopy methods, employing machine learning methods results in additional regions of interest to be used for mass estimation. These additional regions may provide increased accuracy of mass content estimates. Further, we have seen in the literature that machine learning methods have been shown to work for detectors with appreciable calibration drift [57]. Finally, machine learning limits the need for user intervention and expert knowledge once the model has been trained [48] allowing for an automated process that is easily implemented into active interrogation methods for nonproliferation and nuclear security.

6 Materials and Methods

6.1 Facility Design and Experimental Setup

The Oregon State University TRIGA Reactor (OSTR), illustrated in Figure 12, is a natural convection cooled, pool-type 1 megawatt (MW) research reactor. The reactor core contains fuel elements composed of low-enriched uranium homogeneously combined with zirconium-hydride and is surrounded by a graphite reflector [4]. Four beam ports penetrate the concrete shield and reactor tank.

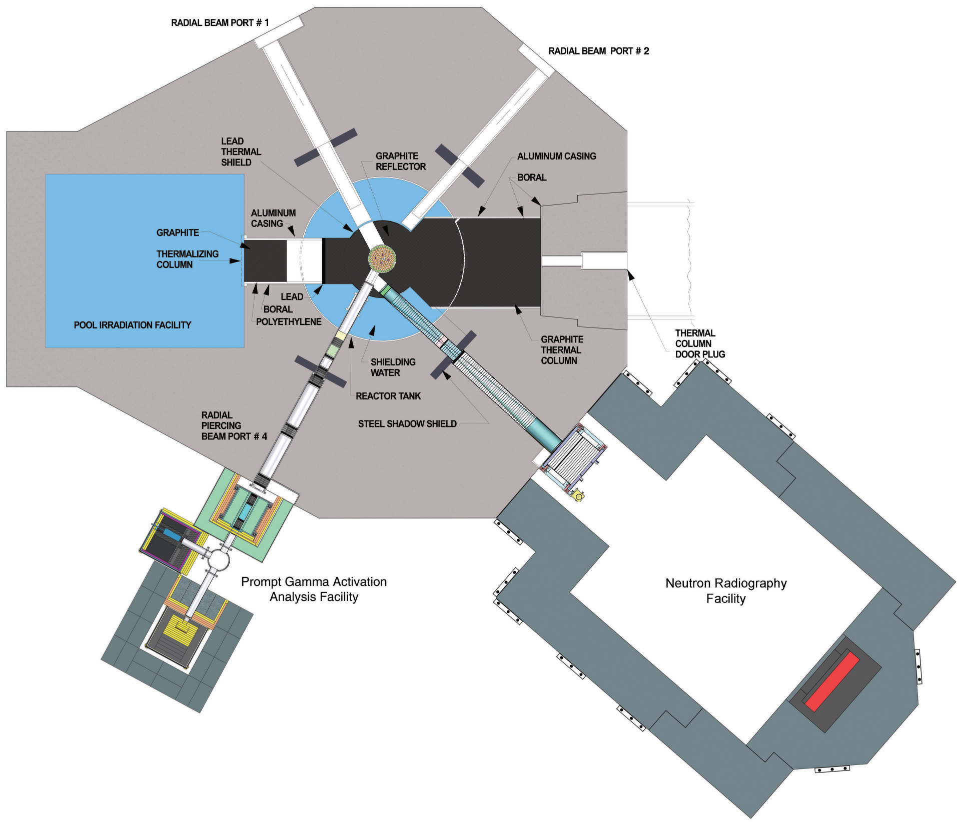


Figure 12: Schematic of the Oregon State University TRIGA Reactor [64].

The Prompt Gamma Neutron Activation Analysis Facility (PGNAA), illustrated in Figures 13 and 14, utilizes beam port 4 of the OSTR. Beam port 4 penetrates the graphite reflector, channeling neutrons from the core providing the highest thermal neutron flux [50]. The facility consists of a collimator, beam shutter, sample chamber, beam stop and a coaxial, high purity geranium (HPGe) detector. The detector is housed within borated polyethylene and lead shielding and is positioned 90 degrees from the neutron beam. Lead and boral rings located in the collimator are used to collimate the neutron beam to a uniform diameter of 2 cm. The beam is filtered by bismuth and sapphire filters which attenuate gamma rays and fast neutrons, respectively. The shutter can be moved to prevent the neutron beam from entering the sample chamber when the beam port is not in use. The facility components are designed so that they can be evacuated or back-filled with helium to reduce neutron interactions with air that would increase background radiation levels and decrease the signal-to-noise ratio of the detector [50].

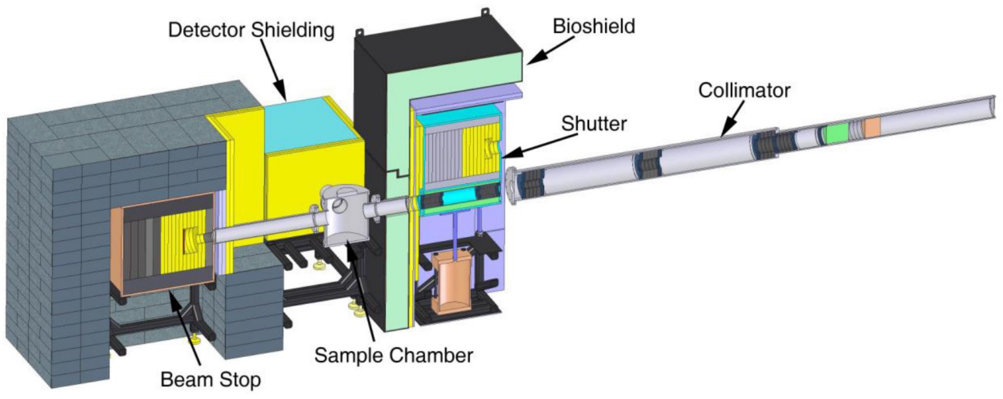


Figure 13: Schematic of beam port 4 [64].

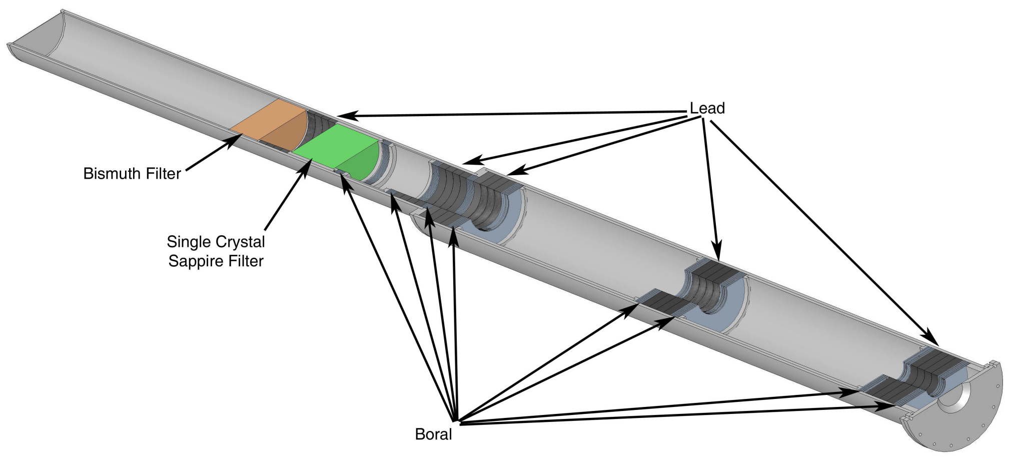


Figure 14: Schematic of beam port 4 [64].

A pneumatic transfer system (Fast Rabbit Pneumatic Transfer System), illustrated in Figure 15, was built around the PGNAA facility at the end of beam port 4 to maximize the counting geometry and reduce background counts for samples. The distance between the sample and the detector is 10 cm, providing better measurement efficiency for low yield fission products from HEU and Pu-239 [64]. The pneumatic transfer system was used to shuttle the sample between the beam and the detector. The polyethylene encapsulated sample was propelled through the system using pressurized helium with a transit time of 100 milliseconds. A programmable logic controller (PLC) was setup to automate the irradiation time, count time and rest time. Two optical sensors were used to send a signal to the PLC indicating when the sample was being counted or irradiated. This information was sent to the digital spectrometer where it was recorded in the List Mode data files. Figure 16 shows photographs of the setup of the pneumatic transfer system and detector.

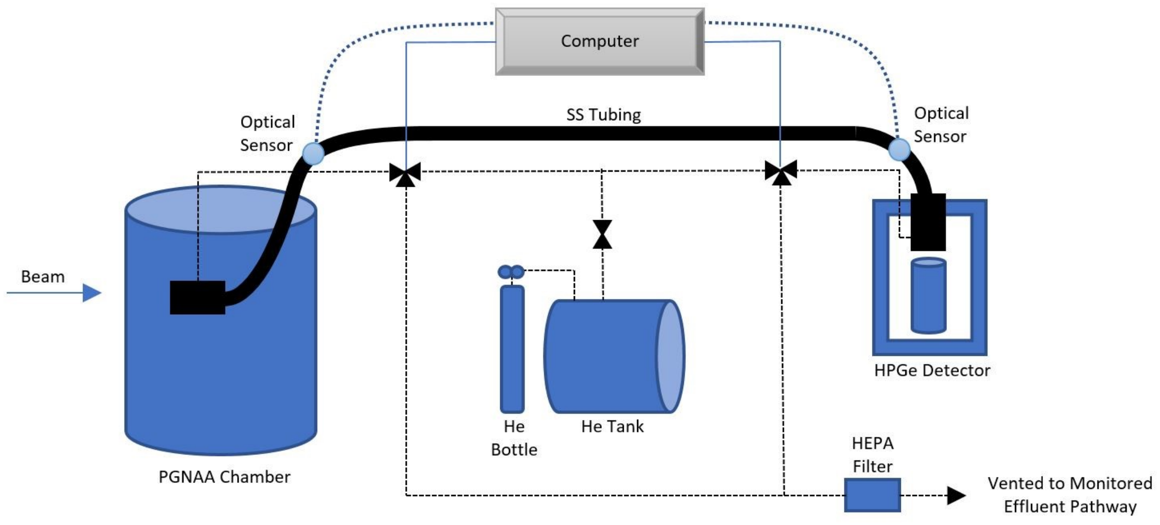


Figure 15: Schematic of the Fast Rabbit Pneumatic Transfer System [64].

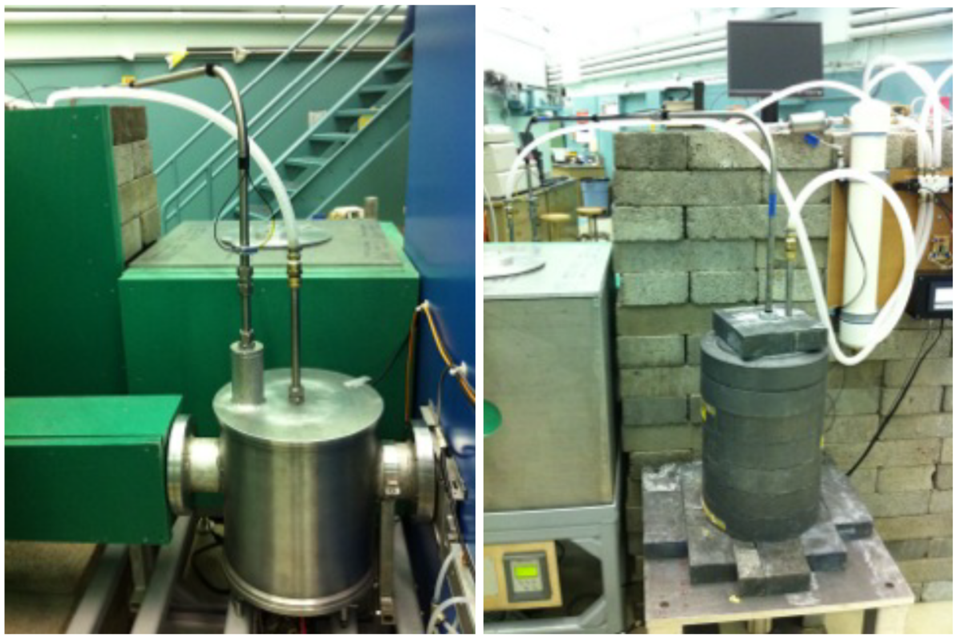
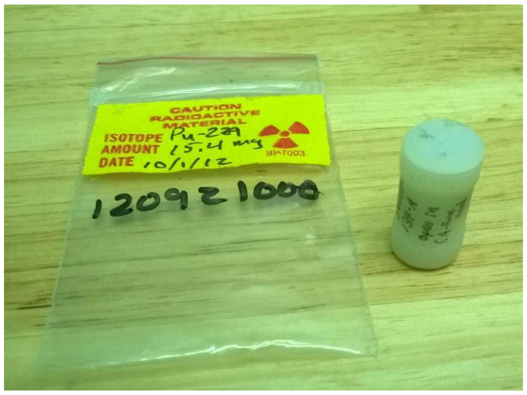


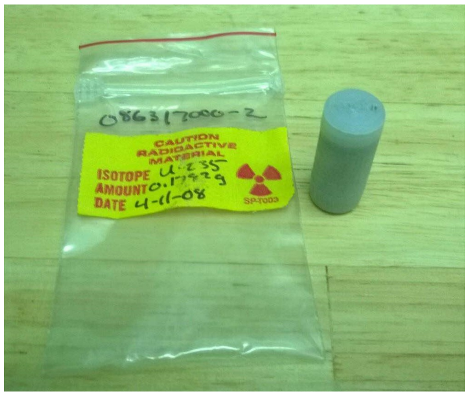
Figure 16: Photographs of the pneumatic transfer system and detector [64].

6.2 Data Collection

Samples of 178.2 mg of U-235 nominally enriched to 93% and 244 mg of Pu-239 are irradiated and counted in one-minute intervals for twenty cycles each using beam port 4 and the fast rabbit pneumatic transfer system. Fissile material was placed in a Teflon bag and the entire sample was placed in a high-density polyethylene (HDPE) capsule. To increase the number of counts, the capsule was oriented so that the bottom of the HDPE capsule was parallel to the detector face. Figures 17*b* and 17*a* show photographs of the samples of the fissile materials.



(a) Pu-239 (244 mg) sample.



(b) HEU (178.2 mg) sample.

Figure 17: Photograph of the HEU and Pu-239 samples.

The detector is coupled with a ORTEC DSPEC Pro digital spectrometer which can be operated in standard mode using GammaVision or in List Mode [64]. As mentioned previously, when operating in List Mode information such as the time of detection, energy channel, as well as live and real-time data are collected and streamed to the computer continuously. This data file can be converted into a usable text file and parsed to evaluate data collected when the sample is in front of the detector. Parsing the raw data results in forty individual one-minute time-stamped data files; twenty files corresponding to the twenty cycles of counting performed for each material. Table 5 shows an excerpt of the time-stamped data files for HEU from cycle 10.

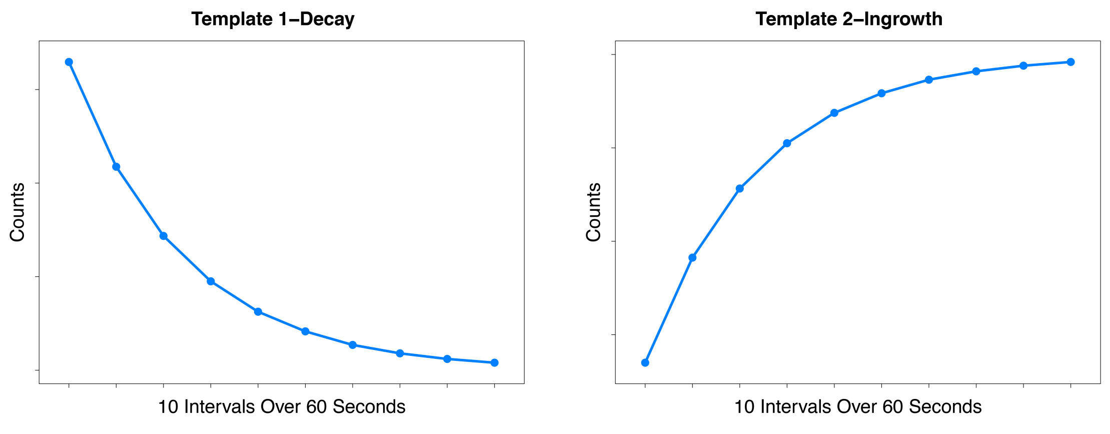
Table 5: Example of *ListMode* data frame.

Cycle	**Real Time**	**Channel**	**UTC Time**
10	1144530	1872	15:59:04.402
10	1144530	228	15:59:04.402
10	1144530	557	15:59:04.403
10	1144530	493	15:59:04.403
10	1144530	356	15:59:04.403
10	1144530	793	15:59:04.403
10	1144530	528	15:59:04.403
10	1144531	113	15:59:04.403
10	1144531	93	15:59:04.403
10	1144531	2945	15:59:04.404

It is noted that each run cycle ends at about 58.8 seconds rather than the 60 second time period entered on the PLC. This is likely due to experimental error which arises from things such as the laser shifting alignment on the transfer tubing and changes in air pressure during sample transit. It is also noted that there is buildup present in the data. This is due to longer lived isotopes not decaying away prior to the next irradiation/counting cycle. Incorporating a rest period would minimize such effects however it will never fully be eliminated. Further, it was shown in previous work by Williford that a dead time correction was unnecessary as dead time was accurately estimated using List Mode [64].

6.3 Template Development

Using the basic principles of the radioactive decay law, two templates were created to form a training set of patterns to be used in the computation of Spearman's rank-order correlation coefficient. To perform the computation of the correlation coefficient, Spearman's requires that the data sets being compared have the same number of values. The templates used for the training set were developed to have ten, six-second intervals to accommodate this requirement. Given that Spearman's is based on ranked data, only one template is used for each pattern since varying the half-life for the decay and ingrowth templates returns the same scores.



(a) Template displaying decay pattern. **(b)** Template displaying ingrowth pattern.

Figure 18: Templates designed for the ρ_s feature.

Given that the ingrowth template is perfectly anticorrelated with the decay template, only the decay template was used to generate the ρ_s score feature. Further, postive and negative linear trend templates were not used. The results of the decay template sensitivity analysis showed that varying the decay rate of the function did not effect the ρ_s score value. Even with a long decay rate, a near negative linear trend, the ρ_s score value was still 1. For completeness, positive and negative linear templates were created and run against the decay data. As expected, ρ_s score values were -1 and 1, respectively.

6.4 Data Pre-Processing

Data pre-processing is user-driven and iterative in nature. It requires reducing data, as appropriate, and identifying features which will be used by the classifier during training. Pre-processing may include transforming the data in such a way that a new feature is developed for training, a method utilized in this research. As the classifier is trained, the user receives feedback regarding its accuracy. It may be necessary to implement various methods of data pre-processing to reach the desired level of accuracy.

To evaluate how the spectrum was changing as a function of time, the parsed raw data needed to be converted to an interval matrix as seen in Table 6. A temporal interval matrix was generated using the raw time-stamped count data. The time field was divided into a sequence of ten, six-second intervals for each one-minute cycle.

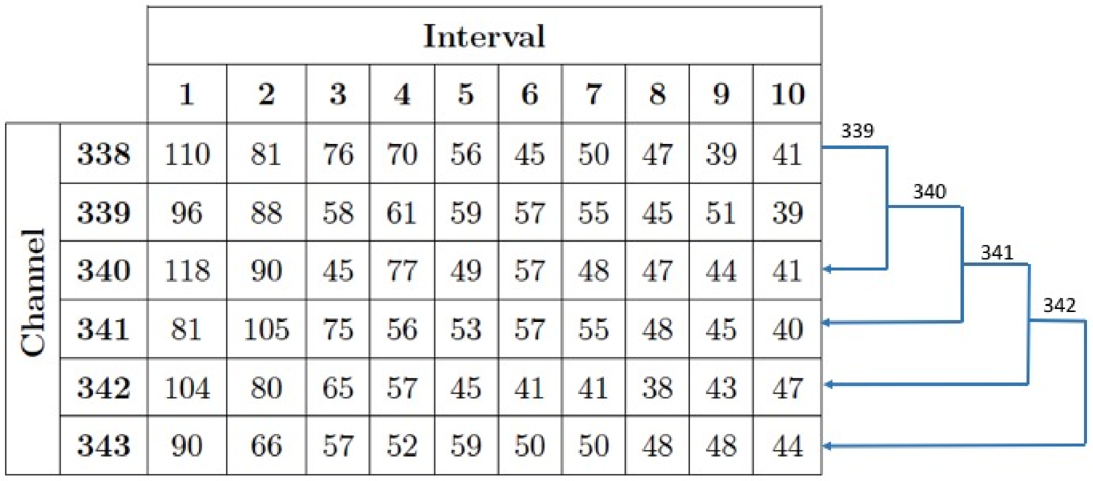
Table **??** is an example of the resulting temporal interval matrix from a single cycle.

Table 6: Excerpt of the HEU temporal interval matrix for cycle 10.

		Interval									
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
Channel	**338**	110	81	76	70	56	45	50	47	39	41
	339	96	88	58	61	59	57	55	45	51	39
	340	118	90	45	77	49	57	48	47	44	41
	341	81	105	75	56	53	57	55	48	45	40

To accurately capture the inherent resolution of an HPGe detector as well as the underlying physical phenomena, sliding windows were implemented. Channels were grouped in windows of three and five prior to feature generation and training of the classifiers. This allowed for a fair assessment of the optimal sliding window to capture the differences in the patterns of decay for the materials under evaluation. Further, channels above 3,000 (corresponding to an energy of 1.2 MeV) were eliminated. This threshold was chosen for two main reasons. First, the primary focus of this research was to evaluate the spectrum below 3 MeV. Second, the channels at the higher energies did not have enough counts present to display a significant temporal pattern.

Sliding windows consisted of performing a rolling sum for each time interval over a given number of consecutive channels. The middle channel number of the consecutively summed channels was chosen as the new channel label for a given group. This ensured that the underlying energy association was preserved. Figure 19 shows an example of how the sliding window for 3 channels was performed.



		Interval									
		1	2	3	4	5	6	7	8	9	10
Channel	338	110	81	76	70	56	45	50	47	39	41
	339	96	88	58	61	59	57	55	45	51	39
	340	118	90	45	77	49	57	48	47	44	41
	341	81	105	75	56	53	57	55	48	45	40
	342	104	80	65	57	45	41	41	38	43	47
	343	90	66	57	52	59	50	50	48	48	44

339
340
341
342

Figure 19: Example of the sliding window of three and the labeling of the new channels.

Feature scaling is sometimes performed when features consist of values with varying magnitudes. Un-normalized features have the potential to be unevenly weighted in some machine learning algorithms. This is most commonly an issue for algorithms that rely on Euclidean distance measures [53, 5]. Tree based methods, Naïve Bayes and gradient descent algorithms are all known to perform well despite features with a range of magnitudes. The main concern in gradient descent algorithms is the speed at which convergence is achieved. Given that the machine learning architectures chosen for this research are not impacted by a lack of feature scaling, normalization of the temporal data was not performed for classification. Rather, the raw counts were used.

Once the new data frames were created, features of mean, median and the ρ_s score were generated. Fractional ranking, based on the number of counts present at each interval, was performed on the temporal interval matrices as well as the templates to compute ρ_s. The temporal interval matrices were compared individually to the templates to tease out decay and ingrowth patterns. A ρ_s score was computed for each channel in all cycles of each material. As an example, Tables 7 and 8 show the temporal interval matrices with computed ρ_s for Pu-239 and HEU.

Table 7: HEU temporal matrix.

		Interval										**Decay**
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	$\rho_{s,HEU}$
Channel	**301**	108	97	75	72	67	57	47	43	40	57	0.90
	302	88	73	69	71	61	55	60	50	56	46	0.92
	303	94	81	72	77	70	43	59	53	54	55	0.81
	433	152	158	157	175	171	161	145	171	160	166	-0.30

Table 8: Pu temporal matrix.

		Interval										**Decay**
		1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	$\rho_{s,Pu}$
Channel	**301**	162	194	189	174	197	205	202	206	226	194	-0.71
	302	186	223	205	177	192	233	228	218	213	232	-0.52
	303	181	172	174	167	155	193	164	191	197	173	-0.23
	433	102	86	66	76	60	62	76	53	60	57	0.81

Further, the ρ_s score feature was discretized into nine categories resulting in an additional feature to be used in the training of the classifiers. The following table shows how the categories were defined.

Table 9: Ranges of ρ_s values used to create discretized categories.

ρ_s **Range**	ρ_s **Category**
1.0000000	Perfect Positive
0.7000000 to 0.999999	Strong Positive
0.5000000 to 0.6999999	Moderate Positive
0.3000000 to 0.4999999	Weak Positive
-0.2999999 to 0.2999999	No Relationship
-0.3000000 to -0.4999999	Weak Negative
-0.5000000 to -0.6999999	Moderate Negative
-0.7000000 to -0.9999999	Strong Negative
-1.0000000	Perfect Negative

A two-sample t-test was used to evaluate each channel/feature combination across the 20 cycles between the two materials. This gave us a metric to use for the ordering and selection of significant channels. Welch's two sample t-test for unequal variances compares two population means. The test assumes normality of the population distribution, independence within and between groups but does not assume equal population variances. Welch's t-test gives approximately valid inference for any sample size.

It is important to state that the assumption of independence within groups was violated in our data due to the buildup of longer lived fission products from one irradiation/counting cycle to the next. Violations of independence can lead to bias and skewness in the data. In practice, it is sometimes difficult to ensure complete independence. If it is expected that the violation will result in significant bias, one option is to try and account for the correlations in your model. While a lack of independence may introduce bias into the results, there are certain situations where the lack of independence does not have a large impact on the outcome. For example, Naïve Bayes classifiers are based on the assumption of conditional independence between attributes of a given class. However, this assumption rarely holds. Despite the violation of independence, these classifiers typically perform quite well and can yield strong classification results. After an initial exploratory look at the data it appeared that the effect of buildup would not significantly affect the underlying patterns used for differentiating the fissile materials. Further, the p-value was used to give an ordering and not necessarily a probabilistic interpretation. Therefore, we chose to continue with the analysis without accounting for buildup.

The t-test was conducted at a 95% confidence level. Channels with a corresponding p-value of 0.05 or lower qualify as statistically significant. Further, p-values < 0.001 are known to be highly significant. P-values ranging from 1 e-02 to 1 e-10 were evaluated as cutoff thresholds for determining which channels showed the greatest difference between fissile materials. Below is a summary table of the number of channels identified at each p-value threshold for both sliding windows of three and five.

Table 10: The number of channels identified as significant at each p-value threshold. The Welch's t-test was used to compute the p-values.

P-value	**Sliding Window 3**	**Sliding Window 5**
1e-02	405	460
1e-03	192	198
1e-04	92	98
1e-05	58	64
1e-06	39	42
1e-07	29	28
1e-08	22	22
1e-09	20	14
1e-10	15	10

Attribute selection was performed using the information gain algorithm from the WEKA library. The information gain attribute evaluator uses a rank search method. Attributes are evaluated and ranked based on how much information they contribute with respect to the class. Attributes were evaluated for each p-value threshold and both sliding windows. The top seven features returned for each scenario were utilized for training. Table 11 shows the data for channel 143 at a sliding window of five which includes all the features generated prior to attribute selection.

Table 11: Example input table for channel 143 which includes all the generated features for both fissile materials.

RunID	**HEU1**	…	**HEU10**	**HEU Mean**	**HEU Median**	**HEU ρ_s**	**HEU ρ_s Category**	**Pu1**	…	**Pu10**	**Pu Mean**	**Pu Median**	**Pu ρ_s**	**Pu ρ_s Category**
1	355	…	182	223.5	202.5	0.8909	StrongPos	657	…	592	626.7	638.5	0.2432	NoRelationship
2	424	…	204	263.4	247.5	1.000	PerfectPos	765	…	690	672.7	657.0	-0.1152	NoRelationship
3	440	…	207	278.8	261.0	0.9515	StrongPos	622	…	703	671.8	672.0	-0.4255	WeakNeg
4	415	…	179	257.6	255.0	0.9515	StrongPos	634	…	653	624.1	627.0	-0.1636	NoRelationship
5	391	…	227	289.4	275.5	0.9515	StrongPos	671	…	646	639.5	640.0	0.1636	NoRelationship
⋮	…	…	…	…	…	…	…	…	…	…	…	…	…	…
20	415	…	229	302.3	292.0	0.9273	StrongPos	702	…	725	677.6	679.0	-0.2000	NoRelationship

Data pre-processing results in a final data frame consisting of a subset of distinctive channels, the top seven features selected and their associated class labels to be used for training the classifiers. Table 12 shows the format of the training data used for training each classifier. In practice, each channel was represented 20 times for both materials. The top seven selected attributes are not shown in this table as it varied across p-values. In addition, the number of channels used for training is also dependent on the p-value threshold. It is also important to note that the classifiers were not trained using actual channel values. This column is present in the table to give reference to the reader. The information (features) associated with each channel were used as input for training.

Table 12: Example input table post attribute selection used for classifier training. The number of features and channels varied for each p-value threshold.

Channel	**Class**	**Interval 1**	...	**Interval 10**	**Mean**	**Median**	ρ_s	ρ_s **Category**
138	Pu	762	...	738	775	784.5	0.0182	NoRelationship
138	HEU	247	...	187	195.2	193	0.7660	StrongPos
875	Pu	401	...	406	397.3	394.5	0.3212	WeakPos
875	HEU	157	...	97	109	101	0.6079	ModeratePos

To date, this process has been completed for pure Pu-239 and HEU samples. However, a library of pre-processed data sets for various mixtures of fissile material may be generated and used for training in the future. Next, we will discuss how the channels identified as significant and their associated features were used for training the classifiers.

6.5 Classifier Training and Validation

The initial evaluation consisted of looking at four classifiers: Bayesian network, neural network, decision tree and random forest. Classifiers were trained in WEKA using 10-fold cross validation based on channels identified at each p-value threshold and the given features selected during attribute selection. In addition, training was performed for both sliding windows. Results were compared and the classifiers with the highest training accuracy for a given combination of p-value threshold and sliding window were chosen for validation. The default parameter settings for each classifier in WEKA were utilized.

Data used for validation were collected four years after the data which was used for training. The data consisted of 19 one-minute irradiation/one-minute count files for each material that could be used as "unknown" data sets. While the user was aware of whether the data being used for validation belonged to either HEU or Pu-239, the trained classifiers were shown an unlabeled data set which ensured successful evaluation of a given classifiers performance. Validation requires that the unseen data be in the same format as the training data. This involved generating temporal interval matrices with features that were determined as significant during data pre-processing. In addition, based on an energy calibration, the data frames were reduced to channels matching the energy ranges used for training. Once the files were in the same format, they were passed through the classifiers and their performance was evaluated. Table 13 is an example of the data format for validating the classifiers for a single "unknown" data set from a single material. We see it is in the same format as Table 12 except that the class label associated with each channel is absent and each channel is only represented one time. It is important to note that the classifiers were not validated using actual channel values. This column is present in the table to give reference to the reader. The information (features) associated with each channel were used as input for validation. All of the 19 "unknown" datasets for each material were formatted and shown to the trained classifiers for validation.

Table 13: Example input table used for classifier validation of a single material. Only two channels are shown. The number of channels varied based on p-value threshold.

Channel	**Class**	**Interval 1**	...	**Interval 10**	**Mean**	**Median**	ρ_s	ρ_s **Category**
138	?	736	...	653	691.8	685	0.6970	ModeratePos
875	?	474	...	426	426.9	429.5	0.4182	WeakPos

6.6 Relative Mass Content Estimation

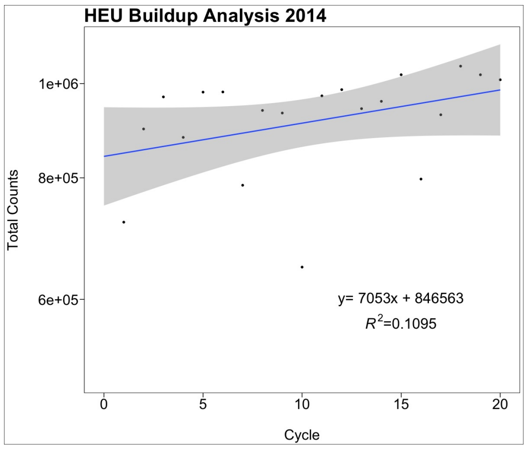
Using the channels identified as significant during data pre-processing, the relative mass content for each material was estimated as proof of concept. This research consisted of looking at pure samples and therefore the relative mass content was estimated for two scenarios: either the sample was 100 percent HEU or 100 percent Pu-239. Mass estimation was performed using the multidimensional analysis described earlier. Williford developed a script in R which takes integrated time interval counts for the reference materials and an unknown material as inputs. The algorithm computes the relative mass percentages of the unknown material as well as the associated error.

Channels evaluated were based on using a sliding window of five. The relative mass algorithm requires that the geometry and irradiation/counting cycles be the same for the reference spectra and the unknown samples [64, 38]. The data collected in 2018 did not consist of a separate data collection for reference spectra. Therefore, the first five cycles of each material, HEU and Pu-239, were summed at each time interval to create two single vectors of temporal counts. The same was done for the next five cycles for each material to create the temporal count vectors for the unknowns. Summing across five cycles was performed to improve counting statistics. Due to the limited data available, only five cycles were summed. Each of the temporal count vectors were then transformed into integrated count vectors. The integrated count vectors were then normalized by a reference time when the counts were the highest. The normalized, integrated count vectors were used to perform relative mass estimation carried out through the statistical program, R.

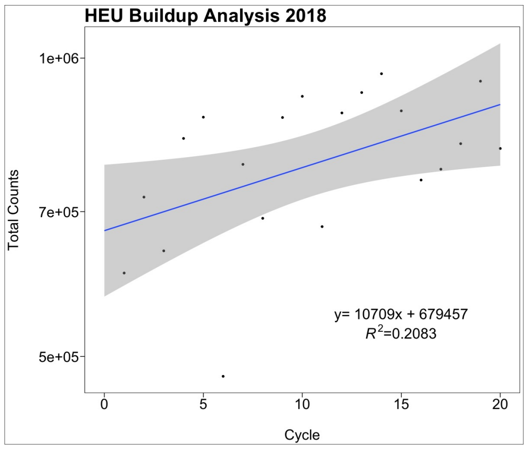
7 Results

7.1 Buildup Analysis

A regression analysis between cycle and the total number of counts was performed on both the 2014 and 2018 data sets to assess for buildup. The regression analysis performed on the 2014 Pu-239 data showed that for each additional cycle, it is expected that the counts will increase by a value of 7,057.6 (p-value< 0.0001). However, for the 2014 HEU data we are unable to reject the null hypothesis that there is no effect between cycle and total number of counts (p-value 0.0844). The increase in counts for Pu-239 in 2014 represents, on average, 0.7% of the total counts for a given cycle. The regression analysis performed on the 2018 data showed that for each additional cycle it is expected that the counts will increase by a value of 10,709 for HEU (p-value 0.0248) and a value of 22,415 for Pu-239, respectively (p-value $<$ 0.0001). The increase in counts for HEU in 2018 represents, on average, 1.3% of the total counts for a given cycle. The increase in counts for Pu-239 in 2018 represents, on average, 2.0% of the total counts for a given cycle. These results confirm the presences of buildup across cycles for both fissile materials.

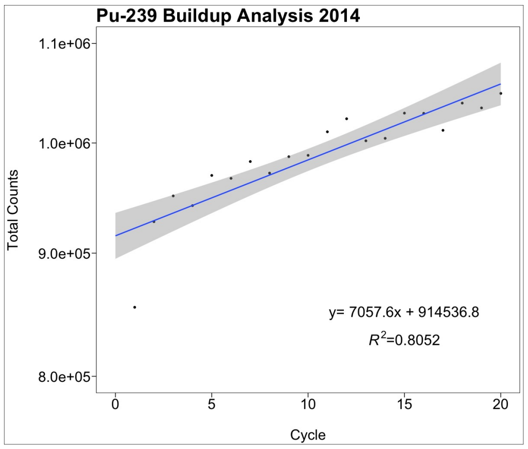


(a)

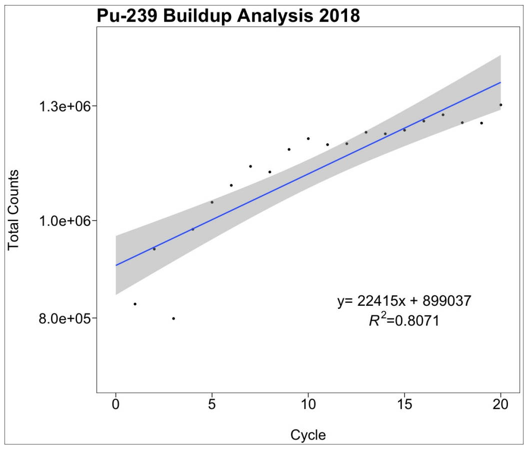


(b)

Figure 20: Regression plots for HEU 2014 and 2018. The gray band represents the 95% confidence interval for the fitted values.



(a)



(b)

Figure 21: Regression plots for Pu-239 for 2014 and 2018. The gray band represents the 95% confidence interval for the fitted values.

7.2 Classifier Evaluation and Selection

Prior to training the classifiers, the information gain algorithm in WEKA was used for attribute selection. The selection process was carried out for both sliding windows, three and five, as well as for each p-value threshold ranging from 1e-02 to 1e-10. Of the 14 features, the top seven attributes for each combination were chosen and used to train four classifiers. A threshold of ≥ 0.85 (on a scale of 0 to 1, with 1 indicating the most information contributed to a given classification outcome) was used to determine which attributes were chosen with the intent of adjusting the number of attributes utilized depending on the accuracies of the classifiers. The most common combination of attributes selected were the temporal intervals 7-10 (corresponding to the 42 -60 second time intervals), ρ_s, median, and the categorical ρ_s variable.

Table 14 shows the features chosen during attribute selection using the information gain algorithm and a sliding window of five at two different p-value thresholds. We see that the same seven attributes were selected. However, the rank orders varied depending on the amount of information each feature contributed for a given p-value threshold. Overall, the information contributed for the attributes at a p-value threshold of 1e-10 and than those at a p-value of 1 e-09 are similar.

Table 14: Top seven attributes selected using information gain and a sliding window of five at p-value thresholds of 1e-09 and 1e-10.

P-value 1e-09		**P-value 1e-10**	
Attributes Selected	**Rank**	**Attributes Selected**	**Rank**
Interval 10	0.964	Interval 10	0.953
Interval 7	0.929	Interval 7	0.931
Interval 9	0.925	Median	0.914
ρ_s	0.919	Interval 9	0.911
Interval 8	0.912	Interval 8	0.908
Median	0.902	ρ_s	0.893
ρ_s Category	0.871	ρ_s Category	0.878

WEKA offers the option to visualize some of the classification model architectures post classifier training. This allows the user to see an overview of the model as well as which features are being used to achieve classification. Figures 22 and 23 show the models generated after training each of the decision tree classifiers using a sliding window of five and the information gain algorithm at both p-value thresholds. We can see that the same features are used in both models however the order varies as well as the numerical thresholds used at a given node.

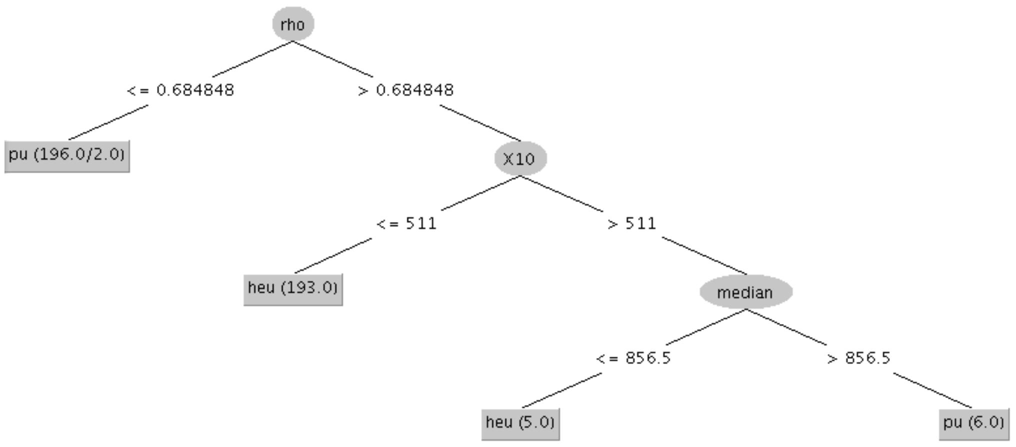


Figure 22: Tree model at a p-value threshold of 1e-10 using information gain for attribute selection.

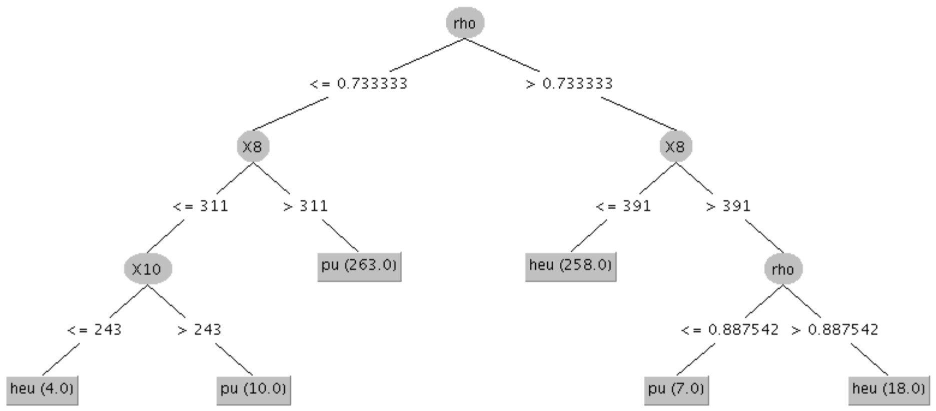


Figure 23: Tree model at a p-value threshold of 1e-09 using information gain for attribute selection.

Figure 24 shows a neural network model generated post training at a p-value threshold of 1e-09. The input features are seen in the green boxes on the left hand side and the classification outputs on the right. The red circles indicate eight nodes of the single hidden layer present in the model.

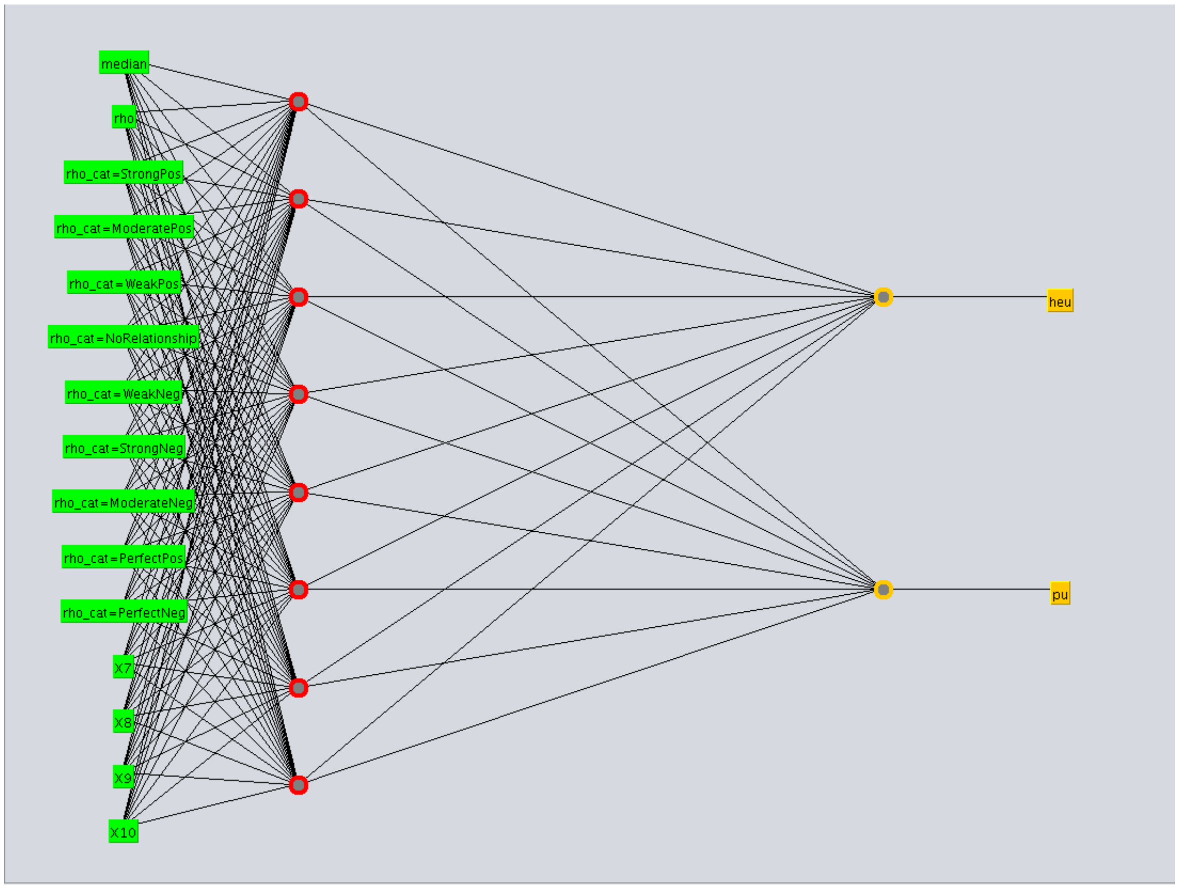


Figure 24: Neural network model at a p-value threshold of 1e-09 using information gain for attribute selection.

Figure 25 shows classifier performance for each p-value threshold using the Welch's t-test. We can see in these plots that as the p-value decreases the accuracy of a given classifier, regardless of the sliding window, increased. In addition, we see that starting at a p-value of 1e-07 the sliding window of five gives consistently higher accuracy compared to a sliding window of three. For a sliding window of three, the performance trend increases as the p-value decreases; however, the increased performance appears less steady when compared to a sliding window of 5, particularly starting at a p-value of 1e-07.

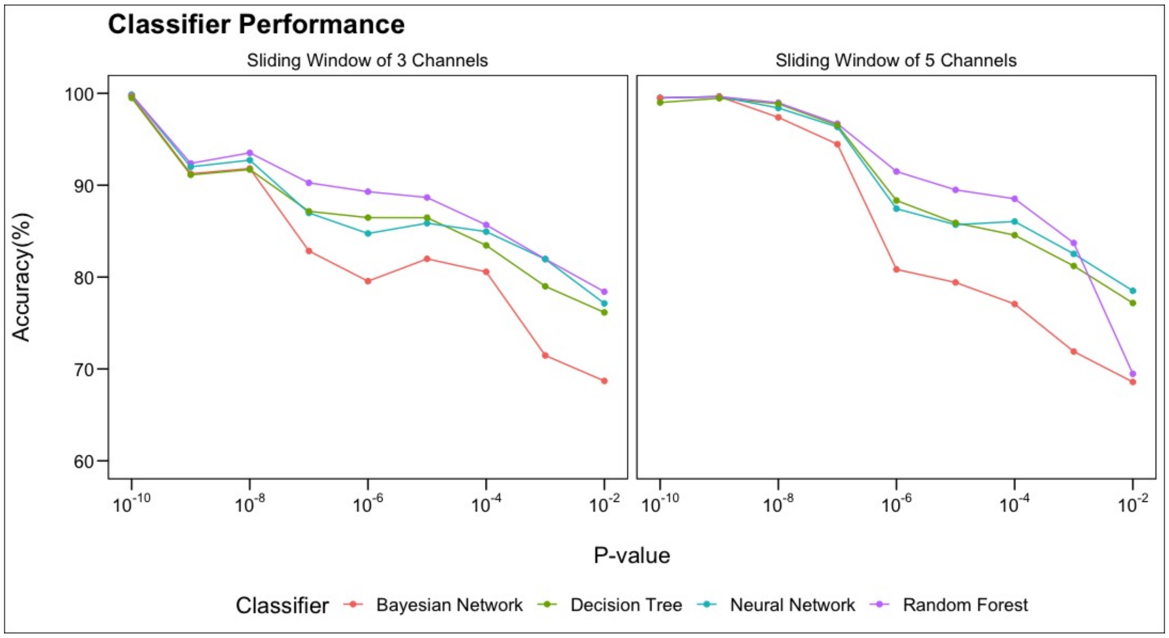


Figure 25: Plot showing p-value vs accuracy for four different classifiers and both sliding windows. Welch's t-test was used to compute p-values.

Further, we see fluctuations in classifier accuracy across p-value thresholds. Looking at the data for a sliding window of five we see the fluctuations begin to even out. However, we observe a small drop in classifier performance moving from a p-value of 1e-09 to a p-value of 1e-10 for the random forest, neural network and decision tree classifiers. For a sliding window of three, we see a similar phenomenon but from a p-value of 1e-08 to a p-value of 1e-09. However, this is then followed by a fairly large increase in accuracy at a p-value of 1e-10. Fluctuations in accuracy from one p-value threshold to another may be due to a combination of factors. One potential factor is the amount of information the classifier is shown for training. For example, the number of channels used for training at a p-value threshold of 1e-09 is greater than the number used at a p-value threshold of 1e-10. Having too few or too many channels (information) for a given classifier can impact the performance. Classifier performance can be further effected by other parameters such as the attributes selected as well as the inherent parameters for a given classifier.

Tables 15 and 16 display the accuracies of all four trained classifiers across all p-value thresholds for both sliding windows. This allows for a more in-depth look of the accuracy differences occurring at the smaller p-value thresholds. Looking at Table 15, we see that the Bayesian network shows the lowest classifier accuracy until a p-value threshold of 1e-08 at which point the lowest performing classifier is the decision tree. However, the differences are quite small. The neural network and random forest classifiers show similar trends in accuracy specifically at the lower p-values. All four classifiers show a high level of accuracy starting at a p-value of 1e-08. It appears that both sliding windows achieve similar accuracies, particularly at the lowest p-value thresholds of 1e-09 and 1e-10 . Given that the accuracy trends for a sliding window of five appear more consistent starting at a p-value of 1e-07, a sliding window of five was chosen for further exploration.

Table 15: Classifier accuracy based on p-value for a sliding window of three using information gain for attribute selection. The top seven attributes were used when training each classifier. The Welch's t-test was used to compute p-values.

P-value	**Random Forest**	**Decision Tree**	**Neural Network**	**Bayesian Network**
1e-02	78.41	76.15	77.14	68.70
1e-03	81.97	79.00	81.95	71.47
1e-04	85.68	83.45	84.95	80.57
1e-05	88.66	86.47	85.86	81.98
1e-06	89.29	86.47	84.74	79.55
1e-07	90.26	87.16	86.98	82.84
1e-08	93.52	91.70	92.73	91.82
1e-09	92.38	91.13	92.00	91.25
1e-10	99.83	99.50	99.83	99.67

Table 16: Classifier accuracy based on p-value for a sliding window of five using information gain for attribute selection. The top seven attributes were used when training each classifier. The Welch's t-test was used to compute p-values.

P-value	**Random Forest**	**Decision Tree**	**Neural Network**	**Bayesian Network**
1e-02	69.47	77.17	78.51	68.58
1e-03	83.71	81.21	82.53	71.89
1e-04	88.52	84.57	86.05	77.07
1e-05	89.49	85.90	85.70	79.41
1e-06	91.49	88.33	87.44	80.83
1e-07	96.70	96.52	96.34	94.46
1e-08	98.98	98.86	98.41	97.39
1e-09	99.64	99.46	99.64	99.64
1e-10	99.50	99.00	99.50	99.50

Evaluating the energy ranges for the channels of a sliding window of five we see that the energy difference is 1.720 keV which is in agreement with the resolution of the HPGe detector of 1.7 keV at 1332 keV as seen in Table 17. Using a sliding window of five ensured that we covered the resolution of the detector and were not analyzing a range of channels at a higher resolution than what the detector can achieve. In addition, we see that the channels identified are clustered together consecutively. At lower energies a given peak can span 7-10 channels, the consecutive channels identified coincide with spanning a given peak at the lower energies. It is important to note that the evaluation of channel energy was performed post data pre-processing and model selection as a method for validating the choice of a sliding window of five and was not necessary for identifying or selecting significant channels for classification or mass estimation.

Table 17: Channel and energy ranges for a sliding window of five at a p-value threshold of 1e-09 identified by the Welch's t-test. The energy range for each channel spans 1.720 keV.

Channel	**Channel Range**	**Energy Range (keV)**	**Energy Difference (keV)**
138	136-140	58.48 - 60.20	1.720
139	137-141	58.91-60.63	1.720
140	138-142	59.34-61.06	1.720
141	139-143	59.77-61.49	1.720
232	230-234	98.81-100.5	1.720
301	299-303	128.4-130.1	1.720
302	300-304	128.8-130.6	1.720
303	301-305	129.3-131.0	1.720
873	871-875	373.8-375.5	1.720
874	872-876	374.2-375.9	1.720
875	873-877	374.6-376.4	1.720
963	961-965	412.4-414.1	1.720
964	962-966	412.8-414.5	1.720
965	963-967	413.2-414.9	1.720

7.3 Classifier Validation

Validation requires new, unseen data be used as input to trained classifiers to evaluate the accuracy on unlabeled data. Data collected in 2018 was used for validation. An energy calibration was performed prior to data collection. The data consisted of a total of 38 cycles, 19 from Pu-239 and 19 from HEU. All four trained classifiers were presented with unlabeled data from each of the 38 cycles. Tables 18 to 25 display the accuracies of each classifier at both p-value thresholds, 1e-09 and 1e-10. The number of channels (instances) shown to each classifier at a p-value threshold of 1e-09 and 1e-10 were 14 and 10, respectively.

Table 18: Random forest classifier performance using a p-value threshold of 1e-09. On average, the Random forest classifier at a p-value threshold of 1e-09 showed 99.6% accuracy in classifying HEU and 98.9% accuracy in classifying Pu-239.

"Unknown" Data Set	**HEU (%)**	**Pu-239 (%)**
1	100	92.9
2	100	92.9
3	100	100
4	100	100
5	100	100
6	100	100
7	100	100
8	100	100
9	100	100
10	100	92.9
11	100	100
12	100	100
13	100	100
14	100	100
15	100	100
16	92.9	100
17	100	100
18	100	100
19	100	100

Table 19: Neural network classifier performance using a p-value threshold of 1e-09. On average, the neural network classifier at a p-value threshold of 1e-09 showed 98.5% accuracy in classifying HEU and 99.6% accuracy in classifying Pu-239.

“Unknown” Data Set	**HEU** (%)	**Pu-239** (%)
1	100	92.9
2	100	100
3	100	100
4	92.9	100
5	92.9	100
6	100	100
7	100	100
8	92.9	100
9	100	100
10	92.9	100
11	100	100
12	100	100
13	100	100
14	100	100
15	100	100
16	100	100
17	100	100
18	100	100
19	100	100

Table 20: Decision tree classifier performance using a p-value threshold of 1e-09. On average, the decision tree classifier at a p-value threshold of 1e-09 showed 99.6% accuracy in classifying both HEU and Pu-239.

“Unknown” Data Set	**HEU (%)**	**Pu-239 (%)**
1	100	100
2	100	92.9
3	100	100
4	100	100
5	100	100
6	100	100
7	100	100
8	100	100
9	92.9	100
10	100	100
11	100	100
12	100	100
13	100	100
14	100	100
15	100	100
16	100	100
17	100	100
18	100	100
19	100	100

Table 21: Bayesian network classifier performance using a p-value threshold of 1e-09. On average, the Bayesian network classifier at a p-value threshold of 1e-09 showed 98.5% accuracy in classifying HEU and 100% accuracy in classifying Pu-239.

"Unknown" Data Set	**HEU (%)**	**Pu-239 (%)**
1	100	100
2	92.9	100
3	92.9	100
4	100	100
5	100	100
6	100	100
7	100	100
8	92.9	100
9	100	100
10	100	100
11	92.9	100
12	100	100
13	100	100
14	100	100
15	100	100
16	100	100
17	100	100
18	100	100
19	100	100

Table 22: Random forest classifier performance using a p-value threshold of 1e-10. On average, the Random forest classifier at a p-value threshold of 1e-10 showed 98.4% accuracy in classifying HEU and 98.9% accuracy in classifying Pu-239.

"Unknown" Data Set	**HEU (%)**	**Pu-239 (%)**
1	100	90
2	100	90
3	100	100
4	100	100
5	100	100
6	100	100
7	100	100
8	100	100
9	100	100
10	100	100
11	100	100
12	100	100
13	100	100
14	100	100
15	100	100
16	70	100
17	100	100
18	100	100
19	100	100

Table 23: Neural network classifier performance using a p-value threshold of 1e-10. On average, the neural network classifier at a p-value threshold of 1e-10 showed 97.9% accuracy in classifying HEU and 99.5% accuracy in classifying Pu-239.

"Unknown" Data Set	**HEU (%)**	**Pu-239 (%)**
1	100	90
2	100	100
3	100	100
4	90	100
5	90	100
6	100	100
7	100	100
8	90	100
9	100	100
10	90	100
11	100	100
12	100	100
13	100	100
14	100	100
15	100	100
16	100	100
17	100	100
18	100	100
19	100	100

Table 24: Decision tree classifier performance using a p-value threshold of 1e-10. On average, the Decision Tree classifier at a p-value threshold of 1e-10 showed 100% accuracy in classifying HEU and 98.9% accuracy in classifying Pu-239. We also see some of the lowest accuracies we have seen occur for HEU. For example, data set 16 at 40%.

"Unknown" Data Set	**HEU (%)**	**Pu-239 (%)**
1	100	80
2	100	90
3	100	100
4	100	100
5	100	100
6	70	100
7	100	100
8	100	100
9	100	100
10	100	100
11	80	100
12	100	100
13	100	100
14	100	100
15	100	100
16	40	100
17	100	100
18	100	100
19	100	100

Table 25: Bayesian network classifier performance using a p-value threshold of 1e-10. On average, the Bayesian network classifier at a p-value threshold of 1e-10 showed 97.4% accuracy in classifying HEU and 99.5% accuracy in classifying Pu-239.

"Unknown" Data Set	**HEU** (%)	**Pu-239** (%)
1	100	100
2	90	100
3	90	100
4	100	100
5	100	100
6	90	100
7	100	90
8	90	100
9	100	100
10	100	100
11	90	100
12	100	100
13	100	100
14	100	100
15	100	100
16	100	100
17	100	100
18	100	100
19	100	100

We see that all four classifiers classified both fissile materials with high accuracy at both p-value thresholds. Tables 26 and 27 show a comparison of the accuracy during training and the average validation accuracy of all four classifiers at both p-value thresholds for both fissile materials. The average validation accuracy for each material was computed using the accuracies for each of the 19 cycles shown to each classifier.

Table 26: Comparison of classifier accuracy during training and validation at p-value thresholds of 1e-09. Validation accuracies are presented as the average accuracy across the 19 data sets tested.

Classifier	**Training Accuracy**	**HEU Validation Accuracy**	**Pu-239 Validation Accuracy**
Random Forest	99.6	99.6	98.8
Neural Network	99.6	98.5	99.6
Decision Tree	99.6	99.6	99.6
Bayesian Network	99.5	98.5	100

Table 27: Comparison of classifier accuracy during training and validation at p-value thresholds of 1e-10. Validation accuracies are presented as the average accuracy across the 19 data sets tested.

Classifier	**Training Accuracy**	**HEU Validation Accuracy**	**Pu-239 Validation Accuracy**
Random Forest	99.5	98.4	98.9
Neural Network	99.5	97.9	99.5
Decision Tree	99.5	94.2	98.4
Bayesian Network	99.0	97.4	99.5

Overall, the accuracy achieved during validation was in good agreement with the accuracy achieved during training. However, we do see some instances where the accuracy during validation was slightly less than the classifier accuracy reported during training. The most significant difference occurred for the decision tree accuracy for HEU at a p-value threshold of 1e-10. The validation accuracy for HEU was 94.2% compared to the training accuracy of 99.5%. Comparing this result to the decision tree accuracy for HEU at a p-value of 1e-09, we see that HEU validation accuracy was equal to the training accuracy. All other accuracies at both p-value thresholds were within 1% of the trained classifiers accuracies. Given how similar the results were for both p-value thresholds, the channels identified at a p-value threshold of 1e-09 were used for mass estimation. This allowed more channels to be evaluated using the multidimensional analysis.

7.4 Relative Mass Content Estimation

Significant channels determined during data pre-processing were used to determine the relative mass content of fissile material. The goal was to see if channels identified without prior knowledge of energy below 3 MeV could be used for mass estimation to potentially improve the accuracy and precision of estimates. For proof of concept, we looked at pure samples of HEU and Pu-239. Tables 28 and 29 show the results for each of the 14 channels that were identified as displaying significant differences between fissile materials. Estimates are presented as percents and the percent error is presented at the 68% and 95% confidence levels.

Table 28: Relative mass content estimates for HEU and Pu-239 for the 14 channels identified as significant during data pre-processing. Mass estimates are based on evaluating an “unknown” sample of HEU. All estimates are presented as percents. Error is presented at the 68% and 95% confidence levels.

Channel	**Estimated HEU**	**Estimated Pu-239**	**Error (1σ)**	**Error (2σ)**
138	100.64	-0.64061	0.07885	0.15770
139	101.09	-1.0935	0.79018	1.5804
140	98.812	1.1880	0.76867	1.5373
141	99.731	0.26939	0.83286	1.6657
232	92.861	7.1395	0.55170	1.1034
301	97.808	2.1923	0.71986	1.4397
302	98.444	1.5560	0.72103	1.4421
303	96.466	3.5340	0.71563	1.4313
873	97.276	0.72376	1.0808	2.1615
874	97.698	2.3025	1.0138	2.0277
875	99.003	0.99749	1.0457	2.0914
963	105.68	-5.6771	1.3392	2.6784
964	104.70	-4.7009	1.2639	2.5278
965	101.96	-1.9646	1.1943	2.3885

Table 29: Relative mass content estimates for HEU and Pu-239 for the 14 channels identified as significant during data pre-processing. Mass estimates are based on evaluating an "unknown" sample of Pu-239. All estimates are presented as percents. The error is presented at the 68% and 95% confidence levels.

Channel	**Estimated HEU**	**Estimated Pu-239**	**Error (1σ)**	**Error (2σ)**
138	-0.92453	100.92	1.4319	2.8638
139	-3.2730	103.27	0.64640	1.2928
140	3.4427	96.557	0.54374	1.0875
141	2.3324	97.668	0.50482	1.0096
232	-22.722	122.72	0.44367	0.88734
301	-10.625	110.63	0.63469	1.2694
302	-12.539	112.54	0.64552	1.2910
303	-15.118	115.12	0.64376	1.2875
873	2.3278	97.672	0.64848	1.2970
874	5.3178	94.682	0.66125	1.3225
875	1.6897	98.310	0.69310	1.3862
963	8.3016	91.698	0.67074	1.3415
964	11.054	88.946	0.70471	1.4094
965	14.157	85.843	0.67243	1.3449

Figures 26, 27, 28 and 29 show each channels estimate for both materials and their associated errors at the 95% confidence level. At the 95 % confidence level, many of the channels showed relative mass estimates close to the relative known masses.

There were seven channels that showed accurate relative mass estimates for at least one of the materials. Overall, the HEU relative mass estimates were accurate at the 95% confidence level for 43% of the channels. The Pu-239 relative mass estimates underestimated the relative mass at the 95% confidence level for 57% of the channels. On average, the Pu-239 relative mass estimates underestimated by 8%.

Figure 26 shows mass estimates for channels 138-141 at the 95% confidence level. Channels 139-141 show accurate relative mass estimates for HEU. However, for the same channels, the Pu-239 relative mass estimates were not in agreement with the Pu-239 known relative mass. Pu-239 showed an accurate relative mass estimate for channel 138. Channels 138 and 141 appear to show promise as the relative mass estimate is accurate for one material and particularly close for the other.

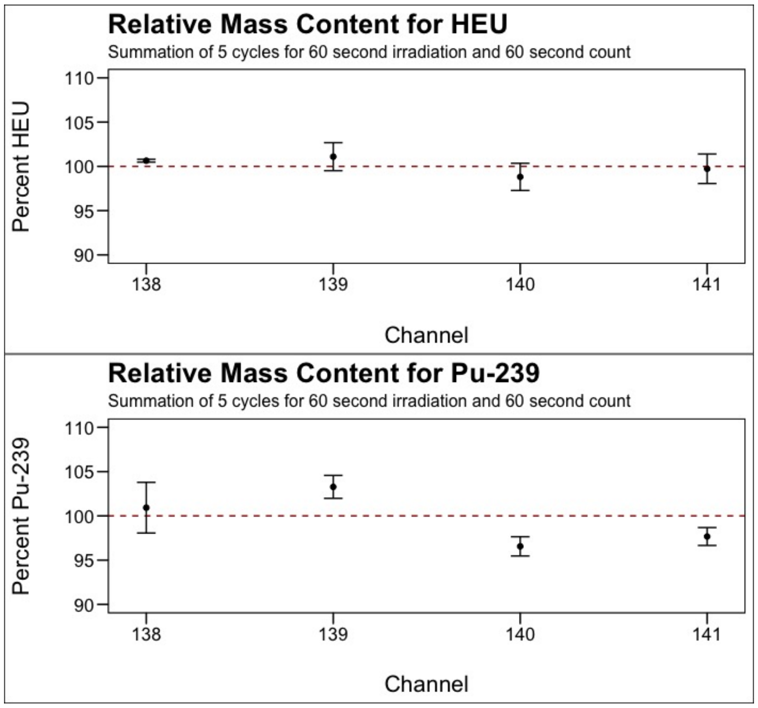


Figure 26: Relative mass content for HEU and Pu-239 for channels 138, 139, 140 and 141 at the 95% confidence level.

Figure 27 shows mass estimates for channels 232 and 301-303 at the 95% confidence level. The relative mass estimates for HEU channels 301-302 were particularly close. However, the relative mass estimates for Pu-239 were overestimated for all four channels. Overall, the relative mass content was underestimated for HEU and overestimated for Pu-239 in these channels.

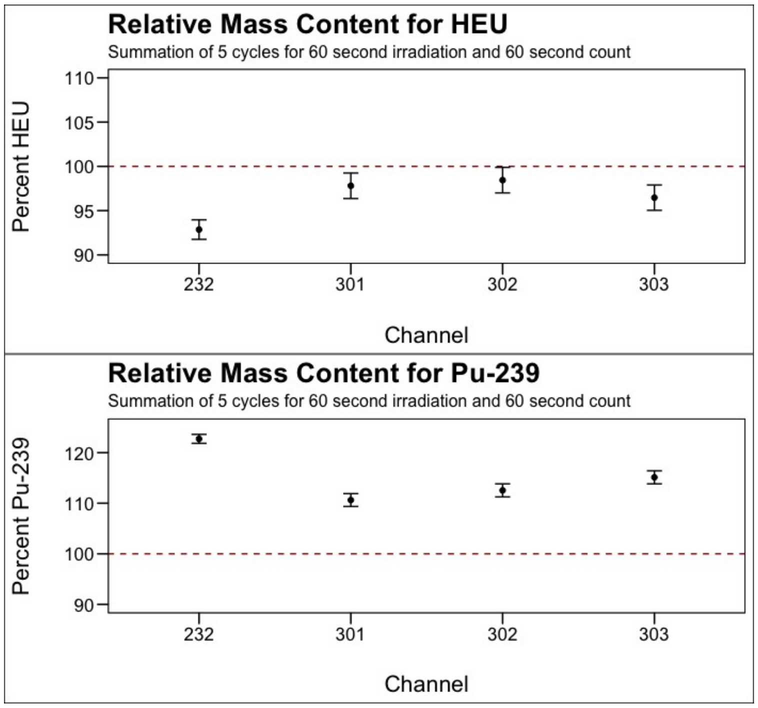


Figure 27: Relative mass content for HEU and Pu-239 for channels 232, 300, 301, 302 and 303 at the 95% confidence level.

Figure 28 shows the relative mass estimates for channels 873-875 at the 95% confidence level. Channel 875 shows an accurate mass estimate for HEU and the relative mass estimates for channels 873 and 874 are quite close to the relative known HEU mass. The Pu-239 relative mass estimate for channel 875 is very close to the Pu-239 known relative mass. However, the Pu-239 relative mass estimates tended to be underestimated for these channels.

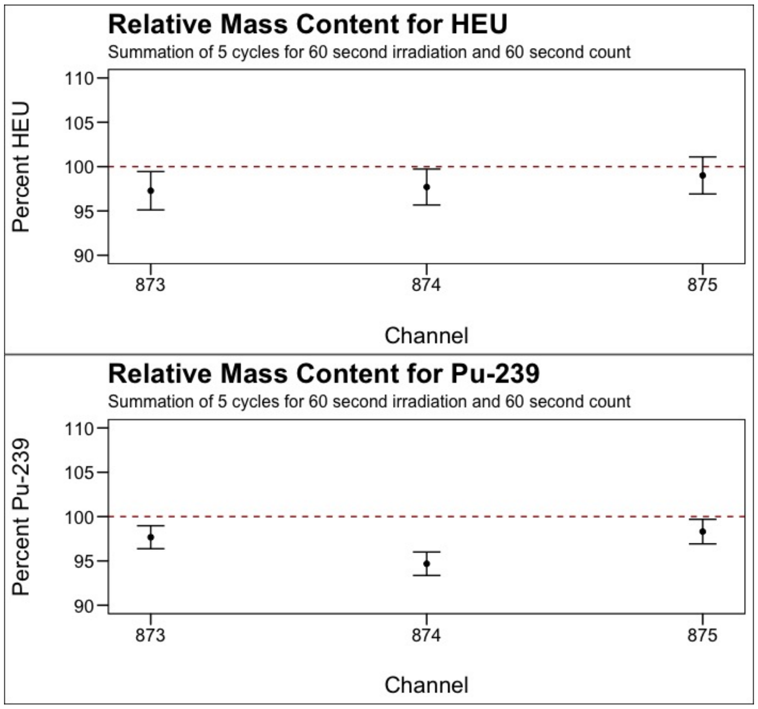


Figure 28: Relative mass content for HEU and Pu-239 for channels 873, 874 and 875 at the 95% confidence level.

Figure 29 shows the relative mass estimates for channels 963-965 at the 95% confidence level. Channel 965 shows a mass estimate in agreement with the relative known mass of HEU. However, for channels 963 and 964 the relative mass estimates were overestimated for HEU. The relative mass estimates for Pu-239 were underestimated in all three channels.

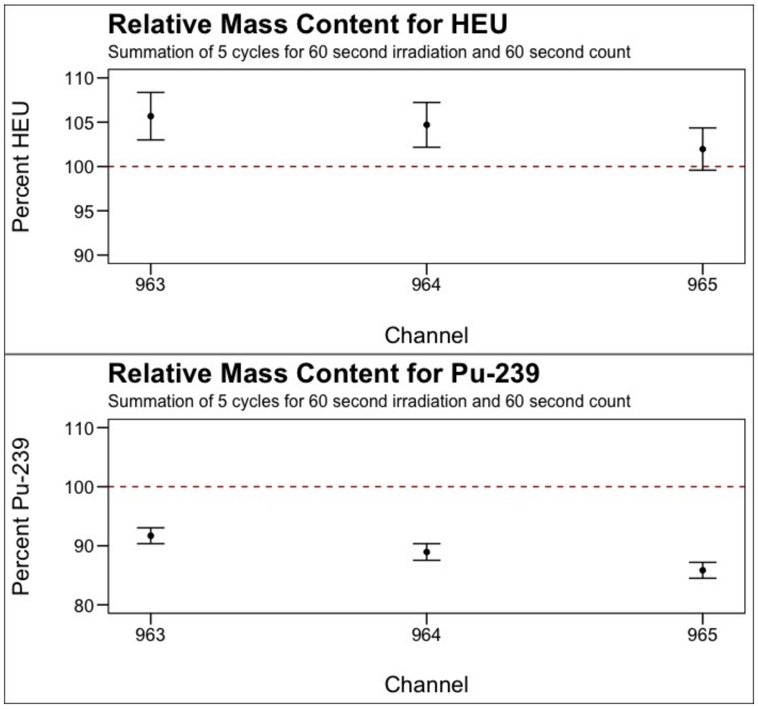


Figure 29: Relative mass content for HEU and Pu-239 for channels 963, 964, 965 and 966 at the 95% confidence level.

Further exploration is required to investigate ways to improve mass estimates. While these channels have shown to be successful for fissile material classification, they have shown less than desirable results for determining relative mass content. However, many of the channels show promise given accurate mass results for one of the fissile materials or very close estimates in other cases.

Previous work by Mannino showed that using the natural abundances of isotopes in the weight matrix improved relative mass estimates [38]. While the natural abundance of U-235 in our HEU sample is easily calculated given that we know the enrichment of the sample, we do not have the same information regarding the abundance of Pu-239 and Pu-240 in our Pu-239 sample. In addition, it is possible that the results are due to the small sample sizes of Pu-239 and HEU. Previous work by

Williford used samples much larger in size. The samples used in this work were 244 mg and 178 mg for Pu-239 and HEU, respectively. In comparison, the samples used in Williford's work were 2.92 g and 2.80 g for Pu-239 and U-235, respectively. One way to increase the number of counts (without altering sample size) is to increase the flux which the sample is exposed to. The TRIGA reactor at Oregon State University has an in-core rabbit system that could be utilized for this analysis, exposing the current samples to a a higher flux and therefore increasing the total number of counts used for analysis.

7.5 Peak Analysis

An analysis was performed post-classification and mass estimation to evaluate the energies associated with the channels utilized in each phase. It was found that based on the p-value thresholds of 1e-09 and 1e-10, fission product peaks were not used for classification nor mass estimation. Rather, the peaks used were x-ray and gamma-ray peaks from the natural decay of Pu-239 and U-235. However, further analysis of the other p-value thresholds showed that fission product peaks were identified as significant during data pre-processing. These additional peaks identified at the larger p-value thresholds provide additional channels to use for mass content estimation and may lead to improved mass estimates in the future. At the largest p-value threshold (1e-02), 460 channels were identified.

Many of the fission product peaks had lower counts than the peaks used for classification and mass estimation. While the peaks presented in this work were useful in providing accurate classification of fissile material, they did not provide accurate mass estimate results for both fissile materials. As mentioned previously, the small sample sizes are one potential reason for the less than desirable relative mass content estimates. Therefore, exploring other channels at the larger p-value thresholds would not necessarily impact the accuracy of the mass estimates in this current work. However, it would be beneficial to look at these additional peaks after exploring additional irradiation/counting protocols as well as methods for increasing the overall counts.

Further, a single set of channels may not be ideal to achieve the desired accuracy for each phase. Determining two different subsets of channels, one for classification and one for relative mass content estimation, during data pre-processing may be necessary. Perhaps performing an analysis on the p-value threshold as a method for identifying a subset of channels specifically for relative mass content estimation would prove to be successful in the future.

8 Conclusions

This research describes a novel process for successful classification of Pu-239 and HEU using machine learning and temporal spectroscopy methods. To date, methods have included peak energy and peak search procedures. Specifically, work by Sullivan et al. involved using a library of energies while Yoshida et al. used peak search procedures based on peak energies of interest [57, 67]. In this research, prior knowledge of specific peak energies which correlate to known isotopes of interest were not utilized. Rather, through data pre-processing, data was transformed using statistical methods and feature generation to describe meaningful patterns present in the spectra. Rather than letting known isotope energies drive the data mining process, the data spoke for itself. Following energy calibration, channels of interest found during data pre-processing were then used to filter raw, unlabeled data for classification.

In addition, four classifiers were found to provide high accuracy in the classification of fissile material. Through the use of a sliding window of five, the information gain algorithm for feature selection and Welch's t-test to identify channels of interest; the random forest, decision tree, neural network and Bayesian network classifiers were successfully trained for future classification of Pu-239 and HEU. All four classifiers achieved similar accuracy, 98-99 % accuracy for Pu-239 and 99-100 % for HEU. However, at a p-value threshold of 1e-09, the decision tree classifier achieved the highest accuracy of 99.6 % for both fissile materials. The feature set utilized for the decision tree consisted of ρ_s score as well as the $8^{th}$and 10^{th} time intervals.

Further, regions of interest below 3 MeV were identified and used for mass estimation of fissile material. Previous work by Williford utilized temporal spectroscopy methods to perform relative mass content estimation on regions of interest greater than 3 MeV [64]. In this work, machine learning methods were employed which resulted in additional regions of interest to be used for mass estimation. Combining temporal spectroscopy with machine learning methods allows the entire spectrum to be utilized in the quantification of mass content. Machine learning limits the need for user intervention and expert knowledge once the model has been trained [48] allowing for an automated process that is easily implemented into active interrogation methods for nonproliferation and nuclear security.

While accurate mass content estimates were not obtained for both fissile materials, many of the channels identified showed promise. Using the natural abundances of the materials of interest may improve mass estimates. Further, collecting additional data to increase the overall number of counts being analyzed has the potential to improve the accuracy of the observed estimates.

9 Future Work

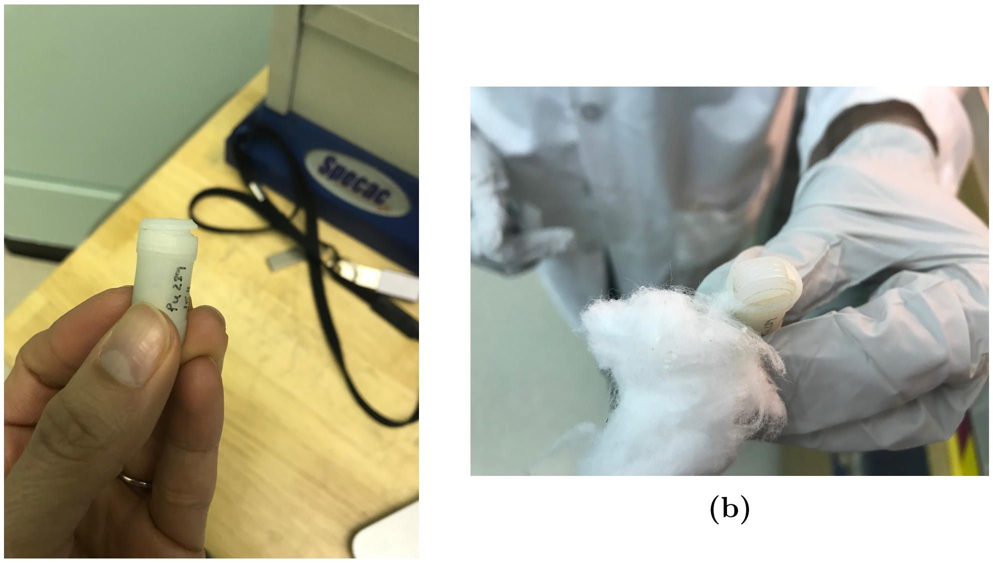
This work served as a proof of concept for both the classification and relative mass content estimation of fissile material with the goal of evaluating channels below 3 MeV. While the results for classification were highly successful there are several methods that could be implemented to improve the mass estimates and overcome some of the limitations of the current work. The following are some suggestions of areas to explore for future work.

Perhaps one of the biggest limitations and contributing factors to the mass estimation results is the lack of counts present in the data given the small sample sizes of fissile material available. One potential solution is to use the in-core rabbit system at the TRIGA reactor at Oregon State University. Increasing the incident flux on the sample would increase the overall counts and may lead to more accurate mass estimates. In the current work, the goal was to evaluate channels below 3 MeV. However, it is also noted that channels corresponding to an energy greater than 1.2 MeV lacked enough counts to be evaluated in this work. While this method is viable for the evaluation of the entire spectrum, without increased counts it is not feasible.

It is also of interest to explore different time binning options. Williford showed that mass estimate accuracy can be impacted based on the time bin used. He noted that with too small of a time bin the count variation was too large yielding poorer mass estimate accuracy. He also noted that mass estimate accuracy decreased with too large of a time bin due to a dampening of the temporal behavior in a particular region of interest. In this work, six-second time bin intervals were used for proof of concept. Evaluating various binning options would be beneficial. However, the true benefit, if there is one, would be discovered after increasing the number of counts.

Further, while fission product peaks were identified during data pre-processing at larger p-value thresholds, classification accuracy improved dramatically using smaller p-value thresholds and incidentally eliminated the fission product peaks. Channels identified during data pre-processing were utilized for both phases of this work, classification and relative mass content estimation. However, this is not a requirement. A separate method could be implemented during data pre-processing for identifying other channels to use for mass estimation. For example, we saw in this work that many other channels were identified at other p-value thresholds during data pre-processing. Finding an optimal p-value threshold for identifying channels of interest specifically for mass content estimation could be implemented in the future. However, as mentioned previously, without improving counting statistics it is unlikely that mass estimates will improve with the analysis of additional channels. Therefore, it is important that the analysis be carried out in combination with methods for increasing counts.

Due to the presence of longer-lived isotopes following fission, the current data shows buildup across cycles. Incorporating a rest protocol would minimize the effects of buildup, while it will never fully eliminate it. An attempt to collect new data and incorporate a rest protocol was performed; however, significant issues were encountered. The sample capsules failed, leading to interruptions in irradiation/cycle sessions making it difficult to ensure consistent rest periods between irradiation and counting cycles. Further, the optical sensors would often become misaligned and turn off during sample transit leading to inaccurate flagging of the sample. It is suggested that updates to the current system be performed prior to further data collection. Figures 30*a* and 30*b* show some of the sample capsule issues encountered during data collection.



(a) **(b)**

Figure 30: Capsule failures for the Pu-239 sample (a) partial capsule failure due to lid unscrewing during transit and (b) complete capsule failure due to lid unscrewing and becoming lodge in the capsule body during transit.

This work focused on looking at pure samples of HEU and Pu-239. However, in practice, this method would likely be used for evaluating many fissionable materials. The current classification methods allow for training classifiers on many different isotopes as well as mixed samples. To address mixed samples for Pu-239 and HEU specifically, several data sets would need to be pre-processed for model training. These data sets should consist of various percentage combinations of fissile material. Once the classifiers are trained, each needs to be validated with raw, unlabeled data. It is possible that a library of classifiers for various isotopes of interest and various mixtures could be trained and utilized in the field.

Further, the current multidimensional analysis allows for the evaluation of multiple isotopes for mass estimation. An important consideration if expanding to other fissile materials is the need to account for the difference in cross sections. In the current work, focus was on pure samples with known sample masses. Therefore, little inference was needed in regards to the mass estimations. However, if the multidimensional anlaysis were to be used on sample mixtures, particularly mixtures containing three or more fissile materials, a weighting function derived from the difference in cross section would likely need to be applied for accurate mass estimation.

Given the large number of learning algorithms available for classification problems, such as classifying special nuclear material, it would be interesting to implement additional machine learning methods. For example, in the literature, support vector machines have been used for gamma-ray spectroscopy. While used for a different purpose other than what is outlined in this research, they are a logical option to consider as an additional classifier to investigate.

In addition, looking into options for further feature generation could be beneficial. While the features generated in this research proved to provide accurate classification, additional features may be beneficial. Finally, comparing different attribute selection algorithms would be useful. WEKA provides many algorithms to choose from and there may be a more optimal feature set identified using alternate selection methods.

References

[1] R.E. Abdel-Aal and M.N. Al-Haddad. Determination of radioisotopes in gamma-ray spectroscopy using abductive machine learning. *Nuclear Instruments and Methods in Physics Research A*, 391:275–288, 1997.

[2] Matt J. Aitkenhead, Mark Owen, and David M. Chambers. Use of artificial neural network in measuring characteristics of shielded plutonium for arms control. *J. Anal. At. Spectrom*, 27:432–439, 2012.

[3] Serkan Akkoyun. Time-of-flight discrimination between gamma-rays and neutrons by neural networks. Technical report, Cumhuriyet University, Unknown.

[4] T. Anderson. *Oregon State TRIGA Reactor Training Manual Volume 1*. Oregon State University, 2010.

[5] Luciana Ferrer Andreas Stolcke, Sachin Kajarekar. Nonparametric feature normalization for svm-based speaker verification. *IEEE International Conference on Accoustics, Speech and Signal Processing*, 2008.

[6] D.H. Beddingfield and F.E. Cecil. Identification of fissile materials from fission product gamma-ray spectra. *Nuclear Instruments and Methods in Physics Research A*, 417:405–412, 1998.

[7] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[8] Charles Henry Brase and Corrinne Pellillo Brase. *Understandable Statistics: concepts and methods*. Richard Stratton, 1983.

[9] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

[10] Leo Breiman. Statistical modeling: The two cultures. *Statistical Science*, 16(3):199–215, 2001.

[11] W. Charlton. Next generation safeguards initiative spent fuel effort external review committee: Final report. Technical report, Washington: National Nuclear Security Administration, 2011.

[12] D.H. Chivers, K. Alfonso, B.L. Goldblum, and B. Ludewigt. Novel methodology for the quantitative assay of fissile materials using temporal and spectral beta-delayed gamma-ray signatures. *Nuclear Instruments and Methods in Physics Research B*, 269:1829–1835, 2011.

[13] C.T. Cunningham. Bayesian spectroscopy and target tracking. In *7th International Conference on Applications of Nuclear Techniques*. Lawrence Livermore National Laboratory, 2001.

[14] Siddhartha R. Dalal and Bing Han. Detection of radioactive material entering national ports: A bayesian approach to radiation portal data. *The Annals of Applied Statistics*, 4(3):1256–1271, 2010.

[15] Barry de Ville. Decision trees. *WIREs Computational Statistics*, 5, 2013.

[16] S. Dragović, A. Onjia, S. Stanković, I. Aničin, and G. Bačić. Artificial neural network modelling of uncertainty in gamma-ray spectrometry. *Nuclear Instruments and Methods in Physics Research A*, 540:455–463, 2005.

[17] T.R. England and B. Rider. Evaluation and compilation of fission product yields. Technical report, Los Alamos National Laboratory, 1994.

[18] R. B. Firestone, G. A. English, J. Reijonen, F. Gicquel, K-N. Leung, D. L. Perry, G. Garabedian, B. Bandong, Zs. Révay, and G. L. Molnár. Analysis of fissile materials by high-energy neutron-induced fission decay gamma-rays. *Journal of Radioanalytical and Nuclear Chemistry*, 265(2):241–245, 2005.

[19] Erwan Scornet Gérad Biau. A random forest guided tour. *IEEE Transactions on pattern analysis and machine intelligence*, PAMI-2(3):242–252, 1980.

[20] Arnaud Giacometti, Eynollah Khanjari Miyaneh, Patrick Marcel, and Arnaud Soulet. A generic framework for rule-based classification. Technical report, Universite Francois Rabelais de Tours, Unknown.

[21] Tsahi Gozani. Active nondestructive assay of nuclear materials principles and applicatins. Technical report, U.S. Nuclear Regulatory Commission, 1981.

[22] Tsahi Gozani. Fission signatures for nuclear material detection. *IEEE Transactions on nuclear science*, 56(3), 2009.

[23] J.M. Hall, S. Asztalos, P. Biltoft, J. Church, M.-A. Descalle, T. Luu, D. Manatt, G. Mauger, E. Norman, D. Petersen, J. Pruet, S. Prussin, and D. Slaughter. The nuclear car wash: Neutron interrogation of cargo containers to detect hidden snm. *Nuclear Instruments and Methods in Physics Research B*, 261:337–340, 2007.

[24] Mark Hall, Eibe Frank, Geoffrey Holmes, Bernhard Pfahringer, Peter Reutemann, and Ian H. Witten. The weka data mining software: An update. Blog, 2009.

[25] Jiawei Han, Micheline Kamber, and Jian Pei. *Data Mining Concepts and Techniques*. Elsevier, third edition, 2012.

[26] D.J. Hand. Principles of data mining. In *Drug Safety*, volume 30, pages 621–622, 2007.

[27] Haruhi Hata, Kaoru Yokoyama, Yuu Ishimori, Yoshiyuki Ohara, Yoshio Tanaka, and Noritake Sugitsue. Application of support vector machine to rapid classification of uranium waste drums using low-resolution gamma-ray spectra. *Applied Radiation and Isotopes*, 104:143–146, 2015.

[28] IAEA. Treaty on the non-proliferation of nuclear weapons. Technical report, Vienna: International Atomic Energy Agency, 1970.

[29] Lars J. Kangas, Paul E. Keller, Edward R. Siciliano, Richard T. Kouzes, and James H. Ely. The use of artificial neural networks in pvt-based radiation portal monitors. *Nuclear Instruments and Methods in Physics Research A*, 587:398–412, 2008.

[30] L.J. Kangas, S. Hashem, P.E. Keller, and R.T. Kouzes. Alpha spectral analysis via artificial neural networks. In *Nuclear Science Symposium Conference*. U.S. Department of Energy, 1994.

[31] P.E. Keller, R.T. Kouzes, and L.J. Kangas. Applications of neural networks to real-time data processing at the environmental and molecular sciences laboratory (emsl). In *8th Conference on Real-time Computer Applications in Nuclear Particle and Plasma Physics*, 1993.

[32] Ross D. King, Stephen Muggleton, Richard A. Lewis, and Michael J. E. Sternberg. Drug design by machine learning: The use of inductive logic programming to model the structure-activity relationships of trimethoprim analogues binding to dihydrofolate reductase. *Proceedings Of The National Academy Of Sciences*, 89(23):11322–11326, 1992.

[33] Chang-Ju Leea and Kun Jai Lee. Application of bayesian network to the probabilistic risk assessment of nuclear waste disposal. *Reliability Engineering and System Safety*, 91:515–532, 2006.

[34] Jeremy M. Lerner and Talwei Lu. Practical neural networks aid spectroscopic analysis. *Photonics spectra*, 27(8):93–, 1993.

[35] Robert C. Little, Mark B. Chadwick, and William L. Myers. Detection of highly enriched uranium through active interrogation. Technical report, Los Alamos National Laboratory, 2006.

[36] Frederick Livingston. Implementing breiman's random forest algorithm into weka. *ECE591Q Machine Conference Paper*, 2005.

[37] Xianguan Long, Ning Huang, Taihua Li, Fuqing He, and Xiufeng Peng. An artifical neural network analysis of low-resolution x-ray fluorescence spectra. Technical report, Institute of Nuclear Science and Technology, 1997.

[38] Mitch Mannino. Real time temporal spectroscopy for characterizing special nuclear material. Master's thesis, Oregon State University, 2017.

[39] R.E. Marrs, E.B. Norman, J.T. Burke, R.A. Macri, H.A. Shugart, E. Browne, and A.R. Smith. Fission-product gamma-ray line pairs sensitive to fissile material and neutron energy. *Nuclear Instruments and Methods in Physics Research A*, 592:463–471, 2008.

[40] Thomas M. Miller, Bruce W. Patton, Brandon R. Grogan, James J. Henkel, Brian D. Murphy, Jeffrey O. Johnson, and John T. Mihalczo. Investigations of active interrogation techniques to detect special nuclear material in maritime environments: Standoff interrogation of small- and medium-sized cargo ships. *Nuclear Instruments and Methods in Physics Research B*, 316:94–104, 2013.

[41] M.M Mukaka. Statistics corner: A guide to appropriate use of correlation coefficient in medical research. *Malawi Medical Journal*, 24(3):69–71, 2012.

[42] R. Nithya and B. Santhi. Decision tree classifiers for mass classification. *International Journal of Signal and Imaging Systems Engineering*, 8(1/2):39–45, 2015.

[43] Eric B. Norman, Stanley G. Prussin, Ruth-Mary Larimer, Howard Shugart, Edgardo Browne, Alan R. Smith, Richard J. McDonald, Heino Nitsche, Puja Gupta, Michael I. Frank, and Thomas B. Gosnell. Signatures of fissile materials: high-energy gamma rays following fission. *Nuclear Instruments and Methods in Physics Research A*, 521:608–610, 2004.

[44] P. Olmos, J. C. Diaz, J. M. Perez, P. Aguayo, P. Gomez, and V. Rodellar. Drift problems in the automatic analysis of gamma-ray spectra using associative memory algorithms. *IEEE Transactions on nuclear science*, 41(3), 1994.

[45] P. Olmos, J. C. Diaz, J. M. Perez, P. Gomez, V. Rodellar, P. Aguayo, A. Bru, G. Garcia-Belmonte, and J. L. de Pablos. A new approach to automatic radiation spectrum analysis. *IEEE Transactions on nuclear science*, 38(4), 1991.

[46] P. Olmos, J.C. Diaz, J.M. Perez, and G. Garcia-Belmonte. Application of neural network techniques in gamma spectroscopy. *Nuclear Instruments and Methods in Physics Research A*, 312:167–173, 1992.

[47] Helen M. O'D. Parker and Malcolm J. Joyce. The use of ionising radiation to image nuclear fuel: A review. *Progress in Nuclear Energy*, 85:297–318, 2015.

[48] V. Pilato. Application of neural netowrks to quantitative spectrometry analysis. *Nuclear Instruments and Methods in Physics Research A*, 422:423–427, 1999.

[49] Doug Reilly, Norbert Ensslin, and Hastings Smith Jr. *Passive Nondestructive Assay of Nuclear Materials.* U.S. Nuclear Regulatory Commission, 1991.

[50] J. Robinson. *Design and Construction and Characterization of a Neutron Depth Profiling Facility at the Oregon State University TRIGA Reactor with an Advanced Digital Spectroscopy System.* PhD thesis, Oregon State University, 2012.

[51] Douglas C. Rodriguez, Elaina Anderson, Kevin K. Anderson, Luke W. Campbell, James E. Fast, Kenneth Jarman, Jonathan Kulisek, Christopher R. Orton, Robert C.Runkle, and SeanStave. Measurementandanalysisofgamma-raysemitted fromspentnuclearfuelabove3mev. *Applied Radiation and Isotopes*, 2013.

[52] Second, editor. *Fundamentals of Nuclear Science and Engineering*. CRC Press Taylor and Francis Group, 2008.

[53] Robert M. Haralick Selim Aksoy. Feature normalization and liklihood-based similarity measures for image retrieval. 2000.

[54] Shiven Sharma, Colin Bellinger, and Nathalie Japkowicz. Anomaly detection in gamma ray spectra: A machine learning perspective. *IEEE Transactions on nuclear science*, 2012.

[55] David J. Sheskin. *Handbook of Parametric and Nonparametric Statistical Procedures*. Chapman and Hall/CRC, fourth edition, 2011.

[56] D.R. Slaughter, M.R. Accatino, A. Bernstein, J.A. Church, M.A. Descalle, T.B. Gosnell, J.M. Hall, A. Loshak, D.R. Manatt, G.J. Mauger, T.L. Moore, E.B. Norman, B.A. Pohl, J.A. Pruet, D.C. Petersen, R.S. Walling, D.L. Weirup, S.G. Prussin, and M. McDowell. Preliminary results utilizing high-energy fission product gamma-rays to detect fissionable material in cargo. *Nuclear Instruments and Methods in Physics Research B*, 241:777–781, 2005.

[57] C.J. Sullivan and J. Stinnett. Validation of a bayesian-based isotope identification algorithm. *Nuclear Instruments and Methods in Physics Research A*, 784:298–305, 2015.

[58] Sergios Theodoridis. *Pattern Recognition*. Elsevier, 4th edition, 1951.

[59] S.J. Tobin. Determination of plutonium content in spent fuel with nondestructive assay. Technical report, Lawrence Livermore National Laboratory, 2010.

[60] Adam Varley, Andrew Tyler, Leslie Smith, Paul Dale, and Mike Davies. Remediating radium contaminated legacy sites: Advances made through machine learning in routine monitoring of "hot" particles. *Science of the Total Environment*, 521-522:270–279, 2015.

[61] Adam Varley, Andrew Tyler, Leslie Smith, Paul Dale, and Mike Davies. Mapping the spatial distribution and activity of 226ra at legacy sites through machine learning interpretation of gamma-ray spectrometry data. *Science of the Total Environment*, 545-546:654–661, 2016.

[62] Vincent Vigneron, Jean Morel, Marie-Christine Lepy, and Jean-Marc Martinez. Statistical modelling by neural networks in gamma-spectrometry. In *Conference and International Symposium on Radionuclide Meterology*, 1995.

[63] Wei Wei, Qian Du, and Nicolas H. Younan. Particle swarm optimization based spectral transformation for radioactive material detection and classification. *IEEE Transactions on nuclear science*, 2010.

[64] Russell S. Williford. *Temporal Gamma-Ray Spectrometry to Quantify Relative Fissile Material Content*. PhD thesis, Oregon State University, 2013.

[65] Ian H. Witten, Eibe Frank, Mark A. Hall, and Christopher J. Pal. *Data Mining Practical Machine Learning Tools and Techniques*. Elsevier, fourth edition, 2017.

[66] Barry J. Wythoff, Steven P. Levine, and Sterling A. Tomellini. Spectral peak verification and recognition using a multilayered neural network. *Analytical Chemistry*, 62(24), 1990.

[67] Eiji Yoshida, Kiyoshi Shizumaa, Satoru Endo, and Takamitsu Oka. Application of neural networks for the analysis of gamma-ray spectra measured with a ge spectrometer. *Nuclear Instruments and Methods in Physics Research A*, 484:557–563, 2002.

A Data Parsing

```
# import modules used here -- sys is a very standard one
import sys
from datetime import datetime, date

print ('Program starting, filename is first arg (NOTE: must not have any blank spaces
     in name!)):', sys.argv[1])
d_filename=sys.argv[1] ; #print("File to read is: "+ d_filename)
idName, rej = d_filename.split(".")
outFile, ext = d_filename.split('.')
print("raw file name="+outFile)
outfileData=outFile+"Data_1.csv"
outfileHeader=outFile+"Header.csv"
outfileData=open(outfileData,'w')
outfileHeader=open(outfileHeader,'w')

header=0; cycleID=0; linecounter=0; writelinecounter=0; ADCcounter=0; EX1counter=0;
    mS1counter=0
detectorOn=0; detectorOff=0; mS0counter=0; writeLine=" "; foundheader=0;newline="";
    milliSec=" ";

with open(d_filename) as fp:
     for line in fp:
          linecounter +=1;
          if (header==0): outfileHeader.write(line)
          if 'Acq Start Reference' in line: header=1;
          if (line.find('ADC:',0,len(line))>-1 and detectorOn==1 and line.find('Real:
              ',0,len(line))>-1 and line.find('UTC Time:',0,len(line))>-1 ):
               rej,newline=line.split('Real:');
               newline1,time=newline.split('UTC Time:');
               newline2,channel=newline1.split('ADC:');
               milliSec,rej=newline2.split('mS');

               #print(str(cycleID)+","+ milliSec +","+ channel +","+ time);
               writeLine=str(cycleID)+","+ milliSec +","+ channel +","+ time;

               #print(writeLine);
               outfileData.write(writeLine);

          if (line.find('EX1',0,len(line))>-1 and line.find("mS: 1",0,len(line))>-1
              and detectorOff==1):
               mS1counter +=1; detectorOn=1; detectorOff=0; cycleID +=1;
               outfileData.close();
               outfileData=outFile+"Data_"+str(cycleID)+".csv";
               outfileData=open(outfileData,'w');
               outfileData.write("runID, millisec, Bin, timestamp \n");
               print("lines read so far: "+ str(linecounter)+ " ,last line of
                   previous cycle: " + str(writeLine)+ "\n new cycle: "+ str(cycleID)
                    + " line: "+ line);

          if (line.find('EX1',0,len(line))>-1 and line.find("mS: 0",0,len(line))>-1):
               mS0counter +=1; detectorOn=0; detectorOff=1;
print("cycleCounter:"+str(cycleID)+" linecounter: "+ str(linecounter) + " mS1counter:
    "+str(mS1counter))
# to see all millisec decimal places in R options(digits=10) and to load into R
    dataframe: filename<-read.csv("parsedFileName")
```

B Data Pre-Processing for HEU

```
##################################################################################
#This code produces the data analysis (used for data pre-processing) for HEU to train
     the classifiers over sliding windows of 3 and 5
##################################################################################
library(tidyverse)
library(qdap)
library(dplyr)
library(zoo)
library(RcppRoll)
library(tidyverse)
library(gdata)
library(gtools)
##################################################################################
#Read in Files (depends on path you need)
##################################################################################
out<-"~/Desktop/TestCodeFiles/HEUTestCode_Train"
setwd(out)
dir(out)
mcsv_r(dir(out))
##################################################################################
#Read in functions
##################################################################################
MakeCutOneCycle<-function(df){
  w<-cut(df$millisec,breaks=10,labels=FALSE); #Creates function called
      MakeCutOneCycle; cuts the millisec field into a sequence of ten intervals;
t<-data.frame(table(df$Bin,by=w));return(t)} #Creates a frequency table for each
    interval
##################################################################################
MakeMatrixOneCycle<-{function(cycle){
  for(i in 1:10){s<-subset(cycle,cycle$by==i);
    s$name<-NULL;s$by<-NULL; colnames(s)<-c("Var1",as.character(i));
    if(i==1){a<-s};
    if(i>1){a<-merge(a,s,by="Var1",all=TRUE)}};
    rownames(a)<-a$Var1;a$Var1<-NULL;return(a)}}
##################################################################################
SpearmansScore<-function(x,y){
  z<-cor.test(as.numeric(x),as.numeric(y),method="spearman",exact=FALSE);
  return(z$estimate)}
##################################################################################
#Read in template file and assign vector to template name. Get rid of first column.
    Only used one template since the decay and ingrowth templates were perfectly
    anticorrelated.
##################################################################################
Templates<-read.csv("Templates.csv") #Includes decay and ingrowth template
t1<-Templates[1,] #Decay template
t1$X<-NULL
##################################################################################
#Make Temporal interval data frame from the raw time stamped data create a data frame
     of interval cuts for one cycle. New file contains channels under 'Var1', count
    under 'Freq', and the time interval under 'by'
##################################################################################
HEUCuts1<-MakeCutOneCycle(HEU1m1)
HEUCuts2<-MakeCutOneCycle(HEU1m2)
HEUCuts3<-MakeCutOneCycle(HEU1m3)
HEUCuts4<-MakeCutOneCycle(HEU1m4)
HEUCuts5<-MakeCutOneCycle(HEU1m5)
HEUCuts6<-MakeCutOneCycle(HEU1m6)
HEUCuts7<-MakeCutOneCycle(HEU1m7)
HEUCuts8<-MakeCutOneCycle(HEU1m8)
HEUCuts9<-MakeCutOneCycle(HEU1m9)
HEUCuts10<-MakeCutOneCycle(HEU1m10)
HEUCuts11<-MakeCutOneCycle(HEU1m11)
HEUCuts12<-MakeCutOneCycle(HEU1m12)
```

```
HEUCuts13<-MakeCutOneCycle(HEU1m13)
HEUCuts14<-MakeCutOneCycle(HEU1m14)
HEUCuts15<-MakeCutOneCycle(HEU1m15)
HEUCuts16<-MakeCutOneCycle(HEU1m16)
HEUCuts17<-MakeCutOneCycle(HEU1m17)
HEUCuts18<-MakeCutOneCycle(HEU1m18)
HEUCuts19<-MakeCutOneCycle(HEU1m19)
HEUCuts20<-MakeCutOneCycle(HEU1m20)
##################################################################################
#Make a matrix from temporal data for each cycle
##################################################################################
HeuMatrix1<-MakeMatrixOneCycle(HEUCuts1)
HeuMatrix2<-MakeMatrixOneCycle(HEUCuts2)
HeuMatrix3<-MakeMatrixOneCycle(HEUCuts3)
HeuMatrix4<-MakeMatrixOneCycle(HEUCuts4)
HeuMatrix5<-MakeMatrixOneCycle(HEUCuts5)
HeuMatrix6<-MakeMatrixOneCycle(HEUCuts6)
HeuMatrix7<-MakeMatrixOneCycle(HEUCuts7)
HeuMatrix8<-MakeMatrixOneCycle(HEUCuts8)
HeuMatrix9<-MakeMatrixOneCycle(HEUCuts9)
HeuMatrix10<-MakeMatrixOneCycle(HEUCuts10)
HeuMatrix11<-MakeMatrixOneCycle(HEUCuts11)
HeuMatrix12<-MakeMatrixOneCycle(HEUCuts12)
HeuMatrix13<-MakeMatrixOneCycle(HEUCuts13)
HeuMatrix14<-MakeMatrixOneCycle(HEUCuts14)
HeuMatrix15<-MakeMatrixOneCycle(HEUCuts15)
HeuMatrix16<-MakeMatrixOneCycle(HEUCuts16)
HeuMatrix17<-MakeMatrixOneCycle(HEUCuts17)
HeuMatrix18<-MakeMatrixOneCycle(HEUCuts18)
HeuMatrix19<-MakeMatrixOneCycle(HEUCuts19)
HeuMatrix20<-MakeMatrixOneCycle(HEUCuts20)
##################################################################################
#Rename the rownames to 'channel'
##################################################################################
HeuMatrix1$channel<-rownames(HeuMatrix1)
HeuMatrix2$channel<-rownames(HeuMatrix2)
HeuMatrix3$channel<-rownames(HeuMatrix3)
HeuMatrix4$channel<-rownames(HeuMatrix4)
HeuMatrix5$channel<-rownames(HeuMatrix5)
HeuMatrix6$channel<-rownames(HeuMatrix6)
HeuMatrix7$channel<-rownames(HeuMatrix7)
HeuMatrix8$channel<-rownames(HeuMatrix8)
HeuMatrix9$channel<-rownames(HeuMatrix9)
HeuMatrix10$channel<-rownames(HeuMatrix10)
HeuMatrix11$channel<-rownames(HeuMatrix11)
HeuMatrix12$channel<-rownames(HeuMatrix12)
HeuMatrix13$channel<-rownames(HeuMatrix13)
HeuMatrix14$channel<-rownames(HeuMatrix14)
HeuMatrix15$channel<-rownames(HeuMatrix15)
HeuMatrix16$channel<-rownames(HeuMatrix16)
HeuMatrix17$channel<-rownames(HeuMatrix17)
HeuMatrix18$channel<-rownames(HeuMatrix18)
HeuMatrix19$channel<-rownames(HeuMatrix19)
HeuMatrix20$channel<-rownames(HeuMatrix20)
##################################################################################
#Arrange channels and fill in missing channels with NA and then to 0.
##################################################################################
HEU1FILL<- HeuMatrix1 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
HEU1FILLNoNA<-HEU1FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
HEU1RunID<-add_column(HEU1FILLNoNA,runID=1) #Assign runID
write.csv(HEU1RunID,file= "HEU1RunID.csv") #Save as .csv
HEU1RunID<-read.csv(file= "HEU1RunID.csv") #Read in .csv
HEU1RunID$X<-NULL #Remove column 'X' due to read/write .csv
```

```
HEU2FILL<- HeuMatrix2 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
HEU2FILLNoNA<-HEU2FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
HEU2RunID<-add_column(HEU2FILLNoNA,runID=2) #Assign runID
write.csv(HEU2RunID,file= "HEU2RunID.csv") #Save as .csv
HEU2RunID<-read.csv(file= "HEU2RunID.csv") #Read in .csv
HEU2RunID$X<-NULL #Remove column 'X' due to read/write .csv

HEU3FILL<- HeuMatrix3 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
HEU3FILLNoNA<-HEU3FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
HEU3RunID<-add_column(HEU3FILLNoNA,runID=3) #Assign runID
write.csv(HEU3RunID,file= "HEU3RunID.csv") #Save as .csv
HEU3RunID<-read.csv(file= "HEU3RunID.csv") #Read in .csv
HEU3RunID$X<-NULL #Remove column 'X' due to read/write .csv

HEU4FILL<- HeuMatrix4 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
HEU4FILLNoNA<-HEU4FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
HEU4RunID<-add_column(HEU4FILLNoNA,runID=4) #Assign runID
write.csv(HEU4RunID,file= "HEU4RunID.csv") #Save as .csv
HEU4RunID<-read.csv(file= "HEU4RunID.csv") #Read in .csv
HEU4RunID$X<-NULL #Remove column 'X' due to read/write .csv

HEU5FILL<- HeuMatrix5 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
HEU5FILLNoNA<-HEU5FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
HEU5RunID<-add_column(HEU5FILLNoNA,runID=5) #Assign runID
write.csv(HEU5RunID,file= "HEU5RunID.csv") #Save as .csv
HEU5RunID<-read.csv(file= "HEU5RunID.csv") #Read in .csv
HEU5RunID$X<-NULL #Remove column 'X' due to read/write .csv

HEU6FILL<- HeuMatrix6 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
HEU6FILLNoNA<-HEU6FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
HEU6RunID<-add_column(HEU6FILLNoNA,runID=6) #Assign runID
write.csv(HEU6RunID,file= "HEU6RunID.csv") #Save as .csv
HEU6RunID<-read.csv(file= "HEU6RunID.csv") #Read in .csv
HEU6RunID$X<-NULL #Remove column 'X' due to read/write .csv

HEU7FILL<- HeuMatrix7 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
HEU7FILLNoNA<-HEU7FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
HEU7RunID<-add_column(HEU7FILLNoNA,runID=7) #Assign runID
write.csv(HEU7RunID,file= "HEU7RunID.csv") #Save as .csv
HEU7RunID<-read.csv(file= "HEU7RunID.csv") #Read in .csv
HEU7RunID$X<-NULL #Remove column 'X' due to read/write .csv

HEU8FILL<- HeuMatrix8 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
HEU8FILLNoNA<-HEU8FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
```

```
HEU8RunID<-add_column(HEU8FILLNoNA,runID=8) #Assign runID
write.csv(HEU8RunID,file= "HEU8RunID.csv") #Save as .csv
HEU8RunID<-read.csv(file= "HEU8RunID.csv") #Read in as .csv
HEU8RunID$X<-NULL #Remove column 'X' due to read/write .csv

HEU9FILL<- HeuMatrix9 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
HEU9FILLNoNA<-HEU9FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
HEU9RunID<-add_column(HEU9FILLNoNA,runID=9) #Assign runID
write.csv(HEU9RunID,file= "HEU9RunID.csv") #Save as .csv
HEU9RunID<-read.csv(file= "HEU9RunID.csv") #Read in as .csv
HEU9RunID$X<-NULL #Remove column 'X' due to read/write .csv

HEU10FILL<- HeuMatrix10 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
HEU10FILLNoNA<-HEU10FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
HEU10RunID<-add_column(HEU10FILLNoNA,runID=10) #Assign runID
write.csv(HEU10RunID,file= "HEU10RunID.csv") #Save as .csv
HEU10RunID<-read.csv(file= "HEU10RunID.csv") #Read in as .csv
HEU10RunID$X<-NULL #Remove column 'X' due to read/write .csv

HEU11FILL<- HeuMatrix11 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
HEU11FILLNoNA<-HEU11FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
HEU11RunID<-add_column(HEU11FILLNoNA,runID=11) #Assign runID
write.csv(HEU11RunID,file= "HEU11RunID.csv") #Save as .csv
HEU11RunID<-read.csv(file= "HEU11RunID.csv") #Read in as .csv
HEU11RunID$X<-NULL #Remove column 'X' due to read/write .csv

HEU12FILL<- HeuMatrix12 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
HEU12FILLNoNA<-HEU12FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
HEU12RunID<-add_column(HEU12FILLNoNA,runID=12) #Assign runID
write.csv(HEU12RunID,file= "HEU12RunID.csv") #Save as .csv
HEU12RunID<-read.csv(file= "HEU12RunID.csv") #Read in as .csv
HEU12RunID$X<-NULL #Remove column 'X' due to read/write .csv

HEU13FILL<- HeuMatrix13 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
HEU13FILLNoNA<-HEU13FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
HEU13RunID<-add_column(HEU13FILLNoNA,runID=13) #Assign runID
write.csv(HEU13RunID,file= "HEU13RunID.csv") #Save as .csv
HEU13RunID<-read.csv(file= "HEU13RunID.csv") #Read in as .csv
HEU13RunID$X<-NULL #Remove column 'X' due to read/write .csv

HEU14FILL<- HeuMatrix14 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
HEU14FILLNoNA<-HEU14FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
HEU14RunID<-add_column(HEU14FILLNoNA,runID=14) #Assign runID
write.csv(HEU14RunID,file= "HEU14RunID.csv") #Save as .csv
HEU14RunID<-read.csv(file= "HEU14RunID.csv") #Read in as .csv
HEU14RunID$X<-NULL #Remove column 'X' due to read/write .csv

HEU15FILL<- HeuMatrix15 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
HEU15FILLNoNA<-HEU15FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
HEU15RunID<-add_column(HEU15FILLNoNA,runID=15) #Assign runID
write.csv(HEU15RunID,file= "HEU15RunID.csv") #Save as .csv
HEU15RunID<-read.csv(file= "HEU15RunID.csv") #Read in as .csv
HEU15RunID$X<-NULL #Remove column 'X' due to read/write as .csv

HEU16FILL<- HeuMatrix16 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
HEU16FILLNoNA<-HEU16FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
HEU16RunID<-add_column(HEU16FILLNoNA,runID=16) #Assign runID
write.csv(HEU16RunID,file= "HEU16RunID.csv") #Save as .csv
HEU16RunID<-read.csv(file= "HEU16RunID.csv") #Read in as .csv
HEU16RunID$X<-NULL #Remove column 'X' due to read/write as .csv

HEU17FILL<- HeuMatrix17 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
HEU17FILLNoNA<-HEU17FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
HEU17RunID<-add_column(HEU17FILLNoNA,runID=17) #Assign runID
write.csv(HEU17RunID,file= "HEU17RunID.csv") #Save as .csv
HEU17RunID<-read.csv(file= "HEU17RunID.csv") #Read in as .csv
HEU17RunID$X<-NULL #Remove column 'X' due to read/write as .csv

HEU18FILL<- HeuMatrix18 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
HEU18FILLNoNA<-HEU18FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
HEU18RunID<-add_column(HEU18FILLNoNA,runID=18) #Assign runID
write.csv(HEU18RunID,file= "HEU18RunID.csv") #Save as .csv
HEU18RunID<-read.csv(file= "HEU18RunID.csv") #Read in as .csv
HEU18RunID$X<-NULL #Remove column 'X' due to read/write as .csv

HEU19FILL<- HeuMatrix19 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
HEU19FILLNoNA<-HEU19FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
HEU19RunID<-add_column(HEU19FILLNoNA,runID=19) #Assign runID
write.csv(HEU19RunID,file= "HEU19RunID.csv") #Save as .csv
HEU19RunID<-read.csv(file= "HEU19RunID.csv") #Read in as .csv
HEU19RunID$X<-NULL #Remove column 'X' due to read/write as .csv

HEU20FILL<- HeuMatrix20 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
HEU20FILLNoNA<-HEU20FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
HEU20RunID<-add_column(HEU20FILLNoNA,runID=20) #Assigns a runID
write.csv(HEU20RunID,file= "HEU20RunID.csv") #Save as .csv
HEU20RunID<-read.csv(file= "HEU20RunID.csv") #Read in as .csv
HEU20RunID$X<-NULL #Remove column 'X' due to read/write as .csv
################################################################################
#Make a list of the files you want to load and create an empty dataframe
################################################################################
HEUdata2 <- c("HEU1RunID", "HEU2RunID", "HEU3RunID", "HEU4RunID", "HEU5RunID",
            "HEU6RunID", "HEU7RunID", "HEU8RunID", "HEU9RunID", "HEU10RunID",
            "HEU11RunID", "HEU12RunID", "HEU13RunID","HEU14RunID", "HEU15RunID",
            "HEU16RunID", "HEU17RunID", "HEU18RunID", "HEU19RunID", "HEU20RunID")
datHEU2014 <- data.frame() #Create an empty data frame to fill
```

```
# Read csv, add a column referring to the runID
# Then combine them into one data folder
for (runID in HEUdata2) {
  filename = paste(runID, ".csv", sep="")
  t <- read.csv(filename)
  t$runID <- runID
  datHEU2014<- rbind(datHEU2014, t)
}
#head(datHEU2014) #Overview of dataframe as a check
rownames(datHEU2014)<-NULL #Delete rownames
datHEU2014$X<-NULL #Delete column X (channel numbers)
##############################################################################
#Sliding window of 3. This sums every three rows for every cycle and creates a new
    dataframe
##############################################################################
newtestdatSW3<-datHEU2014 #Create copy of the dataframe
cyclesSW3<-unique(newtestdatSW3$runID) #Creates a vector of all cycles

get_countsSW3<-function(x){
  HEU<- newtestdatSW3 %>%
    filter(runID==x) %>%  #Select cycle
    arrange(channel)       #Make sure in ascending order based on channel
  counts1<-roll_sum(HEU$X1,n=3)  #Sliding window of three rows
  counts2<-roll_sum(HEU$X2,n=3)
  counts3<-roll_sum(HEU$X3,n=3)
  counts4<-roll_sum(HEU$X4,n=3)
  counts5<-roll_sum(HEU$X5,n=3)
  counts6<-roll_sum(HEU$X6,n=3)
  counts7<-roll_sum(HEU$X7,n=3)
  counts8<-roll_sum(HEU$X8,n=3)
  counts9<-roll_sum(HEU$X9,n=3)
  counts10<-roll_sum(HEU$X10,n=3)
df<-data.frame(X1=counts1,X2=counts2,X3=counts3, X4=counts4,
                 X5=counts5, X6=counts6, X7=counts7, X8=counts8,
                 X9=counts9, X10=counts10,runID=x) #Creates a dataframe
}
SW3<-map_df(cyclesSW3,get_countsSW3) #Apply function to all cycles
##############################################################################
#Apply function to get new channels and their labels for sliding window of 3
##############################################################################
get_channelsSW3<-function(x){
  HEU2<- newtestdatSW3 %>%
    filter(runID==x) %>% #Select cycle
    arrange(channel)
  test4<-data.frame(HEU2$channel, new_channel=dplyr::lead(HEU2$channel,1)) #Names
      channels with middle number
}
NC3<-map_df(cyclesSW3,get_channelsSW3) #apply sliding window of 3 function
##############################################################################
#Filter extra values. Filters NC3 so that the extra 40 values are removed and bind
    with main dataframe
##############################################################################
Filter3<-NC3 %>%
  filter(!is.na(new_channel)) %>%
  subset(new_channel!=16383) %>%
  select(-c(HEU2.channel))
Final3<-cbind(SW3,Filter3) #This binds the two dataframes
##############################################################################
#Feature generation
##############################################################################
Final3$mean<-rowMeans(Final3[,1:10]) #Create feature "mean"
Final3<-Final3 %>%
  rowwise() %>%
  mutate(median = median(c(X1,X2,X3,X4,X5,X6,X7,X8,X9,X10), na.rm = TRUE)) #Create
      feature 'median'
Final3$rho<-apply(Final3[,1:10],1,SpearmansScore,t1) #Create feature 'rho'
```

```
#Adds column of chunked rho scores as an additional feature
Final3<- Final3 %>%
  mutate(rho_cat=case_when(rho >=0.7000000 & rho<=0.9999999 ~ 'StrongPos',
                           rho >=0.5000000 & rho <=0.6999999 ~ 'ModeratePos',
                           rho >= 0.3000000 & rho <= 0.4999999 ~ 'WeakPos',
                           rho >= 0.0000000 & rho <=0.2999999 ~ 'NoRelationship',
                           rho <=0.0000000 & rho >= -0.2999999~ 'NoRelationship',
                           rho <= -0.3000000 & rho >= -0.4999999~ 'WeakNeg',
                           rho <=-0.5000000 & rho >= -0.6999999~ 'ModerateNeg',
                           rho <= -0.7000000 & rho>=-0.9999999~ 'StrongNeg',
                           rho == "1" ~'PerfectPos',
                           rho == "-1" ~ 'PerfectNeg'))
Final3<-Final3[c(1,2,3,4,5,6,7,8,9,10,12,11,13,14,15,16)] #Reorder the data frame #
    this only works after feature generation is complete
##################################################################################
#Save to folder for merge with Pu files and perform statistical tests for finding
    significant channels
Final3HEU<-Final3
write.csv(Final3HEU,file="Final3HEU.csv")
##################################################################################
#Sliding window of 5. This sums the counts every five rows for every cycle and
    creates a new dataframe.
##################################################################################
newtestdatSW5<-datHEU2014 #Create copy of the dataframe (keeps dataframes separate)
cyclesSW5<-unique(newtestdatSW5$runID) #This creates a vector of all cycles
get_countsSW5<-function(x){
  HEU<- newtestdatSW5 %>%
    filter(runID==x) %>%  #Select cycle
    arrange(channel)      #Make sure in ascending order based on channel
  counts1<-roll_sum(HEU$X1,n=5)  #Sliding window of five rows
  counts2<-roll_sum(HEU$X2,n=5)
  counts3<-roll_sum(HEU$X3,n=5)
  counts4<-roll_sum(HEU$X4,n=5)
  counts5<-roll_sum(HEU$X5,n=5)
  counts6<-roll_sum(HEU$X6,n=5)
  counts7<-roll_sum(HEU$X7,n=5)
  counts8<-roll_sum(HEU$X8,n=5)
  counts9<-roll_sum(HEU$X9,n=5)
  counts10<-roll_sum(HEU$X10,n=5)
 df<-data.frame(X1=counts1,X2=counts2,X3=counts3, X4=counts4,
                 X5=counts5, X6=counts6, X7=counts7, X8=counts8,
                 X9=counts9, X10=counts10,runID=x) #Creates a dataframe
}
SW5<-map_df(cyclesSW5,get_countsSW5) #Apply function to all cycles
##################################################################################
#Function for getting new channel labels. Get new channels for sliding window of 5
##################################################################################
get_channelsSW5<-function(x){
  HEU3<- newtestdatSW5 %>%
    filter(runID==x) %>%   #Select cycle
    arrange(channel)
  test4<-data.frame(HEU3$channel, new_channel=dplyr::lead(HEU3$channel,2)) #This
      would name the channels correctly
}
NC5<-map_df(cyclesSW5,get_channelsSW5)
##################################################################################
#Filter extra values. Filters NC5 so that the extra 80 values are removed and bound
    with main dataframe
##################################################################################
Filter5<-NC5 %>%
  filter(!is.na(new_channel)) %>%
  subset(new_channel!=16383) %>%
  subset(new_channel!=16382) %>%
  select(-c(HEU3.channel))
Final5<-cbind(SW5,Filter5) #This binds the two dataframes
##################################################################################
```

```
#Feature generation for SW5
#################################################################################
Final5$mean<-rowMeans(Final5[,1:10]) #Create feature "mean"
Final5<-Final5 %>%
  rowwise() %>%
  mutate(median = median(c(X1,X2,X3,X4,X5,X6,X7,X8,X9,X10), na.rm = TRUE)) #Create
      feature 'median'
Final5$rho<-apply(Final5[,1:10],1,SpearmansScore,t1) #Create feature 'rho'
#Adds column of chunked rho scores as an additional feature for SW5
Final5<- Final5 %>%
  mutate(rho_cat=case_when(rho >=0.7000000 & rho<=0.9999999 ~ 'StrongPos',
                           rho >=0.5000000 & rho <=0.6999999 ~ 'ModeratePos',
                           rho >= 0.3000000 & rho <= 0.4999999 ~ 'WeakPos',
                           rho >= 0.0000000 & rho <=0.2999999 ~ 'NoRelationship',
                           rho <=0.0000000 & rho >= -0.2999999~ 'NoRelationship',
                           rho <= -0.3000000 & rho >= -0.4999999~ 'WeakNeg',
                           rho <=-0.5000000 & rho >= -0.6999999~ 'ModerateNeg',
                           rho <= -0.7000000 & rho>=-0.9999999~ 'StrongNeg',
                           rho == "1" ~'PerfectPos',
                           rho == "-1" ~ 'PerfectNeg'))
Final5<-Final5[c(1,2,3,4,5,6,7,8,9,10,12,11,13,14,15,16)] #Reorder the data frame
    after feature
#generation is complete
#################################################################################
#Save to folder for merge with Pu files and perform statistical tests for finding
    significant channels
Final5HEU<-Final5
write.csv(Final5HEU,file="Final5HEU.csv")
```

C Data Pre-Processing for Pu-239

```
################################################################################
#This code produces the data analysis (used for data pre-processing) for Pu to train
    the classifiers over sliding windows of 3 and 5.
################################################################################
library(tidyverse)
library(qdap)
library(dplyr)
library(zoo)
library(RcppRoll)
library(tidyverse)
library(gdata)
library(gtools)
################################################################################
#Read in Files (depends on path you need)
################################################################################
out<-"~/Desktop/TestCodeFiles/PuTestCode_Train"
setwd(out)
dir(out)
mcsv_r(dir(out))
################################################################################
#Read in functions
################################################################################
MakeCutOneCycle<-function(df){
  w<-cut(df$millisec,breaks=10,labels=FALSE); #Creates function called
      MakeCutOneCycle; cuts the millisec field into a sequence of ten intervals;
  t<-data.frame(table(df$Bin,by=w));return(t)} #Creates a frequency table for each
      interval
################################################################################
MakeMatrixOneCycle<-{function(cycle){
  for(i in 1:10){s<-subset(cycle,cycle$by==i);
  s$name<-NULL;s$by<-NULL; colnames(s)<-c("Var1",as.character(i));
  if(i==1){a<-s};
  if(i>1){a<-merge(a,s,by="Var1",all=TRUE)}};
  rownames(a)<-a$Var1;a$Var1<-NULL;return(a)}}
################################################################################
SpearmansScore<-function(x,y){
  z<-cor.test(as.numeric(x),as.numeric(y),method="spearman",exact=FALSE);
  return(z$estimate)}
################################################################################
#Read in template file and assign vector to template name. Get rid of first column.
    Only used one template since the decay and ingrowth templates were perfectly
    anticorrelated.
################################################################################
Templates<-read.csv("Templates.csv") #Includes decay and ingrowth template
t1<-Templates[1,] #Decay template
t1$X<-NULL
################################################################################
#Make Temporal interval data frame from the raw time stamped data. Create a data
    frame of interval cuts for one cycle. New file contains channels under 'Var1',
    count under 'Freq', and the time interval under 'by'.
################################################################################
PuCuts1<-MakeCutOneCycle(Pu1m1)
PuCuts2<-MakeCutOneCycle(Pu1m2)
PuCuts3<-MakeCutOneCycle(Pu1m3)
PuCuts4<-MakeCutOneCycle(Pu1m4)
PuCuts5<-MakeCutOneCycle(Pu1m5)
PuCuts6<-MakeCutOneCycle(Pu1m6)
PuCuts7<-MakeCutOneCycle(Pu1m7)
PuCuts8<-MakeCutOneCycle(Pu1m8)
PuCuts9<-MakeCutOneCycle(Pu1m9)
PuCuts10<-MakeCutOneCycle(Pu1m10)
PuCuts11<-MakeCutOneCycle(Pu1m11)
PuCuts12<-MakeCutOneCycle(Pu1m12)
```

```
PuCuts13<-MakeCutOneCycle(Pu1m13)
PuCuts14<-MakeCutOneCycle(Pu1m14)
PuCuts15<-MakeCutOneCycle(Pu1m15)
PuCuts16<-MakeCutOneCycle(Pu1m16)
PuCuts17<-MakeCutOneCycle(Pu1m17)
PuCuts18<-MakeCutOneCycle(Pu1m18)
PuCuts19<-MakeCutOneCycle(Pu1m19)
PuCuts20<-MakeCutOneCycle(Pu1m20)
################################################################################
#Make a matrix from temporal data. Make interval matrix for each cycle
################################################################################
PuMatrix1<-MakeMatrixOneCycle(PuCuts1)
PuMatrix2<-MakeMatrixOneCycle(PuCuts2)
PuMatrix3<-MakeMatrixOneCycle(PuCuts3)
PuMatrix4<-MakeMatrixOneCycle(PuCuts4)
PuMatrix5<-MakeMatrixOneCycle(PuCuts5)
PuMatrix6<-MakeMatrixOneCycle(PuCuts6)
PuMatrix7<-MakeMatrixOneCycle(PuCuts7)
PuMatrix8<-MakeMatrixOneCycle(PuCuts8)
PuMatrix9<-MakeMatrixOneCycle(PuCuts9)
PuMatrix10<-MakeMatrixOneCycle(PuCuts10)
PuMatrix11<-MakeMatrixOneCycle(PuCuts11)
PuMatrix12<-MakeMatrixOneCycle(PuCuts12)
PuMatrix13<-MakeMatrixOneCycle(PuCuts13)
PuMatrix14<-MakeMatrixOneCycle(PuCuts14)
PuMatrix15<-MakeMatrixOneCycle(PuCuts15)
PuMatrix16<-MakeMatrixOneCycle(PuCuts16)
PuMatrix17<-MakeMatrixOneCycle(PuCuts17)
PuMatrix18<-MakeMatrixOneCycle(PuCuts18)
PuMatrix19<-MakeMatrixOneCycle(PuCuts19)
PuMatrix20<-MakeMatrixOneCycle(PuCuts20)
################################################################################
#Rename the rownames to 'channel'
################################################################################
PuMatrix1$channel<-rownames(PuMatrix1)
PuMatrix2$channel<-rownames(PuMatrix2)
PuMatrix3$channel<-rownames(PuMatrix3)
PuMatrix4$channel<-rownames(PuMatrix4)
PuMatrix5$channel<-rownames(PuMatrix5)
PuMatrix6$channel<-rownames(PuMatrix6)
PuMatrix7$channel<-rownames(PuMatrix7)
PuMatrix8$channel<-rownames(PuMatrix8)
PuMatrix9$channel<-rownames(PuMatrix9)
PuMatrix10$channel<-rownames(PuMatrix10)
PuMatrix11$channel<-rownames(PuMatrix11)
PuMatrix12$channel<-rownames(PuMatrix12)
PuMatrix13$channel<-rownames(PuMatrix13)
PuMatrix14$channel<-rownames(PuMatrix14)
PuMatrix15$channel<-rownames(PuMatrix15)
PuMatrix16$channel<-rownames(PuMatrix16)
PuMatrix17$channel<-rownames(PuMatrix17)
PuMatrix18$channel<-rownames(PuMatrix18)
PuMatrix19$channel<-rownames(PuMatrix19)
PuMatrix20$channel<-rownames(PuMatrix20)
################################################################################
#Arrange channels and fill in missing channels with NA and then to 0.
################################################################################
Pu1FILL<- PuMatrix1 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Pu1FILLNoNA<-Pu1FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Pu1RunID<-add_column(Pu1FILLNoNA,runID=1) #Assign runID
write.csv(Pu1RunID,file= "Pu1RunID.csv") #Save as .csv
Pu1RunID<-read.csv(file= "Pu1RunID.csv") #Read in .csv
Pu1RunID$X<-NULL #Remove column 'X' due to read/write .csv
```

```
Pu2FILL<- PuMatrix2 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Pu2FILLNoNA<-Pu2FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Pu2RunID<-add_column(Pu2FILLNoNA,runID=2) #Assign runID
write.csv(Pu2RunID,file= "Pu2RunID.csv") #Save as .csv
Pu2RunID<-read.csv(file= "Pu2RunID.csv") #Read in .csv
Pu2RunID$X<-NULL #Remove column 'X' due to read/write .csv

Pu3FILL<- PuMatrix3 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Pu3FILLNoNA<-Pu3FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Pu3RunID<-add_column(Pu3FILLNoNA,runID=3) #Assign runID
write.csv(Pu3RunID,file= "Pu3RunID.csv") #Save as .csv
Pu3RunID<-read.csv(file= "Pu3RunID.csv") #Read in .csv
Pu3RunID$X<-NULL #Remove column 'X' due to read/write .csv

Pu4FILL<- PuMatrix4 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Pu4FILLNoNA<-Pu4FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Pu4RunID<-add_column(Pu4FILLNoNA,runID=4) #Assign runID
write.csv(Pu4RunID,file= "Pu4RunID.csv") #Save as .csv
Pu4RunID<-read.csv(file= "Pu4RunID.csv") #Read in .csv
Pu4RunID$X<-NULL #Remove column 'X' due to read/write .csv

Pu5FILL<- PuMatrix5 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Pu5FILLNoNA<-Pu5FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Pu5RunID<-add_column(Pu5FILLNoNA,runID=5) #Assign runID
write.csv(Pu5RunID,file= "Pu5RunID.csv") #Save as .csv
Pu5RunID<-read.csv(file= "Pu5RunID.csv") #Read in .csv
Pu5RunID$X<-NULL #Remove column 'X' due to read/write .csv

Pu6FILL<- PuMatrix6 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Pu6FILLNoNA<-Pu6FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Pu6RunID<-add_column(Pu6FILLNoNA,runID=6) #Assign runID
write.csv(Pu6RunID,file= "Pu6RunID.csv") #Save as .csv
Pu6RunID<-read.csv(file= "Pu6RunID.csv") #Read in .csv
Pu6RunID$X<-NULL #Remove column 'X' due to read/write .csv

Pu7FILL<- PuMatrix7 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Pu7FILLNoNA<-Pu7FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Pu7RunID<-add_column(Pu7FILLNoNA,runID=7) #Assign runID
write.csv(Pu7RunID,file= "Pu7RunID.csv") #Save as .csv
Pu7RunID<-read.csv(file= "Pu7RunID.csv") #Read in .csv
Pu7RunID$X<-NULL #Remove column 'X' due to read/write .csv

Pu8FILL<- PuMatrix8 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Pu8FILLNoNA<-Pu8FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
```

```
Pu8RunID<-add_column(Pu8FILLNoNA,runID=8) #Assign runID
write.csv(Pu8RunID,file= "Pu8RunID.csv") #Save as .csv
Pu8RunID<-read.csv(file= "Pu8RunID.csv") #Read in as .csv
Pu8RunID$X<-NULL #Remove column 'X' due to read/write .csv

Pu9FILL<- PuMatrix9 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Pu9FILLNoNA<-Pu9FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Pu9RunID<-add_column(Pu9FILLNoNA,runID=9) #Assign runID
write.csv(Pu9RunID,file= "Pu9RunID.csv") #Save as .csv
Pu9RunID<-read.csv(file= "Pu9RunID.csv") #Read in as .csv
Pu9RunID$X<-NULL #Remove column 'X' due to read/write .csv

Pu10FILL<- PuMatrix10 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Pu10FILLNoNA<-Pu10FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Pu10RunID<-add_column(Pu10FILLNoNA,runID=10) #Assign runID
write.csv(Pu10RunID,file= "Pu10RunID.csv") #Save as .csv
Pu10RunID<-read.csv(file= "Pu10RunID.csv") #Read in as .csv
Pu10RunID$X<-NULL #Remove column 'X' due to read/write .csv

Pu11FILL<- PuMatrix11 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Pu11FILLNoNA<-Pu11FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Pu11RunID<-add_column(Pu11FILLNoNA,runID=11) #Assign runID
write.csv(Pu11RunID,file= "Pu11RunID.csv") #Save as .csv
Pu11RunID<-read.csv(file= "Pu11RunID.csv") #Read in as .csv
Pu11RunID$X<-NULL #Remove column 'X' due to read/write .csv

Pu12FILL<- PuMatrix12 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Pu12FILLNoNA<-Pu12FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Pu12RunID<-add_column(Pu12FILLNoNA,runID=12) #Assign runID
write.csv(Pu12RunID,file= "Pu12RunID.csv") #Save as .csv
Pu12RunID<-read.csv(file= "Pu12RunID.csv") #Read in as .csv
Pu12RunID$X<-NULL #Remove column 'X' due to read/write .csv

Pu13FILL<- PuMatrix13 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Pu13FILLNoNA<-Pu13FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Pu13RunID<-add_column(Pu13FILLNoNA,runID=13) #Assign runID
write.csv(Pu13RunID,file= "Pu13RunID.csv") #Save as .csv
Pu13RunID<-read.csv(file= "Pu13RunID.csv") #Read in as .csv
Pu13RunID$X<-NULL #Remove column 'X' due to read/write .csv

Pu14FILL<- PuMatrix14 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Pu14FILLNoNA<-Pu14FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Pu14RunID<-add_column(Pu14FILLNoNA,runID=14) #Assign runID
write.csv(Pu14RunID,file= "Pu14RunID.csv") #Save as .csv
Pu14RunID<-read.csv(file= "Pu14RunID.csv") #Read in as .csv
Pu14RunID$X<-NULL #Remove column 'X' due to read/write .csv

Pu15FILL<- PuMatrix15 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Pu15FILLNoNA<-Pu15FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Pu15RunID<-add_column(Pu15FILLNoNA,runID=15) #Assign runID
write.csv(Pu15RunID,file= "Pu15RunID.csv") #Save as .csv
Pu15RunID<-read.csv(file= "Pu15RunID.csv") #Read in as .csv
Pu15RunID$X<-NULL #Remove column 'X' due to read/write as .csv

Pu16FILL<- PuMatrix16 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Pu16FILLNoNA<-Pu16FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Pu16RunID<-add_column(Pu16FILLNoNA,runID=16) #Assign runID
write.csv(Pu16RunID,file= "Pu16RunID.csv") #Save as .csv
Pu16RunID<-read.csv(file= "Pu16RunID.csv") #Read in as .csv
Pu16RunID$X<-NULL #Remove column 'X' due to read/write as .csv

Pu17FILL<- PuMatrix17 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Pu17FILLNoNA<-Pu17FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Pu17RunID<-add_column(Pu17FILLNoNA,runID=17) #Assign runID
write.csv(Pu17RunID,file= "Pu17RunID.csv") #Save as .csv
Pu17RunID<-read.csv(file= "Pu17RunID.csv") #Read in as .csv
Pu17RunID$X<-NULL #Remove column 'X' due to read/write as .csv

Pu18FILL<- PuMatrix18 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Pu18FILLNoNA<-Pu18FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Pu18RunID<-add_column(Pu18FILLNoNA,runID=18) #Assign runID
write.csv(Pu18RunID,file= "Pu18RunID.csv") #Save as .csv
Pu18RunID<-read.csv(file= "Pu18RunID.csv") #Read in as .csv
Pu18RunID$X<-NULL #Remove column 'X' due to read/write as .csv

Pu19FILL<- PuMatrix19 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Pu19FILLNoNA<-Pu19FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Pu19RunID<-add_column(Pu19FILLNoNA,runID=19) #Assign runID
write.csv(Pu19RunID,file= "Pu19RunID.csv") #Save as .csv
Pu19RunID<-read.csv(file= "Pu19RunID.csv") #Read in as .csv
Pu19RunID$X<-NULL #Remove column 'X' due to read/write as .csv

Pu20FILL<- PuMatrix20 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Pu20FILLNoNA<-Pu20FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Pu20RunID<-add_column(Pu20FILLNoNA,runID=20) #Assigns a runID
write.csv(Pu20RunID,file= "Pu20RunID.csv") #Save as .csv
Pu20RunID<-read.csv(file= "Pu20RunID.csv") #Read in as .csv
Pu20RunID$X<-NULL #Remove column 'X' due to read/write as .csv
################################################################################
#Make a list of the files you want to load and create an empty dataframe
################################################################################
Pudata2 <- c("Pu1RunID", "Pu2RunID", "Pu3RunID", "Pu4RunID", "Pu5RunID", "Pu6RunID",
                 "Pu7RunID", "Pu8RunID", "Pu9RunID", "Pu10RunID", "Pu11RunID",
                 "Pu12RunID", "Pu13RunID", "Pu14RunID", "Pu15RunID","Pu16RunID",
                 "Pu17RunID", "Pu18RunID", "Pu19RunID", "Pu20RunID")
datPu2014 <- data.frame() #create an empty data frame to fill
```

```
# Read csv, add a column referring to the runID then combine them into one data
    folder
for (runID in Pudata2) {
  filename = paste(runID, ".csv", sep="")
  t <- read.csv(filename)
  t$runID <- runID
  datPu2014<- rbind(datPu2014, t)
}
#head(datPu2014) #Overview of dataframe as a check
rownames(datPu2014)<-NULL #Delete rownames
datPu2014$X<-NULL #Delete column X (channel numbers)
################################################################################
#Sliding window of 3. This sums every three rows for every cycle and creates a new
    dataframe
################################################################################
newtestdatSW3<-datPu2014 #Create copy of the dataframe
cyclesSW3<-unique(newtestdatSW3$runID) #This creates a vector of all cycles

get_countsSW3<-function(x){
  Pu<- newtestdatSW3 %>%
    filter(runID==x) %>%  #Select cycle
    arrange(channel)       #Make sure in ascending order based on channel
  counts1<-roll_sum(Pu$X1,n=3)  #Sliding window of three rows
  counts2<-roll_sum(Pu$X2,n=3)
  counts3<-roll_sum(Pu$X3,n=3)
  counts4<-roll_sum(Pu$X4,n=3)
  counts5<-roll_sum(Pu$X5,n=3)
  counts6<-roll_sum(Pu$X6,n=3)
  counts7<-roll_sum(Pu$X7,n=3)
  counts8<-roll_sum(Pu$X8,n=3)
  counts9<-roll_sum(Pu$X9,n=3)
  counts10<-roll_sum(Pu$X10,n=3)
 df<-data.frame(X1=counts1,X2=counts2,X3=counts3, X4=counts4,
                 X5=counts5, X6=counts6, X7=counts7, X8=counts8,
                 X9=counts9, X10=counts10,runID=x) #creates a dataframe
}
SW3<-map_df(cyclesSW3,get_countsSW3) #Apply function to all cycles
################################################################################
#Apply function to get new channels and their labels for sliding window of 3
################################################################################
get_channelsSW3<-function(x){
  Pu2<- newtestdatSW3 %>%
    filter(runID==x) %>%   #Select cycle
    arrange(channel)
  test4<-data.frame(Pu2$channel, new_channel=dplyr::lead(Pu2$channel,1)) #Names
      channels with middle number
}
NC3<-map_df(cyclesSW3,get_channelsSW3) #Apply sliding window of 3 function
################################################################################
#Filter extra values. Filters NC3 so that the extra 40 values are removed and bind
    with main dataframe
################################################################################
Filter3<-NC3 %>%
  filter(!is.na(new_channel)) %>%
  subset(new_channel!=16383) %>%
  select(-c(Pu2.channel))
Final3<-cbind(SW3,Filter3)#This binds the two dataframes
################################################################################
#Feature generation
################################################################################
Final3$mean<-rowMeans(Final3[,1:10]) #Create feature "mean"
Final3<-Final3 %>%
  rowwise() %>%
  mutate(median = median(c(X1,X2,X3,X4,X5,X6,X7,X8,X9,X10), na.rm = TRUE)) #Create
      feature 'median'
Final3$rho<-apply(Final3[,1:10],1,SpearmansScore,t1) #Create feature 'rho'
```

```
#Adds column of chunked rho scores as an additional feature
Final3<- Final3 %>%
  mutate(rho_cat=case_when(rho >=0.7000000 & rho<=0.9999999 ~ 'StrongPos',
                           rho >=0.5000000 & rho <=0.6999999 ~ 'ModeratePos',
                           rho >= 0.3000000 & rho <= 0.4999999 ~ 'WeakPos',
                           rho >= 0.0000000 & rho <=0.2999999 ~ 'NoRelationship',
                           rho <=0.0000000 & rho >= -0.2999999~ 'NoRelationship',
                           rho <= -0.3000000 & rho >= -0.4999999~ 'WeakNeg',
                           rho <=-0.5000000 & rho >= -0.6999999~ 'ModerateNeg',
                           rho <= -0.7000000 & rho>=-0.9999999~ 'StrongNeg',
                           rho == "1" ~'PerfectPos',
                           rho == "-1" ~ 'PerfectNeg'))
Final3<-Final3[c(1,2,3,4,5,6,7,8,9,10,12,11,13,14,15,16)] #Reorder the data frame
    after feature generation is complete
##################################################################################
#Save to folder for merge with Pu files and perform statistical tests for finding
    significant channels
Final3Pu<-Final3
write.csv(Final3Pu,file="Final3Pu.csv")
##################################################################################
#Sliding window of 5. This sums the counts every five rows for every cycle and
    creates a new dataframe.
##################################################################################
newtestdatSW5<-datPu2014 #Create copy of the dataframe (keeps dataframes separate)
cyclesSW5<-unique(newtestdatSW5$runID) #This creates a vector of all cycles
get_countsSW5<-function(x){
  Pu<- newtestdatSW5 %>%
    filter(runID==x) %>%  #Select cycle
    arrange(channel)       #Make sure in ascending order based on channel
  counts1<-roll_sum(Pu$X1,n=5)  #Sliding window of five rows
  counts2<-roll_sum(Pu$X2,n=5)
  counts3<-roll_sum(Pu$X3,n=5)
  counts4<-roll_sum(Pu$X4,n=5)
  counts5<-roll_sum(Pu$X5,n=5)
  counts6<-roll_sum(Pu$X6,n=5)
  counts7<-roll_sum(Pu$X7,n=5)
  counts8<-roll_sum(Pu$X8,n=5)
  counts9<-roll_sum(Pu$X9,n=5)
  counts10<-roll_sum(Pu$X10,n=5)
df<-data.frame(X1=counts1,X2=counts2,X3=counts3, X4=counts4,
                X5=counts5, X6=counts6, X7=counts7, X8=counts8,
                X9=counts9, X10=counts10,runID=x) #Creates a dataframe
}
SW5<-map_df(cyclesSW5,get_countsSW5) #Apply function to all cycles
##################################################################################
#Function for getting new channel labels. Get new channels for sliding window of 5
##################################################################################
get_channelsSW5<-function(x){
  Pu3<- newtestdatSW5 %>%
    filter(runID==x) %>%   #Select cycle
    arrange(channel)
  test4<-data.frame(Pu3$channel, new_channel=dplyr::lead(Pu3$channel,2)) #This would
      name the channels correctly
}
NC5<-map_df(cyclesSW5,get_channelsSW5)
##################################################################################
#Filter extra values. Filters NC5 so that the extra 80 values are removed and bound
    with main dataframe
##################################################################################
Filter5<-NC5 %>%
  filter(!is.na(new_channel)) %>%
  subset(new_channel!=16383) %>%
  subset(new_channel!=16382) %>%
  select(-c(Pu3.channel))
Final5<-cbind(SW5,Filter5)#This binds the two dataframes
##################################################################################
```

```
#Feature generation for SW5
################################################################################
Final5$mean<-rowMeans(Final5[,1:10]) #Create feature "mean"
Final5<-Final5 %>%
  rowwise() %>%
  mutate(median = median(c(X1,X2,X3,X4,X5,X6,X7,X8,X9,X10), na.rm = TRUE)) #Create
      feature 'median'
Final5$rho<-apply(Final5[,1:10],1,SpearmansScore,t1) #Create feature 'rho'
#Adds column of chunked rho scores as an additional feature for SW5
Final5<- Final5 %>%
  mutate(rho_cat=case_when(rho >=0.7000000 & rho<=0.9999999 ~ 'StrongPos',
                           rho >=0.5000000 & rho <=0.6999999 ~ 'ModeratePos',
                           rho >= 0.3000000 & rho <= 0.4999999 ~ 'WeakPos',
                           rho >= 0.0000000 & rho <=0.2999999 ~ 'NoRelationship',
                           rho <=0.0000000 & rho >= -0.2999999~ 'NoRelationship',
                           rho <= -0.3000000 & rho >= -0.4999999~ 'WeakNeg',
                           rho <=-0.5000000 & rho >= -0.6999999~ 'ModerateNeg',
                           rho <= -0.7000000 & rho>=-0.9999999~ 'StrongNeg',
                           rho == "1" ~'PerfectPos',
                           rho == "-1" ~ 'PerfectNeg'))
Final5<-Final5[c(1,2,3,4,5,6,7,8,9,10,12,11,13,14,15,16)] #Reorder the data frame
    after feature generation is complete
################################################################################
#Save to folder for merge with Pu files and perform statistical tests for finding
    significant channels
Final5Pu<-Final5
write.csv(Final5Pu,file="Final5Pu.csv")
```

D Determining Significant Channels

```
################################################################################
#This code merges the HEU and Pu dataframes to apply statistical tests to evaluate
    channels of importance based on features generated. Sliding windows of 3 and 5
    included. Depending on p-value of interest, channels will need to be converted to
     energy utilizing the  energy calibration corresponding to when the data was
    collected. The channels of interest may then be pulled out of unknown data set to
     be run through the trained classifier. Channel conversion was done in excel.
################################################################################
library(tidyverse)
library(qdap)
library(dplyr)
library(zoo)
library(RcppRoll)
library(tidyverse)
library(gdata)
library(gtools)
library(broom)
library(car)
################################################################################
#Read in Files (depends on path you need)
################################################################################
out<-"~/Desktop/TestCodeFiles/SigChannelTest"
setwd(out)
dir(out)
mcsv_r(dir(out))
################################################################################
#Remove extra column
################################################################################
Final3HEU$X<-NULL
Final3Pu$X<-NULL
Final5HEU$X<-NULL
Final5Pu$X<-NULL
################################################################################
#Change the names of columns of interest for both data frames so they are unique to
    the material
################################################################################
colnames(Final3HEU)[colnames(Final3HEU)=="new_channel"] <- "HEU_new_channel"
colnames(Final3HEU)[colnames(Final3HEU)=="runID"] <- "HEU_runID"
colnames(Final3HEU)[colnames(Final3HEU)=="mean"] <- "HEU_mean"
colnames(Final3HEU)[colnames(Final3HEU)=="median"] <- "HEU_median"
colnames(Final3HEU)[colnames(Final3HEU)=="rho"] <- "HEU_rho"
colnames(Final3HEU)[colnames(Final3HEU)=="rho_cat"] <- "HEU_rho_cat"

colnames(Final5HEU)[colnames(Final5HEU)=="new_channel"] <- "HEU_new_channel"
colnames(Final5HEU)[colnames(Final5HEU)=="runID"] <- "HEU_runID"
colnames(Final5HEU)[colnames(Final5HEU)=="mean"] <- "HEU_mean"
colnames(Final5HEU)[colnames(Final5HEU)=="median"] <- "HEU_median"
colnames(Final5HEU)[colnames(Final5HEU)=="rho"] <- "HEU_rho"
colnames(Final5HEU)[colnames(Final5HEU)=="rho_cat"] <- "HEU_rho_cat"

colnames(Final3Pu)[colnames(Final3Pu)=="new_channel"] <- "Pu_new_channel"
colnames(Final3Pu)[colnames(Final3Pu)=="runID"] <- "Pu_runID"
colnames(Final3Pu)[colnames(Final3Pu)=="mean"] <- "Pu_mean"
colnames(Final3Pu)[colnames(Final3Pu)=="median"] <- "Pu_median"
colnames(Final3Pu)[colnames(Final3Pu)=="rho"] <- "Pu_rho"
colnames(Final3Pu)[colnames(Final3Pu)=="rho_cat"] <- "Pu_rho_cat"

colnames(Final5Pu)[colnames(Final5Pu)=="new_channel"] <- "Pu_new_channel"
colnames(Final5Pu)[colnames(Final5Pu)=="runID"] <- "Pu_runID"
colnames(Final5Pu)[colnames(Final5Pu)=="mean"] <- "Pu_mean"
colnames(Final5Pu)[colnames(Final5Pu)=="median"] <- "Pu_median"
colnames(Final5Pu)[colnames(Final5Pu)=="rho"] <- "Pu_rho"
colnames(Final5Pu)[colnames(Final5Pu)=="rho_cat"] <- "Pu_rho_cat"
```

```
##################################################################################
#Eliminate channels above 1.2 MeV due to lack of counts/information and to keep
    continuity of channels for both materials
##################################################################################
Final3HEU<-Final3HEU %>%
  subset(HEU_new_channel <= 3000)
Final3Pu<-Final3Pu %>%
  subset(Pu_new_channel <= 3000)
Final5HEU<-Final5HEU %>%
  subset(HEU_new_channel <= 3000)
Final5Pu<-Final5Pu %>%
  subset(Pu_new_channel <= 3000)
##################################################################################
#Merge HEU and Pu dataframes for statistical tests
##################################################################################
Merged3<-cbind(Final3HEU,Final3Pu)
Merged5<-cbind(Final5HEU,Final5Pu)
write.csv(Merged3,file="Merged3.csv")
write.csv(Merged5,file="Merged5.csv")
##################################################################################
#Apply t-test (Welch's is default in R, assumes variances are unequal) by runID for
    each feature for SW3
##################################################################################
MeanTestStat3 <- Merged3 %>%      #This is for the 95% confidence level
  group_by(HEU_new_channel) %>%
  do(tidy(t.test(.$HEU_mean,
                 .$Pu_mean,
                 mu = 0,
                 alt = "two.sided",
                 paired = FALSE,
                 conf.level = 0.95)))

MedianTestStat3<- Merged3 %>%     #This is for the 95% confidence level
  group_by(HEU_new_channel) %>%
  do(tidy(t.test(.$HEU_median,
                 .$Pu_median,
                 mu = 0,
                 alt = "two.sided",
                 paired = FALSE,
                 conf.level = 0.95)))

RhoTestStat3<-Merged3 %>%         #This is for the 95% confidence level
  group_by(HEU_new_channel) %>%
  do(tidy(t.test(.$HEU_rho,
                 .$Pu_rho,
                 mu = 0,
                 alt = "two.sided",
                 paired = FALSE,
                 conf.level = 0.95)))

RhoTestStat993<-Merged3 %>%       #This is for the 99% confidence level, was not used
     in final analysis
  group_by(HEU_new_channel) %>%
  do(tidy(t.test(.$HEU_rho,
                 .$Pu_rho,
                 mu = 0,
                 alt = "two.sided",
                 paired = FALSE,
                 conf.level = 0.99)))

write.csv(MeanTestStat3,file="MeanTestStat3.csv")
write.csv(MedianTestStat3,file="MedianTestStat3.csv")
write.csv(RhoTestStat3,file="RhoTestStat3.csv")
write.csv(RhoTestStat993,file="RhoTestStat993.csv")
##################################################################################
#Select columns of interest to merge into one dataframe and change p-value label to
```

```
    match specific stat test
##################################################################################
RhoTest993<-as.data.frame(RhoTestStat993) #Turn statistical output data into a
    dataframe to combine dataframes

RhoTest993<- RhoTest993 %>%
  select(HEU_new_channel,p.value) %>%
  rename(Rho_p.value=p.value)

RhoTestStat953<-RhoTestStat3 %>%
  select(HEU_new_channel,p.value)%>%
  rename(Rho95_p.value=p.value)

MedianTestStat3<-MedianTestStat3 %>%
  select(HEU_new_channel,p.value)%>%
  rename(Median_p.value=p.value)

MeanTestStat3<-MeanTestStat3 %>%
  select(HEU_new_channel,p.value)%>%
  rename(Mean_p.value=p.value)
##################################################################################
#Bind dataframes by column, select p-value columns and then merge with original
    dataframe
##################################################################################
TestStatPValues<-bind_cols(RhoTest993, MedianTestStat3, MeanTestStat3,RhoTestStat953)
TestStatPValues<-TestStatPValues %>%
  select(Rho95_p.value,Median_p.value,Mean_p.value) #Did not evaluate 99% confidence
      level
FindSigChannels3<-cbind(TestStatPValues,Merged3) #Merge stat test data output with
    original dataframe
write.csv(FindSigChannels3, file="FindSigChannels3.csv")
##################################################################################
#Determine which channels are significant based on t.test p-value ranging from 1e-02
    to 1e-10
##################################################################################
SigChannels32<-FindSigChannels3 %>%
  subset(Mean_p.value <= 0.01 & Median_p.value <= 0.01 & Rho95_p.value<= 0.01)
write.csv(SigChannels32,file="SigChannels32.csv")

SigChannels33<-FindSigChannels3 %>%
  subset(Mean_p.value <= 0.001 & Median_p.value <= 0.001 & Rho95_p.value<= 0.001)
write.csv(SigChannels33,file="SigChannels33.csv")

SigChannels34<-FindSigChannels3 %>%
  subset(Mean_p.value <= 0.0001 & Median_p.value <= 0.0001 & Rho95_p.value<= 0.0001)
write.csv(SigChannels34,file="SigChannels34.csv")

SigChannels35<-FindSigChannels3 %>%
  subset(Mean_p.value <= 0.00001 & Median_p.value <= 0.00001 & Rho95_p.value<=
      0.00001)
write.csv(SigChannels35,file="SigChannels35.csv")

SigChannels36<-FindSigChannels3 %>%
  subset(Mean_p.value <= 0.000001 & Median_p.value <= 0.000001 & Rho95_p.value<=
      0.000001)
write.csv(SigChannels36,file="SigChannels36.csv")

SigChannels37<-FindSigChannels3 %>%
  subset(Mean_p.value <= 0.0000001 & Median_p.value <= 0.0000001 & Rho95_p.value<=
      0.0000001)
write.csv(SigChannels37,file="SigChannels37.csv")

SigChannels38<-FindSigChannels3 %>%
  subset(Mean_p.value <= 0.00000001 & Median_p.value <= 0.00000001 & Rho95_p.value<=
      0.00000001)
write.csv(SigChannels38,file="SigChannels38.csv")
```

```
SigChannels39<-FindSigChannels3 %>%
  subset(Mean_p.value <= 0.000000001 & Median_p.value <= 0.000000001 & Rho95_p.value
      <= 0.000000001)
write.csv(SigChannels39,file="SigChannels39.csv")

SigChannels310<-FindSigChannels3 %>%
  subset(Mean_p.value <= 0.0000000001 & Median_p.value <= 0.0000000001 & Rho95_p.
      value<= 0.0000000001)
write.csv(SigChannels310,file="SigChannels310.csv")
##################################################################################
#Arrange dataframes based on channel
##################################################################################
SigChannels32<-SigChannels32 %>%
  arrange(HEU_new_channel)

SigChannels33<-SigChannels33 %>%
  arrange(HEU_new_channel)

SigChannels33<-SigChannels33 %>%
  arrange(HEU_new_channel)

SigChannels34<-SigChannels34 %>%
  arrange(HEU_new_channel)

SigChannels35<-SigChannels35 %>%
  arrange(HEU_new_channel)

SigChannels36<-SigChannels36 %>%
  arrange(HEU_new_channel)

SigChannels37<-SigChannels37 %>%
  arrange(HEU_new_channel)

SigChannels38<-SigChannels38 %>%
  arrange(HEU_new_channel)

SigChannels39<-SigChannels39 %>%
  arrange(HEU_new_channel)

SigChannels310<-SigChannels310 %>%
  arrange(HEU_new_channel)
##################################################################################
#Create list of significant channels for energy conversion using energy calibration
    of unknown data
##################################################################################
ChannelEnergy32<-SigChannels32 %>%
  filter(HEU_runID == "HEU1RunID") %>%
  select(HEU_new_channel)
write.csv(ChannelEnergy32,file="ChannelEnergy32.csv")

ChannelEnergy33<-SigChannels33 %>%
  filter(HEU_runID == "HEU1RunID") %>%
  select(HEU_new_channel)
write.csv(ChannelEnergy33,file="ChannelEnergy33.csv")

ChannelEnergy34<-SigChannels34 %>%
  filter(HEU_runID == "HEU1RunID") %>%
  select(HEU_new_channel)
write.csv(ChannelEnergy34,file="ChannelEnergy34.csv")

ChannelEnergy35<-SigChannels35 %>%
  filter(HEU_runID == "HEU1RunID") %>%
  select(HEU_new_channel)
write.csv(ChannelEnergy35,file="ChannelEnergy35.csv")
```

```
ChannelEnergy36<-SigChannels36 %>%
  filter(HEU_runID == "HEU1RunID") %>%
  select(HEU_new_channel)
write.csv(ChannelEnergy36,file="ChannelEnergy36.csv")

ChannelEnergy37<-SigChannels37 %>%
  filter(HEU_runID == "HEU1RunID") %>%
  select(HEU_new_channel)
write.csv(ChannelEnergy37,file="ChannelEnergy37.csv")

ChannelEnergy38<-SigChannels38 %>%
  filter(HEU_runID == "HEU1RunID") %>%
  select(HEU_new_channel)
write.csv(ChannelEnergy38,file="ChannelEnergy38.csv")

ChannelEnergy39<-SigChannels39 %>%
  filter(HEU_runID == "HEU1RunID") %>%
  select(HEU_new_channel)
write.csv(ChannelEnergy39,file="ChannelEnergy39.csv")

ChannelEnergy310<-SigChannels310 %>%
  filter(HEU_runID == "HEU1RunID") %>%
  select(HEU_new_channel)
write.csv(ChannelEnergy310,file="ChannelEnergy310.csv")
##################################################################################
#Pair down dataframe for input into classifier for training
##################################################################################
Train32<-SigChannels32 %>%
  select(HEU_new_channel,HEU_runID,HEU_mean,HEU_median,HEU_rho,HEU_rho_cat,X1,X2,X3,
      X4,X5,X6,X7,X8,X9,X10,
         Pu_runID,Pu_mean,Pu_median, Pu_rho,Pu_rho_cat,X1.1,X2.1,X3.1,X4.1,X5.1,X6.1,
             X7.1,X8.1,X9.1,X10.1)
colnames(Train32)[colnames(Train32)=="HEU_new_channel"] <- "new_channel"
TrainTest32<-Train32 %>%
  arrange(new_channel,HEU_runID)
#Changes all run ID's to material label.Results in 20 instances of each material for
    each channel
TrainTest32$HEU_runID<-"heu"
TrainTest32$Pu_runID<-"pu"
#Pull out columns associated with Pu only
PuTrain32<-TrainTest32 %>%
  select(c(1,17:31))
#Edit column names
colnames(PuTrain32)[colnames(PuTrain32)=="Pu_runID"] <- "class"
colnames(PuTrain32)[colnames(PuTrain32)=="Pu_rho"] <- "rho"
colnames(PuTrain32)[colnames(PuTrain32)=="Pu_rho_cat"] <- "rho_cat"
colnames(PuTrain32)[colnames(PuTrain32)=="Pu_mean"] <- "mean"
colnames(PuTrain32)[colnames(PuTrain32)=="Pu_median"] <- "median"
colnames(PuTrain32)[colnames(PuTrain32)=="X1.1"] <- "X1"
colnames(PuTrain32)[colnames(PuTrain32)=="X2.1"] <- "X2"
colnames(PuTrain32)[colnames(PuTrain32)=="X3.1"] <- "X3"
colnames(PuTrain32)[colnames(PuTrain32)=="X4.1"] <- "X4"
colnames(PuTrain32)[colnames(PuTrain32)=="X5.1"] <- "X5"
colnames(PuTrain32)[colnames(PuTrain32)=="X6.1"] <- "X6"
colnames(PuTrain32)[colnames(PuTrain32)=="X7.1"] <- "X7"
colnames(PuTrain32)[colnames(PuTrain32)=="X8.1"] <- "X8"
colnames(PuTrain32)[colnames(PuTrain32)=="X9.1"] <- "X9"
colnames(PuTrain32)[colnames(PuTrain32)=="X10.1"] <- "X10"
#Pull out columns associated with HEU only
HEUTrain32<-TrainTest32%>%
  select(1:16)
#Edit column names
colnames(HEUTrain32)[colnames(HEUTrain32)=="HEU_runID"] <- "class"
colnames(HEUTrain32)[colnames(HEUTrain32)=="HEU_rho"] <- "rho"
colnames(HEUTrain32)[colnames(HEUTrain32)=="HEU_rho_cat"] <- "rho_cat"
colnames(HEUTrain32)[colnames(HEUTrain32)=="HEU_mean"] <- "mean"
colnames(HEUTrain32)[colnames(HEUTrain32)=="HEU_median"] <- "median"

TrainFinal32<-rbind(HEUTrain32,PuTrain32) #Creates merged dataframe of significant
    channels. Used for training at that p-value threshold
write.csv(TrainFinal32,file="TrainFinal32.csv")
#################################################################################
Train33<-SigChannels33 %>%
  select(HEU_new_channel,HEU_runID,HEU_mean,HEU_median,HEU_rho,HEU_rho_cat,X1,X2,X3,
      X4,X5,X6,X7,X8,X9,X10,
         Pu_runID,Pu_mean,Pu_median, Pu_rho,Pu_rho_cat,X1.1,X2.1,X3.1,X4.1,X5.1,X6.1,
             X7.1,X8.1,X9.1,X10.1)
colnames(Train33)[colnames(Train33)=="HEU_new_channel"] <- "new_channel"
TrainTest33<-Train33 %>%
  arrange(new_channel,HEU_runID)
#Changes all run ID's to material label.Results in 20 instances of each material for
    each channel
TrainTest33$HEU_runID<-"heu"
TrainTest33$Pu_runID<-"pu"
#Pull out columns associated with Pu only
PuTrain33<-TrainTest33 %>%
  select(c(1,17:31))
#Edit column names
colnames(PuTrain33)[colnames(PuTrain33)=="Pu_runID"] <- "class"
colnames(PuTrain33)[colnames(PuTrain33)=="Pu_rho"] <- "rho"
colnames(PuTrain33)[colnames(PuTrain33)=="Pu_rho_cat"] <- "rho_cat"
colnames(PuTrain33)[colnames(PuTrain33)=="Pu_mean"] <- "mean"
colnames(PuTrain33)[colnames(PuTrain33)=="Pu_median"] <- "median"
colnames(PuTrain33)[colnames(PuTrain33)=="X1.1"] <- "X1"
colnames(PuTrain33)[colnames(PuTrain33)=="X2.1"] <- "X2"
colnames(PuTrain33)[colnames(PuTrain33)=="X3.1"] <- "X3"
colnames(PuTrain33)[colnames(PuTrain33)=="X4.1"] <- "X4"
colnames(PuTrain33)[colnames(PuTrain33)=="X5.1"] <- "X5"
colnames(PuTrain33)[colnames(PuTrain33)=="X6.1"] <- "X6"
colnames(PuTrain33)[colnames(PuTrain33)=="X7.1"] <- "X7"
colnames(PuTrain33)[colnames(PuTrain33)=="X8.1"] <- "X8"
colnames(PuTrain33)[colnames(PuTrain33)=="X9.1"] <- "X9"
colnames(PuTrain33)[colnames(PuTrain33)=="X10.1"] <- "X10"
#Pull out columns associated with HEU only
HEUTrain33<-TrainTest33%>%
  select(1:16)
#Edit column names
colnames(HEUTrain33)[colnames(HEUTrain33)=="HEU_runID"] <- "class"
colnames(HEUTrain33)[colnames(HEUTrain33)=="HEU_rho"] <- "rho"
colnames(HEUTrain33)[colnames(HEUTrain33)=="HEU_rho_cat"] <- "rho_cat"
colnames(HEUTrain33)[colnames(HEUTrain33)=="HEU_mean"] <- "mean"
colnames(HEUTrain33)[colnames(HEUTrain33)=="HEU_median"] <- "median"

TrainFinal33<-rbind(HEUTrain33,PuTrain33)  #Creates merged dataframe of significant
    channels. Used for training at that p-value threshold
write.csv(TrainFinal33,file="TrainFinal33.csv")
#################################################################################
Train34<-SigChannels34 %>%
  select(HEU_new_channel,HEU_runID,HEU_mean,HEU_median,HEU_rho,HEU_rho_cat,X1,X2,X3,
      X4,X5,X6,X7,X8,X9,X10,
         Pu_runID,Pu_mean,Pu_median, Pu_rho,Pu_rho_cat,X1.1,X2.1,X3.1,X4.1,X5.1,X6.1,
             X7.1,X8.1,X9.1,X10.1)
colnames(Train34)[colnames(Train34)=="HEU_new_channel"] <- "new_channel"
TrainTest34<-Train34 %>%
  arrange(new_channel,HEU_runID)
#Changes all run ID's to material label. Results in 20 instances of each material for
     each channel
TrainTest34$HEU_runID<-"heu"
TrainTest34$Pu_runID<-"pu"
#Pull out columns associated with Pu only
PuTrain34<-TrainTest34 %>%
  select(c(1,17:31))
```

```
#Edit column names
colnames(PuTrain34)[colnames(PuTrain34)=="Pu_runID"] <- "class"
colnames(PuTrain34)[colnames(PuTrain34)=="Pu_rho"] <- "rho"
colnames(PuTrain34)[colnames(PuTrain34)=="Pu_rho_cat"] <- "rho_cat"
colnames(PuTrain34)[colnames(PuTrain34)=="Pu_mean"] <- "mean"
colnames(PuTrain34)[colnames(PuTrain34)=="Pu_median"] <- "median"
colnames(PuTrain34)[colnames(PuTrain34)=="X1.1"] <- "X1"
colnames(PuTrain34)[colnames(PuTrain34)=="X2.1"] <- "X2"
colnames(PuTrain34)[colnames(PuTrain34)=="X3.1"] <- "X3"
colnames(PuTrain34)[colnames(PuTrain34)=="X4.1"] <- "X4"
colnames(PuTrain34)[colnames(PuTrain34)=="X5.1"] <- "X5"
colnames(PuTrain34)[colnames(PuTrain34)=="X6.1"] <- "X6"
colnames(PuTrain34)[colnames(PuTrain34)=="X7.1"] <- "X7"
colnames(PuTrain34)[colnames(PuTrain34)=="X8.1"] <- "X8"
colnames(PuTrain34)[colnames(PuTrain34)=="X9.1"] <- "X9"
colnames(PuTrain34)[colnames(PuTrain34)=="X10.1"] <- "X10"
#Pull out columns associated with HEU only
HEUTrain34<-TrainTest34%>%
  select(1:16)
#Edit column names
colnames(HEUTrain34)[colnames(HEUTrain34)=="HEU_runID"] <- "class"
colnames(HEUTrain34)[colnames(HEUTrain34)=="HEU_rho"] <- "rho"
colnames(HEUTrain34)[colnames(HEUTrain34)=="HEU_rho_cat"] <- "rho_cat"
colnames(HEUTrain34)[colnames(HEUTrain34)=="HEU_mean"] <- "mean"
colnames(HEUTrain34)[colnames(HEUTrain34)=="HEU_median"] <- "median"

TrainFinal34<-rbind(HEUTrain34,PuTrain34) #Creates merged dataframe of significant
    channels. Used for training at that p-value threshold
write.csv(TrainFinal34,file="TrainFinal34.csv")
###############################################################################
Train35<-SigChannels35 %>%
  select(HEU_new_channel,HEU_runID,HEU_mean,HEU_median,HEU_rho,HEU_rho_cat,X1,X2,X3,
      X4,X5,X6,X7,X8,X9,X10,
         Pu_runID,Pu_mean,Pu_median, Pu_rho,Pu_rho_cat,X1.1,X2.1,X3.1,X4.1,X5.1,X6.1,
             X7.1,X8.1,X9.1,X10.1)
colnames(Train35)[colnames(Train35)=="HEU_new_channel"] <- "new_channel"
TrainTest35<-Train35 %>%
  arrange(new_channel,HEU_runID)
#Changes all run ID's to material label.Results in 20 instances of each material for
    each channel
TrainTest35$HEU_runID<-"heu"
TrainTest35$Pu_runID<-"pu"
#Pull out columns associated with Pu only
PuTrain35<-TrainTest35 %>%
  select(c(1,17:31))
#Edit column names
colnames(PuTrain35)[colnames(PuTrain35)=="Pu_runID"] <- "class"
colnames(PuTrain35)[colnames(PuTrain35)=="Pu_rho"] <- "rho"
colnames(PuTrain35)[colnames(PuTrain35)=="Pu_rho_cat"] <- "rho_cat"
colnames(PuTrain35)[colnames(PuTrain35)=="Pu_mean"] <- "mean"
colnames(PuTrain35)[colnames(PuTrain35)=="Pu_median"] <- "median"
colnames(PuTrain35)[colnames(PuTrain35)=="X1.1"] <- "X1"
colnames(PuTrain35)[colnames(PuTrain35)=="X2.1"] <- "X2"
colnames(PuTrain35)[colnames(PuTrain35)=="X3.1"] <- "X3"
colnames(PuTrain35)[colnames(PuTrain35)=="X4.1"] <- "X4"
colnames(PuTrain35)[colnames(PuTrain35)=="X5.1"] <- "X5"
colnames(PuTrain35)[colnames(PuTrain35)=="X6.1"] <- "X6"
colnames(PuTrain35)[colnames(PuTrain35)=="X7.1"] <- "X7"
colnames(PuTrain35)[colnames(PuTrain35)=="X8.1"] <- "X8"
colnames(PuTrain35)[colnames(PuTrain35)=="X9.1"] <- "X9"
colnames(PuTrain35)[colnames(PuTrain35)=="X10.1"] <- "X10"
#Pull out columns associated with HEU only
HEUTrain35<-TrainTest35%>%
  select(1:16)
#Edit column names
colnames(HEUTrain35)[colnames(HEUTrain35)=="HEU_runID"] <- "class"
colnames(HEUTrain35)[colnames(HEUTrain35)=="HEU_rho"] <- "rho"
colnames(HEUTrain35)[colnames(HEUTrain35)=="HEU_rho_cat"] <- "rho_cat"
colnames(HEUTrain35)[colnames(HEUTrain35)=="HEU_mean"] <- "mean"
colnames(HEUTrain35)[colnames(HEUTrain35)=="HEU_median"] <- "median"

TrainFinal35<-rbind(HEUTrain35,PuTrain35) #Creates merged dataframe of significant
    channels. Used for training at that p-value threshold
write.csv(TrainFinal35,file="TrainFinal35.csv")
##################################################################################
Train36<-SigChannels36 %>%
  select(HEU_new_channel,HEU_runID,HEU_mean,HEU_median,HEU_rho,HEU_rho_cat,X1,X2,X3,
      X4,X5,X6,X7,X8,X9,X10,
         Pu_runID,Pu_mean,Pu_median, Pu_rho,Pu_rho_cat,X1.1,X2.1,X3.1,X4.1,X5.1,X6.1,
             X7.1,X8.1,X9.1,X10.1)
colnames(Train36)[colnames(Train36)=="HEU_new_channel"] <- "new_channel"
TrainTest36<-Train36 %>%
  arrange(new_channel,HEU_runID)
#Changes all run ID's to material label.Results in 20 instances of each material for
    each channel
TrainTest36$HEU_runID<-"heu"
TrainTest36$Pu_runID<-"pu"
#Pull out columns associated with Pu only
PuTrain36<-TrainTest36 %>%
  select(c(1,17:31))
#Edit column names
colnames(PuTrain36)[colnames(PuTrain36)=="Pu_runID"] <- "class"
colnames(PuTrain36)[colnames(PuTrain36)=="Pu_rho"] <- "rho"
colnames(PuTrain36)[colnames(PuTrain36)=="Pu_rho_cat"] <- "rho_cat"
colnames(PuTrain36)[colnames(PuTrain36)=="Pu_mean"] <- "mean"
colnames(PuTrain36)[colnames(PuTrain36)=="Pu_median"] <- "median"
colnames(PuTrain36)[colnames(PuTrain36)=="X1.1"] <- "X1"
colnames(PuTrain36)[colnames(PuTrain36)=="X2.1"] <- "X2"
colnames(PuTrain36)[colnames(PuTrain36)=="X3.1"] <- "X3"
colnames(PuTrain36)[colnames(PuTrain36)=="X4.1"] <- "X4"
colnames(PuTrain36)[colnames(PuTrain36)=="X5.1"] <- "X5"
colnames(PuTrain36)[colnames(PuTrain36)=="X6.1"] <- "X6"
colnames(PuTrain36)[colnames(PuTrain36)=="X7.1"] <- "X7"
colnames(PuTrain36)[colnames(PuTrain36)=="X8.1"] <- "X8"
colnames(PuTrain36)[colnames(PuTrain36)=="X9.1"] <- "X9"
colnames(PuTrain36)[colnames(PuTrain36)=="X10.1"] <- "X10"
#Pull out columns associated with HEU only
HEUTrain36<-TrainTest36%>%
  select(1:16)
#Edit column names
colnames(HEUTrain36)[colnames(HEUTrain36)=="HEU_runID"] <- "class"
colnames(HEUTrain36)[colnames(HEUTrain36)=="HEU_rho"] <- "rho"
colnames(HEUTrain36)[colnames(HEUTrain36)=="HEU_rho_cat"] <- "rho_cat"
colnames(HEUTrain36)[colnames(HEUTrain36)=="HEU_mean"] <- "mean"
colnames(HEUTrain36)[colnames(HEUTrain36)=="HEU_median"] <- "median"

TrainFinal36<-rbind(HEUTrain36,PuTrain36) #Creates merged dataframe of significant
    channels. Used for training at that p-value threshold
write.csv(TrainFinal36,file="TrainFinal36.csv")
##################################################################################
Train37<-SigChannels37 %>%
  select(HEU_new_channel,HEU_runID,HEU_mean,HEU_median,HEU_rho,HEU_rho_cat,X1,X2,X3,
      X4,X5,X6,X7,X8,X9,X10,
         Pu_runID,Pu_mean,Pu_median, Pu_rho,Pu_rho_cat,X1.1,X2.1,X3.1,X4.1,X5.1,X6.1,
             X7.1,X8.1,X9.1,X10.1)
colnames(Train37)[colnames(Train37)=="HEU_new_channel"] <- "new_channel"
TrainTest37<-Train37 %>%
  arrange(new_channel,HEU_runID)
#Changes all run ID's to material label.Results in 20 instances of each material for
    each channel
TrainTest37$HEU_runID<-"heu"
TrainTest37$Pu_runID<-"pu"
```

```
#Pull out columns associated with Pu only
PuTrain37<-TrainTest37 %>%
  select(c(1,17:31))
#Edit column names
colnames(PuTrain37)[colnames(PuTrain37)=="Pu_runID"] <- "class"
colnames(PuTrain37)[colnames(PuTrain37)=="Pu_rho"] <- "rho"
colnames(PuTrain37)[colnames(PuTrain37)=="Pu_rho_cat"] <- "rho_cat"
colnames(PuTrain37)[colnames(PuTrain37)=="Pu_mean"] <- "mean"
colnames(PuTrain37)[colnames(PuTrain37)=="Pu_median"] <- "median"
colnames(PuTrain37)[colnames(PuTrain37)=="X1.1"] <- "X1"
colnames(PuTrain37)[colnames(PuTrain37)=="X2.1"] <- "X2"
colnames(PuTrain37)[colnames(PuTrain37)=="X3.1"] <- "X3"
colnames(PuTrain37)[colnames(PuTrain37)=="X4.1"] <- "X4"
colnames(PuTrain37)[colnames(PuTrain37)=="X5.1"] <- "X5"
colnames(PuTrain37)[colnames(PuTrain37)=="X6.1"] <- "X6"
colnames(PuTrain37)[colnames(PuTrain37)=="X7.1"] <- "X7"
colnames(PuTrain37)[colnames(PuTrain37)=="X8.1"] <- "X8"
colnames(PuTrain37)[colnames(PuTrain37)=="X9.1"] <- "X9"
colnames(PuTrain37)[colnames(PuTrain37)=="X10.1"] <- "X10"
#Pull out columns associated with HEU only
HEUTrain37<-TrainTest37%>%
  select(1:16)
#Edit column names
colnames(HEUTrain37)[colnames(HEUTrain37)=="HEU_runID"] <- "class"
colnames(HEUTrain37)[colnames(HEUTrain37)=="HEU_rho"] <- "rho"
colnames(HEUTrain37)[colnames(HEUTrain37)=="HEU_rho_cat"] <- "rho_cat"
colnames(HEUTrain37)[colnames(HEUTrain37)=="HEU_mean"] <- "mean"
colnames(HEUTrain37)[colnames(HEUTrain37)=="HEU_median"] <- "median"

TrainFinal37<-rbind(HEUTrain37,PuTrain37) #Creates merged dataframe of significant
    channels. Used for training at that p-value threshold
write.csv(TrainFinal37,file="TrainFinal37.csv")
##################################################################################
Train38<-SigChannels38 %>%
  select(HEU_new_channel,HEU_runID,HEU_mean,HEU_median,HEU_rho,HEU_rho_cat,X1,X2,X3,
      X4,X5,X6,X7,X8,X9,X10,
         Pu_runID,Pu_mean,Pu_median, Pu_rho,Pu_rho_cat,X1.1,X2.1,X3.1,X4.1,X5.1,X6.1,
            X7.1,X8.1,X9.1,X10.1)
colnames(Train38)[colnames(Train38)=="HEU_new_channel"] <- "new_channel"
TrainTest38<-Train38 %>%
  arrange(new_channel,HEU_runID)
#Changes all run ID's to material label.Results in 20 instances of each material for
    each channel
TrainTest38$HEU_runID<-"heu"
TrainTest38$Pu_runID<-"pu"
#Pull out columns associated with Pu only
PuTrain38<-TrainTest38 %>%
  select(c(1,17:31))
#Edit column names
colnames(PuTrain38)[colnames(PuTrain38)=="Pu_runID"] <- "class"
colnames(PuTrain38)[colnames(PuTrain38)=="Pu_rho"] <- "rho"
colnames(PuTrain38)[colnames(PuTrain38)=="Pu_rho_cat"] <- "rho_cat"
colnames(PuTrain38)[colnames(PuTrain38)=="Pu_mean"] <- "mean"
colnames(PuTrain38)[colnames(PuTrain38)=="Pu_median"] <- "median"
colnames(PuTrain38)[colnames(PuTrain38)=="X1.1"] <- "X1"
colnames(PuTrain38)[colnames(PuTrain38)=="X2.1"] <- "X2"
colnames(PuTrain38)[colnames(PuTrain38)=="X3.1"] <- "X3"
colnames(PuTrain38)[colnames(PuTrain38)=="X4.1"] <- "X4"
colnames(PuTrain38)[colnames(PuTrain38)=="X5.1"] <- "X5"
colnames(PuTrain38)[colnames(PuTrain38)=="X6.1"] <- "X6"
colnames(PuTrain38)[colnames(PuTrain38)=="X7.1"] <- "X7"
colnames(PuTrain38)[colnames(PuTrain38)=="X8.1"] <- "X8"
colnames(PuTrain38)[colnames(PuTrain38)=="X9.1"] <- "X9"
colnames(PuTrain38)[colnames(PuTrain38)=="X10.1"] <- "X10"
#Pull out columns associated with HEU only
HEUTrain38<-TrainTest38%>%
  select(1:16)
#Edit column names
colnames(HEUTrain38)[colnames(HEUTrain38)=="HEU_runID"] <- "class"
colnames(HEUTrain38)[colnames(HEUTrain38)=="HEU_rho"] <- "rho"
colnames(HEUTrain38)[colnames(HEUTrain38)=="HEU_rho_cat"] <- "rho_cat"
colnames(HEUTrain38)[colnames(HEUTrain38)=="HEU_mean"] <- "mean"
colnames(HEUTrain38)[colnames(HEUTrain38)=="HEU_median"] <- "median"

TrainFinal38<-rbind(HEUTrain38,PuTrain38) #Creates merged dataframe of significant
    channels. Used for training at that p-value threshold
write.csv(TrainFinal38,file="TrainFinal38.csv")
##############################################################################
Train39<-SigChannels39 %>%
  select(HEU_new_channel,HEU_runID,HEU_mean,HEU_median,HEU_rho,HEU_rho_cat,X1,X2,X3,
      X4,X5,X6,X7,X8,X9,X10,
         Pu_runID,Pu_mean,Pu_median, Pu_rho,Pu_rho_cat,X1.1,X2.1,X3.1,X4.1,X5.1,X6.1,
             X7.1,X8.1,X9.1,X10.1)
colnames(Train39)[colnames(Train39)=="HEU_new_channel"] <- "new_channel"
TrainTest39<-Train39 %>%
  arrange(new_channel,HEU_runID)
#Changes all run ID's to material label.Results in 20 instances of each material for
    each channel
TrainTest39$HEU_runID<-"heu"
TrainTest39$Pu_runID<-"pu"
#Pull out columns associated with Pu only
PuTrain39<-TrainTest39 %>%
  select(c(1,17:31))
#Edit column names
colnames(PuTrain39)[colnames(PuTrain39)=="Pu_runID"] <- "class"
colnames(PuTrain39)[colnames(PuTrain39)=="Pu_rho"] <- "rho"
colnames(PuTrain39)[colnames(PuTrain39)=="Pu_rho_cat"] <- "rho_cat"
colnames(PuTrain39)[colnames(PuTrain39)=="Pu_mean"] <- "mean"
colnames(PuTrain39)[colnames(PuTrain39)=="Pu_median"] <- "median"
colnames(PuTrain39)[colnames(PuTrain39)=="X1.1"] <- "X1"
colnames(PuTrain39)[colnames(PuTrain39)=="X2.1"] <- "X2"
colnames(PuTrain39)[colnames(PuTrain39)=="X3.1"] <- "X3"
colnames(PuTrain39)[colnames(PuTrain39)=="X4.1"] <- "X4"
colnames(PuTrain39)[colnames(PuTrain39)=="X5.1"] <- "X5"
colnames(PuTrain39)[colnames(PuTrain39)=="X6.1"] <- "X6"
colnames(PuTrain39)[colnames(PuTrain39)=="X7.1"] <- "X7"
colnames(PuTrain39)[colnames(PuTrain39)=="X8.1"] <- "X8"
colnames(PuTrain39)[colnames(PuTrain39)=="X9.1"] <- "X9"
colnames(PuTrain39)[colnames(PuTrain39)=="X10.1"] <- "X10"
#Pull out columns associated with HEU only
HEUTrain39<-TrainTest39%>%
  select(1:16)
#Edit column names
colnames(HEUTrain39)[colnames(HEUTrain39)=="HEU_runID"] <- "class"
colnames(HEUTrain39)[colnames(HEUTrain39)=="HEU_rho"] <- "rho"
colnames(HEUTrain39)[colnames(HEUTrain39)=="HEU_rho_cat"] <- "rho_cat"
colnames(HEUTrain39)[colnames(HEUTrain39)=="HEU_mean"] <- "mean"
colnames(HEUTrain39)[colnames(HEUTrain39)=="HEU_median"] <- "median"

TrainFinal39<-rbind(HEUTrain39,PuTrain39) #Creates merged dataframe of significant
    channels. Used for training at that p-value threshold
write.csv(TrainFinal39,file="TrainFinal39.csv")
##############################################################################
Train310<-SigChannels310 %>%
  select(HEU_new_channel,HEU_runID,HEU_mean,HEU_median,HEU_rho,HEU_rho_cat,X1,X2,X3,
      X4,X5,X6,X7,X8,X9,X10,
         Pu_runID,Pu_mean,Pu_median, Pu_rho,Pu_rho_cat,X1.1,X2.1,X3.1,X4.1,X5.1,X6.1,
             X7.1,X8.1,X9.1,X10.1)
colnames(Train310)[colnames(Train310)=="HEU_new_channel"] <- "new_channel"
TrainTest310<-Train310 %>%
  arrange(new_channel,HEU_runID)
#Changes all run ID's to material label.Results in 20 instances of each material for
```

```
    each channel
TrainTest310$HEU_runID<-"heu"
TrainTest310$Pu_runID<-"pu"
#Pull out columns associated with Pu only
PuTrain310<-TrainTest310 %>%
  select(c(1,17:31))
#Edit column names
colnames(PuTrain310)[colnames(PuTrain310)=="Pu_runID"] <- "class"
colnames(PuTrain310)[colnames(PuTrain310)=="Pu_rho"] <- "rho"
colnames(PuTrain310)[colnames(PuTrain310)=="Pu_rho_cat"] <- "rho_cat"
colnames(PuTrain310)[colnames(PuTrain310)=="Pu_mean"] <- "mean"
colnames(PuTrain310)[colnames(PuTrain310)=="Pu_median"] <- "median"
colnames(PuTrain310)[colnames(PuTrain310)=="X1.1"] <- "X1"
colnames(PuTrain310)[colnames(PuTrain310)=="X2.1"] <- "X2"
colnames(PuTrain310)[colnames(PuTrain310)=="X3.1"] <- "X3"
colnames(PuTrain310)[colnames(PuTrain310)=="X4.1"] <- "X4"
colnames(PuTrain310)[colnames(PuTrain310)=="X5.1"] <- "X5"
colnames(PuTrain310)[colnames(PuTrain310)=="X6.1"] <- "X6"
colnames(PuTrain310)[colnames(PuTrain310)=="X7.1"] <- "X7"
colnames(PuTrain310)[colnames(PuTrain310)=="X8.1"] <- "X8"
colnames(PuTrain310)[colnames(PuTrain310)=="X9.1"] <- "X9"
colnames(PuTrain310)[colnames(PuTrain310)=="X10.1"] <- "X10"
#Pull out columns associated with HEU only
HEUTrain310<-TrainTest310%>%
  select(1:16)
#Edit column names
colnames(HEUTrain310)[colnames(HEUTrain310)=="HEU_runID"] <- "class"
colnames(HEUTrain310)[colnames(HEUTrain310)=="HEU_rho"] <- "rho"
colnames(HEUTrain310)[colnames(HEUTrain310)=="HEU_rho_cat"] <- "rho_cat"
colnames(HEUTrain310)[colnames(HEUTrain310)=="HEU_mean"] <- "mean"
colnames(HEUTrain310)[colnames(HEUTrain310)=="HEU_median"] <- "median"

TrainFinal310<-rbind(HEUTrain310,PuTrain310) #Creates merged dataframe of significant
     channels. Used for training at that p-value threshold
write.csv(TrainFinal310,file="TrainFinal310.csv")
################################################################################
#Apply t-test (Welch's) by runID for each feature for SW5
################################################################################
MeanTestStat5 <- Merged5 %>%       #This is for the 95% confidence level
  group_by(HEU_new_channel) %>%
  do(tidy(t.test(.$HEU_mean,
                 .$Pu_mean,
                 mu = 0,
                 alt = "two.sided",
                 paired = FALSE,
                 conf.level = 0.95)))
write.csv(MeanTestStat5,file="MeanTestStat5.csv")

MedianTestStat5<- Merged5 %>%      #This is for the 95% confidence level
  group_by(HEU_new_channel) %>%
  do(tidy(t.test(.$HEU_median,
                 .$Pu_median,
                 mu = 0,
                 alt = "two.sided",
                 paired = FALSE,
                 conf.level = 0.95)))
write.csv(MedianTestStat5,file="MedianTestStat5.csv")

RhoTestStat955<-Merged5 %>%        #This is for the 95% confidence level
  group_by(HEU_new_channel) %>%
  do(tidy(t.test(.$HEU_rho,
                 .$Pu_rho,
                 mu = 0,
                 alt = "two.sided",
                 paired = FALSE,
                 conf.level = 0.95)))
```

```
write.csv(RhoTestStat955,file="RhoTestStat955.csv")

RhoTestStat995<-Merged5 %>%        #This is for the 99% confidence level, was not
    used in final analysis
  group_by(HEU_new_channel) %>%
  do(tidy(t.test(.$HEU_rho,
                 .$Pu_rho,
                 mu = 0,
                 alt = "two.sided",
                 paired = FALSE,
                 conf.level = 0.99)))
write.csv(RhoTestStat995,file="RhoTestStat995.csv")
##################################################################################
#Select columns of interest to merge into one dataframe and change p-value label to
    match specific stat test
##################################################################################
RhoTest995<-as.data.frame(RhoTestStat995) #Turn statistical output data into a
    dataframe to combine dataframes

RhoTest995<- RhoTest995 %>%
  select(HEU_new_channel,p.value) %>%
  rename(Rho_p.value=p.value)

RhoTestStat955<-RhoTestStat955 %>%
  select(HEU_new_channel,p.value)%>%
  rename(Rho95_p.value=p.value)

MedianTestStat5<-MedianTestStat5 %>%
  select(HEU_new_channel,p.value)%>%
  rename(Median_p.value=p.value)

MeanTestStat5<-MeanTestStat5 %>%
  select(HEU_new_channel,p.value)%>%
  rename(Mean_p.value=p.value)

TestStatPValues<-bind_cols(RhoTest995, MedianTestStat5, MeanTestStat5,RhoTestStat955)
TestStatPValues<-TestStatPValues %>%
  select(Rho95_p.value,Median_p.value,Mean_p.value)
FindSigChannels5<-cbind(TestStatPValues,Merged5) #Merge stat test data output with
    original dataframe
write.csv(FindSigChannels5, file="FindSigChannels5.csv")
##################################################################################
#Determine which channels are significant based on t.test p-value ranging from 1e-02
    to 1e-10
##################################################################################
SigChannels52<-FindSigChannels5 %>%
  subset(Mean_p.value <= 0.01 & Median_p.value <= 0.01 & Rho95_p.value<= 0.01)
write.csv(SigChannels52,file="SigChannels52")

SigChannels53<-FindSigChannels5 %>%
  subset(Mean_p.value <= 0.001 & Median_p.value <= 0.001 & Rho95_p.value<= 0.001)
write.csv(SigChannels53,file="SigChannels53.csv")

SigChannels54<-FindSigChannels5 %>%
  subset(Mean_p.value <= 0.0001 & Median_p.value <= 0.0001 & Rho95_p.value<= 0.0001)
write.csv(SigChannels54,file="SigChannels54.csv")

SigChannels55<-FindSigChannels5 %>%
  subset(Mean_p.value <= 0.00001 & Median_p.value <= 0.00001 & Rho95_p.value<=
      0.00001)
write.csv(SigChannels55,file="SigChannels55.csv")

SigChannels56<-FindSigChannels5 %>%
  subset(Mean_p.value <= 0.000001 & Median_p.value <= 0.000001 & Rho95_p.value<=
      0.000001)
write.csv(SigChannels56,file="SigChannels56.csv")
```

```
SigChannels57<-FindSigChannels5 %>%
  subset(Mean_p.value <= 0.0000001 & Median_p.value <= 0.0000001 & Rho95_p.value<=
      0.0000001)
write.csv(SigChannels57,file="SigChannels57.csv")

SigChannels58<-FindSigChannels5 %>%
  subset(Mean_p.value <= 0.00000001 & Median_p.value <= 0.00000001 & Rho95_p.value<=
      0.00000001)
write.csv(SigChannels58,file="SigChannels58.csv")

SigChannels59<-FindSigChannels5 %>%
  subset(Mean_p.value <= 0.000000001 & Median_p.value <= 0.000000001 & Rho95_p.value
      <= 0.000000001)
write.csv(SigChannels59,file="SigChannels59.csv")

SigChannels510<-FindSigChannels5 %>%
  subset(Mean_p.value <= 0.0000000001 & Median_p.value <= 0.0000000001 & Rho95_p.
      value<= 0.0000000001)
write.csv(SigChannels510, file="SigChannels510.csv")
#################################################################################
#Arrange dataframe by channel
#################################################################################
SigChannels52<-SigChannels52 %>%
  arrange(HEU_new_channel)

SigChannels53<-SigChannels53 %>%
  arrange(HEU_new_channel)

SigChannels53<-SigChannels53 %>%
  arrange(HEU_new_channel)

SigChannels54<-SigChannels54 %>%
  arrange(HEU_new_channel)

SigChannels55<-SigChannels55 %>%
  arrange(HEU_new_channel)

SigChannels56<-SigChannels56 %>%
  arrange(HEU_new_channel)

SigChannels57<-SigChannels57 %>%
  arrange(HEU_new_channel)

SigChannels58<-SigChannels58 %>%
  arrange(HEU_new_channel)

SigChannels59<-SigChannels59 %>%
  arrange(HEU_new_channel)

SigChannels510<-SigChannels510 %>%
  arrange(HEU_new_channel)
#################################################################################
#Create list of significant channels for energy conversion using energy calibration
    of unknown data
#################################################################################
ChannelEnergy52<-SigChannels52 %>%
  filter(HEU_runID == "HEU1RunID") %>%
  select(HEU_new_channel)
write.csv(ChannelEnergy52,file="ChannelEnergy52.csv")

ChannelEnergy53<-SigChannels53 %>%
  filter(HEU_runID == "HEU1RunID") %>%
  select(HEU_new_channel)
write.csv(ChannelEnergy53,file="ChannelEnergy53.csv")
```

```
ChannelEnergy54<-SigChannels54 %>%
  filter(HEU_runID == "HEU1RunID") %>%
  select(HEU_new_channel)
write.csv(ChannelEnergy54,file="ChannelEnergy54.csv")

ChannelEnergy55<-SigChannels55 %>%
  filter(HEU_runID == "HEU1RunID") %>%
  select(HEU_new_channel)
write.csv(ChannelEnergy55,file="ChannelEnergy55.csv")

ChannelEnergy56<-SigChannels56 %>%
  filter(HEU_runID == "HEU1RunID") %>%
  select(HEU_new_channel)
write.csv(ChannelEnergy56,file="ChannelEnergy56.csv")

ChannelEnergy57<-SigChannels57 %>%
  filter(HEU_runID == "HEU1RunID") %>%
  select(HEU_new_channel)
write.csv(ChannelEnergy57,file="ChannelEnergy57.csv")

ChannelEnergy58<-SigChannels58 %>%
  filter(HEU_runID == "HEU1RunID") %>%
  select(HEU_new_channel)
write.csv(ChannelEnergy58,file="ChannelEnergy58.csv")

ChannelEnergy59<-SigChannels59 %>%
  filter(HEU_runID == "HEU1RunID") %>%
  select(HEU_new_channel)
write.csv(ChannelEnergy59,file="ChannelEnergy59.csv")

ChannelEnergy510<-SigChannels510 %>%
  filter(HEU_runID == "HEU1RunID") %>%
  select(HEU_new_channel)
write.csv(ChannelEnergy510,file="ChannelEnergy510.csv")
###############################################################################
#Pair down dataframe for input into classifier for training
###############################################################################
Train52<-SigChannels52 %>%
  select(HEU_new_channel,HEU_runID,HEU_mean, HEU_median,HEU_rho,HEU_rho_cat,X1,X2,X3,
      X4,X5,X6,X7,X8,X9,X10,
         Pu_runID,Pu_mean,Pu_median, Pu_rho,Pu_rho_cat,X1.1,X2.1,X3.1,X4.1,X5.1,X6.1,
             X7.1,X8.1,X9.1,X10.1)
colnames(Train52)[colnames(Train52)=="HEU_new_channel"] <- "new_channel"
TrainTest52<-Train52 %>%
  arrange(new_channel,HEU_runID)
#Changes all run ID's to material label.Results in 20 instances of each material for
    each channel
TrainTest52$HEU_runID<-"heu"
TrainTest52$Pu_runID<-"pu"
#Pull out columns associated with Pu only
PuTrain52<-TrainTest52 %>%
  select(c(1,17:31))
#Edit column names
colnames(PuTrain52)[colnames(PuTrain52)=="Pu_runID"] <- "class"
colnames(PuTrain52)[colnames(PuTrain52)=="Pu_rho"] <- "rho"
colnames(PuTrain52)[colnames(PuTrain52)=="Pu_rho_cat"] <- "rho_cat"
colnames(PuTrain52)[colnames(PuTrain52)=="Pu_mean"] <- "mean"
colnames(PuTrain52)[colnames(PuTrain52)=="Pu_median"] <- "median"
colnames(PuTrain52)[colnames(PuTrain52)=="X1.1"] <- "X1"
colnames(PuTrain52)[colnames(PuTrain52)=="X2.1"] <- "X2"
colnames(PuTrain52)[colnames(PuTrain52)=="X3.1"] <- "X3"
colnames(PuTrain52)[colnames(PuTrain52)=="X4.1"] <- "X4"
colnames(PuTrain52)[colnames(PuTrain52)=="X5.1"] <- "X5"
colnames(PuTrain52)[colnames(PuTrain52)=="X6.1"] <- "X6"
colnames(PuTrain52)[colnames(PuTrain52)=="X7.1"] <- "X7"
colnames(PuTrain52)[colnames(PuTrain52)=="X8.1"] <- "X8"
colnames(PuTrain52)[colnames(PuTrain52)=="X9.1"] <- "X9"
colnames(PuTrain52)[colnames(PuTrain52)=="X10.1"] <- "X10"
#Pull out columns associated with HEU only
HEUTrain52<-TrainTest52%>%
  select(1:16)
#Edit column names
colnames(HEUTrain52)[colnames(HEUTrain52)=="HEU_runID"] <- "class"
colnames(HEUTrain52)[colnames(HEUTrain52)=="HEU_rho"] <- "rho"
colnames(HEUTrain52)[colnames(HEUTrain52)=="HEU_rho_cat"] <- "rho_cat"
colnames(HEUTrain52)[colnames(HEUTrain52)=="HEU_mean"] <- "mean"
colnames(HEUTrain52)[colnames(HEUTrain52)=="HEU_median"] <- "median"

TrainFinal52<-rbind(HEUTrain52,PuTrain52) #Creates merged dataframe of significant
    channels. Used for training at that p-value threshold
write.csv(TrainFinal52,file="TrainFinal52.csv")
################################################################################
Train53<-SigChannels53 %>%
  select(HEU_new_channel,HEU_runID,HEU_mean,HEU_median,HEU_rho,HEU_rho_cat,X1,X2,X3,
      X4,X5,X6,X7,X8,X9,X10,
         Pu_runID,Pu_mean,Pu_median, Pu_rho,Pu_rho_cat,X1.1,X2.1,X3.1,X4.1,X5.1,X6.1,
             X7.1,X8.1,X9.1,X10.1)
colnames(Train53)[colnames(Train53)=="HEU_new_channel"] <- "new_channel"
TrainTest53<-Train53 %>%
  arrange(new_channel,HEU_runID)
#Changes all run ID's to material label.Results in 20 instances of each material for
    each channel
TrainTest53$HEU_runID<-"heu"
TrainTest53$Pu_runID<-"pu"
#Pull out columns associated with Pu only
PuTrain53<-TrainTest53 %>%
  select(c(1,17:31))
#Edit column names
colnames(PuTrain53)[colnames(PuTrain53)=="Pu_runID"] <- "class"
colnames(PuTrain53)[colnames(PuTrain53)=="Pu_rho"] <- "rho"
colnames(PuTrain53)[colnames(PuTrain53)=="Pu_rho_cat"] <- "rho_cat"
colnames(PuTrain53)[colnames(PuTrain53)=="Pu_mean"] <- "mean"
colnames(PuTrain53)[colnames(PuTrain53)=="Pu_median"] <- "median"
colnames(PuTrain53)[colnames(PuTrain53)=="X1.1"] <- "X1"
colnames(PuTrain53)[colnames(PuTrain53)=="X2.1"] <- "X2"
colnames(PuTrain53)[colnames(PuTrain53)=="X3.1"] <- "X3"
colnames(PuTrain53)[colnames(PuTrain53)=="X4.1"] <- "X4"
colnames(PuTrain53)[colnames(PuTrain53)=="X5.1"] <- "X5"
colnames(PuTrain53)[colnames(PuTrain53)=="X6.1"] <- "X6"
colnames(PuTrain53)[colnames(PuTrain53)=="X7.1"] <- "X7"
colnames(PuTrain53)[colnames(PuTrain53)=="X8.1"] <- "X8"
colnames(PuTrain53)[colnames(PuTrain53)=="X9.1"] <- "X9"
colnames(PuTrain53)[colnames(PuTrain53)=="X10.1"] <- "X10"
#Pull out columns associated with HEU only
HEUTrain53<-TrainTest53%>%
  select(1:16)
#Edit column names
colnames(HEUTrain53)[colnames(HEUTrain53)=="HEU_runID"] <- "class"
colnames(HEUTrain53)[colnames(HEUTrain53)=="HEU_rho"] <- "rho"
colnames(HEUTrain53)[colnames(HEUTrain53)=="HEU_rho_cat"] <- "rho_cat"
colnames(HEUTrain53)[colnames(HEUTrain53)=="HEU_mean"] <- "mean"
colnames(HEUTrain53)[colnames(HEUTrain53)=="HEU_median"] <- "median"

TrainFinal53<-rbind(HEUTrain53,PuTrain53) #Creates merged dataframe of significant
    channels. Used for training at that p-value threshold
write.csv(TrainFinal53,file="TrainFinal53.csv")
################################################################################
Train54<-SigChannels54 %>%
  select(HEU_new_channel,HEU_runID,HEU_mean,HEU_median,HEU_rho,HEU_rho_cat,X1,X2,X3,
      X4,X5,X6,X7,X8,X9,X10,
         Pu_runID,Pu_mean,Pu_median, Pu_rho,Pu_rho_cat,X1.1,X2.1,X3.1,X4.1,X5.1,X6.1,
             X7.1,X8.1,X9.1,X10.1)
```

```
colnames(Train54)[colnames(Train54)=="HEU_new_channel"] <- "new_channel"
TrainTest54<-Train54 %>%
  arrange(new_channel,HEU_runID)
#Changes all run ID's to material label.Results in 20 instances of each material for
    each channel
TrainTest54$HEU_runID<-"heu"
TrainTest54$Pu_runID<-"pu"
#Pull out columns associated with Pu only
PuTrain54<-TrainTest54 %>%
  select(c(1,17:31))
#Edit column names
colnames(PuTrain54)[colnames(PuTrain54)=="Pu_runID"] <- "class"
colnames(PuTrain54)[colnames(PuTrain54)=="Pu_rho"] <- "rho"
colnames(PuTrain54)[colnames(PuTrain54)=="Pu_rho_cat"] <- "rho_cat"
colnames(PuTrain54)[colnames(PuTrain54)=="Pu_mean"] <- "mean"
colnames(PuTrain54)[colnames(PuTrain54)=="Pu_median"] <- "median"
colnames(PuTrain54)[colnames(PuTrain54)=="X1.1"] <- "X1"
colnames(PuTrain54)[colnames(PuTrain54)=="X2.1"] <- "X2"
colnames(PuTrain54)[colnames(PuTrain54)=="X3.1"] <- "X3"
colnames(PuTrain54)[colnames(PuTrain54)=="X4.1"] <- "X4"
colnames(PuTrain54)[colnames(PuTrain54)=="X5.1"] <- "X5"
colnames(PuTrain54)[colnames(PuTrain54)=="X6.1"] <- "X6"
colnames(PuTrain54)[colnames(PuTrain54)=="X7.1"] <- "X7"
colnames(PuTrain54)[colnames(PuTrain54)=="X8.1"] <- "X8"
colnames(PuTrain54)[colnames(PuTrain54)=="X9.1"] <- "X9"
colnames(PuTrain54)[colnames(PuTrain54)=="X10.1"] <- "X10"
#Pull out columns associated with HEU only
HEUTrain54<-TrainTest54%>%
  select(1:16)
#Edit column names
colnames(HEUTrain54)[colnames(HEUTrain54)=="HEU_runID"] <- "class"
colnames(HEUTrain54)[colnames(HEUTrain54)=="HEU_rho"] <- "rho"
colnames(HEUTrain54)[colnames(HEUTrain54)=="HEU_rho_cat"] <- "rho_cat"
colnames(HEUTrain54)[colnames(HEUTrain54)=="HEU_mean"] <- "mean"
colnames(HEUTrain54)[colnames(HEUTrain54)=="HEU_median"] <- "median"

TrainFinal54<-rbind(HEUTrain54,PuTrain54) #Creates merged dataframe of significant
    channels. Used for training at that p-value threshold
write.csv(TrainFinal54,file="TrainFinal54.csv")
##############################################################################
Train55<-SigChannels55 %>%
  select(HEU_new_channel,HEU_runID,HEU_mean,HEU_median,HEU_rho,HEU_rho_cat,X1,X2,X3,
      X4,X5,X6,X7,X8,X9,X10,
         Pu_runID,Pu_mean,Pu_median, Pu_rho,Pu_rho_cat,X1.1,X2.1,X3.1,X4.1,X5.1,X6.1,
             X7.1,X8.1,X9.1,X10.1)
colnames(Train55)[colnames(Train55)=="HEU_new_channel"] <-"new_channel"
TrainTest55<-Train55 %>%
  arrange(new_channel,HEU_runID)
#Changes all run ID's to material label.Results in 20 instances of each material for
    each channel
TrainTest55$HEU_runID<-"heu"
TrainTest55$Pu_runID<-"pu"
#Pull out columns associated with Pu only
PuTrain55<-TrainTest55 %>%
  select(c(1,17:31))
#Edit column names
colnames(PuTrain55)[colnames(PuTrain55)=="Pu_runID"] <- "class"
colnames(PuTrain55)[colnames(PuTrain55)=="Pu_rho"] <- "rho"
colnames(PuTrain55)[colnames(PuTrain55)=="Pu_rho_cat"] <- "rho_cat"
colnames(PuTrain55)[colnames(PuTrain55)=="Pu_mean"] <- "mean"
colnames(PuTrain55)[colnames(PuTrain55)=="Pu_median"] <- "median"
colnames(PuTrain55)[colnames(PuTrain55)=="X1.1"] <- "X1"
colnames(PuTrain55)[colnames(PuTrain55)=="X2.1"] <- "X2"
colnames(PuTrain55)[colnames(PuTrain55)=="X3.1"] <- "X3"
colnames(PuTrain55)[colnames(PuTrain55)=="X4.1"] <- "X4"
colnames(PuTrain55)[colnames(PuTrain55)=="X5.1"] <- "X5"
colnames(PuTrain55)[colnames(PuTrain55)=="X6.1"] <- "X6"
colnames(PuTrain55)[colnames(PuTrain55)=="X7.1"] <- "X7"
colnames(PuTrain55)[colnames(PuTrain55)=="X8.1"] <- "X8"
colnames(PuTrain55)[colnames(PuTrain55)=="X9.1"] <- "X9"
colnames(PuTrain55)[colnames(PuTrain55)=="X10.1"] <- "X10"
#Pull out columns associated with HEU only
HEUTrain55<-TrainTest55%>%
  select(1:16)
#Edit column names
colnames(HEUTrain55)[colnames(HEUTrain55)=="HEU_runID"] <- "class"
colnames(HEUTrain55)[colnames(HEUTrain55)=="HEU_rho"] <- "rho"
colnames(HEUTrain55)[colnames(HEUTrain55)=="HEU_rho_cat"] <- "rho_cat"
colnames(HEUTrain55)[colnames(HEUTrain55)=="HEU_mean"] <- "mean"
colnames(HEUTrain55)[colnames(HEUTrain55)=="HEU_median"] <- "median"

TrainFinal55<-rbind(HEUTrain55,PuTrain55) #Creates merged dataframe of significant
    channels. Used for training at that p-value threshold
write.csv(TrainFinal55,file="TrainFinal55.csv")
##################################################################################
Train56<-SigChannels56 %>%
  select(HEU_new_channel,HEU_runID,HEU_mean,HEU_median,HEU_rho,HEU_rho_cat,X1,X2,X3,
      X4,X5,X6,X7,X8,X9,X10,
         Pu_runID,Pu_mean,Pu_median, Pu_rho,Pu_rho_cat,X1.1,X2.1,X3.1,X4.1,X5.1,X6.1,
             X7.1,X8.1,X9.1,X10.1)
colnames(Train56)[colnames(Train56)=="HEU_new_channel"] <- "new_channel"
TrainTest56<-Train56 %>%
  arrange(new_channel,HEU_runID)
#Changes all run ID's to material label.Results in 20 instances of each material for
    each channel
TrainTest56$HEU_runID<-"heu"
TrainTest56$Pu_runID<-"pu"
#Pull out columns associated with Pu only
PuTrain56<-TrainTest56 %>%
  select(c(1,17:31))
#Edit column names
colnames(PuTrain56)[colnames(PuTrain56)=="Pu_runID"] <- "class"
colnames(PuTrain56)[colnames(PuTrain56)=="Pu_rho"] <- "rho"
colnames(PuTrain56)[colnames(PuTrain56)=="Pu_rho_cat"] <- "rho_cat"
colnames(PuTrain56)[colnames(PuTrain56)=="Pu_mean"] <- "mean"
colnames(PuTrain56)[colnames(PuTrain56)=="Pu_median"] <- "median"
colnames(PuTrain56)[colnames(PuTrain56)=="X1.1"] <- "X1"
colnames(PuTrain56)[colnames(PuTrain56)=="X2.1"] <- "X2"
colnames(PuTrain56)[colnames(PuTrain56)=="X3.1"] <- "X3"
colnames(PuTrain56)[colnames(PuTrain56)=="X4.1"] <- "X4"
colnames(PuTrain56)[colnames(PuTrain56)=="X5.1"] <- "X5"
colnames(PuTrain56)[colnames(PuTrain56)=="X6.1"] <- "X6"
colnames(PuTrain56)[colnames(PuTrain56)=="X7.1"] <- "X7"
colnames(PuTrain56)[colnames(PuTrain56)=="X8.1"] <- "X8"
colnames(PuTrain56)[colnames(PuTrain56)=="X9.1"] <- "X9"
colnames(PuTrain56)[colnames(PuTrain56)=="X10.1"] <- "X10"
#Pull out columns associated with HEU only
HEUTrain56<-TrainTest56%>%
  select(1:16)
#Edit column names
colnames(HEUTrain56)[colnames(HEUTrain56)=="HEU_runID"] <- "class"
colnames(HEUTrain56)[colnames(HEUTrain56)=="HEU_rho"] <- "rho"
colnames(HEUTrain56)[colnames(HEUTrain56)=="HEU_rho_cat"] <- "rho_cat"
colnames(HEUTrain56)[colnames(HEUTrain56)=="HEU_mean"] <- "mean"
colnames(HEUTrain56)[colnames(HEUTrain56)=="HEU_median"] <- "median"

TrainFinal56<-rbind(HEUTrain56,PuTrain56) #Creates merged dataframe of significant
    channels. Used for training at that p-value threshold
write.csv(TrainFinal56,file="TrainFinal56.csv")
##################################################################################
Train57<-SigChannels57 %>%
  select(HEU_new_channel,HEU_runID,HEU_mean,HEU_median,HEU_rho,HEU_rho_cat,X1,X2,X3,
      X4,X5,X6,X7,X8,X9,X10,
         Pu_runID,Pu_mean,Pu_median, Pu_rho,Pu_rho_cat,X1.1,X2.1,X3.1,X4.1,X5.1,X6.1,
             X7.1,X8.1,X9.1,X10.1)
colnames(Train57)[colnames(Train57)=="HEU_new_channel"] <- "new_channel"
TrainTest57<-Train57 %>%
  arrange(new_channel,HEU_runID)
#Changes all run ID's to material label.Results in 20 instances of each material for
    each channel
TrainTest57$HEU_runID<-"heu"
TrainTest57$Pu_runID<-"pu"
#Pull out columns associated with Pu only
PuTrain57<-TrainTest57 %>%
  select(c(1,17:31))
#Edit column names
colnames(PuTrain57)[colnames(PuTrain57)=="Pu_runID"] <- "class"
colnames(PuTrain57)[colnames(PuTrain57)=="Pu_rho"] <- "rho"
colnames(PuTrain57)[colnames(PuTrain57)=="Pu_rho_cat"] <- "rho_cat"
colnames(PuTrain57)[colnames(PuTrain57)=="Pu_mean"] <- "mean"
colnames(PuTrain57)[colnames(PuTrain57)=="Pu_median"] <- "median"
colnames(PuTrain57)[colnames(PuTrain57)=="X1.1"] <- "X1"
colnames(PuTrain57)[colnames(PuTrain57)=="X2.1"] <- "X2"
colnames(PuTrain57)[colnames(PuTrain57)=="X3.1"] <- "X3"
colnames(PuTrain57)[colnames(PuTrain57)=="X4.1"] <- "X4"
colnames(PuTrain57)[colnames(PuTrain57)=="X5.1"] <- "X5"
colnames(PuTrain57)[colnames(PuTrain57)=="X6.1"] <- "X6"
colnames(PuTrain57)[colnames(PuTrain57)=="X7.1"] <- "X7"
colnames(PuTrain57)[colnames(PuTrain57)=="X8.1"] <- "X8"
colnames(PuTrain57)[colnames(PuTrain57)=="X9.1"] <- "X9"
colnames(PuTrain57)[colnames(PuTrain57)=="X10.1"] <- "X10"
#Pull out columns associated with HEU only
HEUTrain57<-TrainTest57%>%
  select(1:16)
#Edit column names
colnames(HEUTrain57)[colnames(HEUTrain57)=="HEU_runID"] <- "class"
colnames(HEUTrain57)[colnames(HEUTrain57)=="HEU_rho"] <- "rho"
colnames(HEUTrain57)[colnames(HEUTrain57)=="HEU_rho_cat"] <- "rho_cat"
colnames(HEUTrain57)[colnames(HEUTrain57)=="HEU_mean"] <- "mean"
colnames(HEUTrain57)[colnames(HEUTrain57)=="HEU_median"] <- "median"

TrainFinal57<-rbind(HEUTrain57,PuTrain57) #Creates merged dataframe of significant
    channels. Used for training at that p-value threshold
write.csv(TrainFinal57,file="TrainFinal57.csv")
##############################################################################
Train58<-SigChannels58 %>%
  select(HEU_new_channel,HEU_runID,HEU_mean,HEU_median,HEU_rho,HEU_rho_cat,X1,X2,X3,
      X4,X5,X6,X7,X8,X9,X10,
         Pu_runID,Pu_mean,Pu_median, Pu_rho,Pu_rho_cat,X1.1,X2.1,X3.1,X4.1,X5.1,X6.1,
             X7.1,X8.1,X9.1,X10.1)
colnames(Train58)[colnames(Train58)=="HEU_new_channel"] <- "new_channel"
TrainTest58<-Train58 %>%
  arrange(new_channel,HEU_runID
#Changes all run ID's to material label.Results in 20 instances of each material for
    each channel
TrainTest58$HEU_runID<-"heu"
TrainTest58$Pu_runID<-"pu"
#Pull out columns associated with Pu only
PuTrain58<-TrainTest58 %>%
  select(c(1,17:31))
#Edit column names
colnames(PuTrain58)[colnames(PuTrain58)=="Pu_runID"] <- "class"
colnames(PuTrain58)[colnames(PuTrain58)=="Pu_rho"] <- "rho"
colnames(PuTrain58)[colnames(PuTrain58)=="Pu_rho_cat"] <- "rho_cat"
colnames(PuTrain58)[colnames(PuTrain58)=="Pu_mean"] <- "mean"
colnames(PuTrain58)[colnames(PuTrain58)=="Pu_median"] <- "median"
colnames(PuTrain58)[colnames(PuTrain58)=="X1.1"] <- "X1"
colnames(PuTrain58)[colnames(PuTrain58)=="X2.1"] <- "X2"
colnames(PuTrain58)[colnames(PuTrain58)=="X3.1"] <- "X3"
colnames(PuTrain58)[colnames(PuTrain58)=="X4.1"] <- "X4"
colnames(PuTrain58)[colnames(PuTrain58)=="X5.1"] <- "X5"
colnames(PuTrain58)[colnames(PuTrain58)=="X6.1"] <- "X6"
colnames(PuTrain58)[colnames(PuTrain58)=="X7.1"] <- "X7"
colnames(PuTrain58)[colnames(PuTrain58)=="X8.1"] <- "X8"
colnames(PuTrain58)[colnames(PuTrain58)=="X9.1"] <- "X9"
colnames(PuTrain58)[colnames(PuTrain58)=="X10.1"] <- "X10"
#Pull out columns associated with HEU only
HEUTrain58<-TrainTest58%>%
  select(1:16)
#Edit column names
colnames(HEUTrain58)[colnames(HEUTrain58)=="HEU_runID"] <- "class"
colnames(HEUTrain58)[colnames(HEUTrain58)=="HEU_rho"] <- "rho"
colnames(HEUTrain58)[colnames(HEUTrain58)=="HEU_rho_cat"] <- "rho_cat"
colnames(HEUTrain58)[colnames(HEUTrain58)=="HEU_mean"] <- "mean"
colnames(HEUTrain58)[colnames(HEUTrain58)=="HEU_median"] <- "median"

TrainFinal58<-rbind(HEUTrain58,PuTrain58) #Creates merged dataframe of significant
    channels. Used for training at that p-value threshold
write.csv(TrainFinal58,file="TrainFinal58.csv")
##################################################################################
Train59<-SigChannels59 %>%
  select(HEU_new_channel,HEU_runID,HEU_mean,HEU_median,HEU_rho,HEU_rho_cat,X1,X2,X3,
      X4,X5,X6,X7,X8,X9,X10,
         Pu_runID,Pu_mean,Pu_median, Pu_rho,Pu_rho_cat,X1.1,X2.1,X3.1,X4.1,X5.1,X6.1,
             X7.1,X8.1,X9.1,X10.1)
colnames(Train59)[colnames(Train59)=="HEU_new_channel"] <- "new_channel"
TrainTest59<-Train59 %>%
  arrange(new_channel,HEU_runID)
#Changes all run ID's to material label.Results in 20 instances of each material for
    each channel
TrainTest59$HEU_runID<-"heu"
TrainTest59$Pu_runID<-"pu"
#Pull out columns associated with Pu only
PuTrain59<-TrainTest59 %>%
  select(c(1,17:31))
#Edit column names
colnames(PuTrain59)[colnames(PuTrain59)=="Pu_runID"] <- "class"
colnames(PuTrain59)[colnames(PuTrain59)=="Pu_rho"] <- "rho"
colnames(PuTrain59)[colnames(PuTrain59)=="Pu_rho_cat"] <- "rho_cat"
colnames(PuTrain59)[colnames(PuTrain59)=="Pu_mean"] <- "mean"
colnames(PuTrain59)[colnames(PuTrain59)=="Pu_median"] <- "median"
colnames(PuTrain59)[colnames(PuTrain59)=="X1.1"] <- "X1"
colnames(PuTrain59)[colnames(PuTrain59)=="X2.1"] <- "X2"
colnames(PuTrain59)[colnames(PuTrain59)=="X3.1"] <- "X3"
colnames(PuTrain59)[colnames(PuTrain59)=="X4.1"] <- "X4"
colnames(PuTrain59)[colnames(PuTrain59)=="X5.1"] <- "X5"
colnames(PuTrain59)[colnames(PuTrain59)=="X6.1"] <- "X6"
colnames(PuTrain59)[colnames(PuTrain59)=="X7.1"] <- "X7"
colnames(PuTrain59)[colnames(PuTrain59)=="X8.1"] <- "X8"
colnames(PuTrain59)[colnames(PuTrain59)=="X9.1"] <- "X9"
colnames(PuTrain59)[colnames(PuTrain59)=="X10.1"] <- "X10"
#Pull out columns associated with HEU only
HEUTrain59<-TrainTest59%>%
  select(1:16)
#Edit column names
colnames(HEUTrain59)[colnames(HEUTrain59)=="HEU_runID"] <- "class"
colnames(HEUTrain59)[colnames(HEUTrain59)=="HEU_rho"] <- "rho"
colnames(HEUTrain59)[colnames(HEUTrain59)=="HEU_rho_cat"] <- "rho_cat"
colnames(HEUTrain59)[colnames(HEUTrain59)=="HEU_mean"] <- "mean"
colnames(HEUTrain59)[colnames(HEUTrain59)=="HEU_median"] <- "median"

TrainFinal59<-rbind(HEUTrain59,PuTrain59) #Creates merged dataframe of significant
    channels. Used for training at that p-value threshold
write.csv(TrainFinal59,file="TrainFinal59.csv")
```

```
##################################################################################
Train510<-SigChannels510 %>%
  select(HEU_new_channel,HEU_runID,HEU_mean,HEU_median,HEU_rho,HEU_rho_cat,X1,X2,X3,
      X4,X5,X6,X7,X8,X9,X10,
         Pu_runID,Pu_mean,Pu_median, Pu_rho,Pu_rho_cat,X1.1,X2.1,X3.1,X4.1,X5.1,X6.1,
             X7.1,X8.1,X9.1,X10.1)
colnames(Train510)[colnames(Train510)=="HEU_new_channel"] <- "new_channel"
TrainTest510<-Train510 %>%
  arrange(new_channel,HEU_runID)
#Changes all run ID's to material label.Results in 20 instances of each material for
    each channel
TrainTest510$HEU_runID<-"heu"
TrainTest510$Pu_runID<-"pu"
#Pull out columns associated with Pu only
PuTrain510<-TrainTest510 %>%
  select(c(1,17:31))
#Edit column names
colnames(PuTrain510)[colnames(PuTrain510)=="Pu_runID"] <- "class"
colnames(PuTrain510)[colnames(PuTrain510)=="Pu_rho"] <- "rho"
colnames(PuTrain510)[colnames(PuTrain510)=="Pu_rho_cat"] <- "rho_cat"
colnames(PuTrain510)[colnames(PuTrain510)=="Pu_mean"] <- "mean"
colnames(PuTrain510)[colnames(PuTrain510)=="Pu_median"] <- "median"
colnames(PuTrain510)[colnames(PuTrain510)=="X1.1"] <- "X1"
colnames(PuTrain510)[colnames(PuTrain510)=="X2.1"] <- "X2"
colnames(PuTrain510)[colnames(PuTrain510)=="X3.1"] <- "X3"
colnames(PuTrain510)[colnames(PuTrain510)=="X4.1"] <- "X4"
colnames(PuTrain510)[colnames(PuTrain510)=="X5.1"] <- "X5"
colnames(PuTrain510)[colnames(PuTrain510)=="X6.1"] <- "X6"
colnames(PuTrain510)[colnames(PuTrain510)=="X7.1"] <- "X7"
colnames(PuTrain510)[colnames(PuTrain510)=="X8.1"] <- "X8"
colnames(PuTrain510)[colnames(PuTrain510)=="X9.1"] <- "X9"
colnames(PuTrain510)[colnames(PuTrain510)=="X10.1"] <- "X10"
#Pull out columns associated with HEU only
HEUTrain510<-TrainTest510%>%
  select(1:16)
#Edit column names
colnames(HEUTrain510)[colnames(HEUTrain510)=="HEU_runID"] <- "class"
colnames(HEUTrain510)[colnames(HEUTrain510)=="HEU_rho"] <- "rho"
colnames(HEUTrain510)[colnames(HEUTrain510)=="HEU_rho_cat"] <- "rho_cat"
colnames(HEUTrain510)[colnames(HEUTrain510)=="HEU_mean"] <- "mean"
colnames(HEUTrain510)[colnames(HEUTrain510)=="HEU_median"] <- "median"

TrainFinal510<-rbind(HEUTrain510,PuTrain510) #Creates merged dataframe of significant
     channels. Used for training at that p-value threshold
write.csv(TrainFinal510,file="TrainFinal510.csv")
```

E Unknown Sample Analysis

```
################################################################################
#Creates the "unknown" data sets from the 2018 data for validation of trained
    classifiers. This is for a sliding window of 5, the window selected for
    validation analysis.
################################################################################
library(tidyverse)
library(qdap)
library(dplyr)
library(zoo)
library(RcppRoll)
library(tidyverse)
library(gdata)
library(gtools)
################################################################################
#Read in Files (depends on path you need)
################################################################################
out<-"~/Desktop/TestCodeFiles/PuTestCode_Validate"
setwd(out)
dir(out)
mcsv_r(dir(out))
################################################################################
#Read in functions
################################################################################
MakeCutOneCycle<-function(df){
  w<-cut(df$millisec,breaks=10,labels=FALSE); #Creates function called
      MakeCutOneCycle; cuts the millisec field into a sequence of ten intervals;
  t<-data.frame(table(df$Bin,by=w));return(t)} #Creates a frequency table for each
      interval
################################################################################
MakeMatrixOneCycle<-{function(cycle){
  for(i in 1:10){s<-subset(cycle,cycle$by==i);
  s$name<-NULL;s$by<-NULL; colnames(s)<-c("Var1",as.character(i));
  if(i==1){a<-s};
  if(i>1){a<-merge(a,s,by="Var1",all=TRUE)}};
  rownames(a)<-a$Var1;a$Var1<-NULL;return(a)}}
################################################################################
SpearmansScore<-function(x,y){
  z<-cor.test(as.numeric(x),as.numeric(y),method="spearman",exact=FALSE);
  return(z$estimate)}
################################################################################
#Read in template file and assign vector to template name. Get rid of first column.
    Only used one template since the decay and ingrowth templates were perfectly
    anticorrelated.
################################################################################
Templates<-read.csv("Templates.csv") #Includes decay and ingrowth template
t1<-Templates[1,] #Decay template
t1$X<-NULL
################################################################################
#Make Temporal interval data frame from the raw time stamped data. Create a data
    frame of interval cuts for one cycle. New file contains channels under 'Var1',
    count under 'Freq', and the time interval under 'by'.
################################################################################
UnknownCuts1<-MakeCutOneCycle(Unknown1m1)
UnknownCuts2<-MakeCutOneCycle(Unknown1m2)
UnknownCuts3<-MakeCutOneCycle(Unknown1m3)
UnknownCuts4<-MakeCutOneCycle(Unknown1m4)
UnknownCuts5<-MakeCutOneCycle(Unknown1m5)
UnknownCuts6<-MakeCutOneCycle(Unknown1m6)
UnknownCuts7<-MakeCutOneCycle(Unknown1m7)
UnknownCuts8<-MakeCutOneCycle(Unknown1m8)
UnknownCuts9<-MakeCutOneCycle(Unknown1m9)
UnknownCuts10<-MakeCutOneCycle(Unknown1m10)
UnknownCuts11<-MakeCutOneCycle(Unknown1m11)
```

```
UnknownCuts12<-MakeCutOneCycle(Unknown1m12)
UnknownCuts13<-MakeCutOneCycle(Unknown1m13)
UnknownCuts14<-MakeCutOneCycle(Unknown1m14)
UnknownCuts15<-MakeCutOneCycle(Unknown1m15)
UnknownCuts16<-MakeCutOneCycle(Unknown1m16)
UnknownCuts17<-MakeCutOneCycle(Unknown1m17)
UnknownCuts18<-MakeCutOneCycle(Unknown1m18)
UnknownCuts19<-MakeCutOneCycle(Unknown1m19)
################################################################################
#Make a matrix from temporal data for each cycle
#Makes interval matrix for each cycle
################################################################################
UnknownMatrix1<-MakeMatrixOneCycle(UnknownCuts1)
UnknownMatrix2<-MakeMatrixOneCycle(UnknownCuts2)
UnknownMatrix3<-MakeMatrixOneCycle(UnknownCuts3)
UnknownMatrix4<-MakeMatrixOneCycle(UnknownCuts4)
UnknownMatrix5<-MakeMatrixOneCycle(UnknownCuts5)
UnknownMatrix6<-MakeMatrixOneCycle(UnknownCuts6)
UnknownMatrix7<-MakeMatrixOneCycle(UnknownCuts7)
UnknownMatrix8<-MakeMatrixOneCycle(UnknownCuts8)
UnknownMatrix9<-MakeMatrixOneCycle(UnknownCuts9)
UnknownMatrix10<-MakeMatrixOneCycle(UnknownCuts10)
UnknownMatrix11<-MakeMatrixOneCycle(UnknownCuts11)
UnknownMatrix12<-MakeMatrixOneCycle(UnknownCuts12)
UnknownMatrix13<-MakeMatrixOneCycle(UnknownCuts13)
UnknownMatrix14<-MakeMatrixOneCycle(UnknownCuts14)
UnknownMatrix15<-MakeMatrixOneCycle(UnknownCuts15)
UnknownMatrix16<-MakeMatrixOneCycle(UnknownCuts16)
UnknownMatrix17<-MakeMatrixOneCycle(UnknownCuts17)
UnknownMatrix18<-MakeMatrixOneCycle(UnknownCuts18)
UnknownMatrix19<-MakeMatrixOneCycle(UnknownCuts19)
################################################################################
#Rename the rownames to 'channel'
################################################################################
UnknownMatrix1$channel<-rownames(UnknownMatrix1)
UnknownMatrix2$channel<-rownames(UnknownMatrix2)
UnknownMatrix3$channel<-rownames(UnknownMatrix3)
UnknownMatrix4$channel<-rownames(UnknownMatrix4)
UnknownMatrix5$channel<-rownames(UnknownMatrix5)
UnknownMatrix6$channel<-rownames(UnknownMatrix6)
UnknownMatrix7$channel<-rownames(UnknownMatrix7)
UnknownMatrix8$channel<-rownames(UnknownMatrix8)
UnknownMatrix9$channel<-rownames(UnknownMatrix9)
UnknownMatrix10$channel<-rownames(UnknownMatrix10)
UnknownMatrix11$channel<-rownames(UnknownMatrix11)
UnknownMatrix12$channel<-rownames(UnknownMatrix12)
UnknownMatrix13$channel<-rownames(UnknownMatrix13)
UnknownMatrix14$channel<-rownames(UnknownMatrix14)
UnknownMatrix15$channel<-rownames(UnknownMatrix15)
UnknownMatrix16$channel<-rownames(UnknownMatrix16)
UnknownMatrix17$channel<-rownames(UnknownMatrix17)
UnknownMatrix18$channel<-rownames(UnknownMatrix18)
UnknownMatrix19$channel<-rownames(UnknownMatrix19)
################################################################################
#Arrange channels and fill in missing channels with NA and then to 0.
################################################################################
Unknown1FILL<- UnknownMatrix1 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Unknown1FILLNoNA<-Unknown1FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Unknown1RunID<-add_column(Unknown1FILLNoNA,runID=1) #Assign runID
write.csv(Unknown1RunID,file= "Unknown1RunID.csv") #Save as .csv
Unknown1RunID<-read.csv(file= "Unknown1RunID.csv") #Read in .csv
Unknown1RunID$X<-NULL #Remove column 'X' due to read/write .csv
```

```
Unknown2FILL<- UnknownMatrix2 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Unknown2FILLNoNA<-Unknown2FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Unknown2RunID<-add_column(Unknown2FILLNoNA,runID=2) #Assign runID
write.csv(Unknown2RunID,file= "Unknown2RunID.csv") #Save as .csv
Unknown2RunID<-read.csv(file= "Unknown2RunID.csv") #Read in .csv
Unknown2RunID$X<-NULL #Remove column 'X' due to read/write .csv

Unknown3FILL<- UnknownMatrix3 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Unknown3FILLNoNA<-Unknown3FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Unknown3RunID<-add_column(Unknown3FILLNoNA,runID=3) #Assign runID
write.csv(Unknown3RunID,file= "Unknown3RunID.csv") #Save as .csv
Unknown3RunID<-read.csv(file= "Unknown3RunID.csv") #Read in .csv
Unknown3RunID$X<-NULL #Remove column 'X' due to read/write .csv

Unknown4FILL<- UnknownMatrix4 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Unknown4FILLNoNA<-Unknown4FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Unknown4RunID<-add_column(Unknown4FILLNoNA,runID=4) #Assign runID
write.csv(Unknown4RunID,file= "Unknown4RunID.csv") #Save as .csv
Unknown4RunID<-read.csv(file= "Unknown4RunID.csv") #Read in .csv
Unknown4RunID$X<-NULL #Remove column 'X' due to read/write .csv

Unknown5FILL<- UnknownMatrix5 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Unknown5FILLNoNA<-Unknown5FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Unknown5RunID<-add_column(Unknown5FILLNoNA,runID=5) #Assign runID
write.csv(Unknown5RunID,file= "Unknown5RunID.csv") #Save as .csv
Unknown5RunID<-read.csv(file= "Unknown5RunID.csv") #Read in .csv
Unknown5RunID$X<-NULL #Remove column 'X' due to read/write .csv

Unknown6FILL<- UnknownMatrix6 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Unknown6FILLNoNA<-Unknown6FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Unknown6RunID<-add_column(Unknown6FILLNoNA,runID=6) #Assign runID
write.csv(Unknown6RunID,file= "Unknown6RunID.csv") #Save as .csv
Unknown6RunID<-read.csv(file= "Unknown6RunID.csv") #Read in .csv
Unknown6RunID$X<-NULL #Remove column 'X' due to read/write .csv

Unknown7FILL<- UnknownMatrix7 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Unknown7FILLNoNA<-Unknown7FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Unknown7RunID<-add_column(Unknown7FILLNoNA,runID=7) #Assign runID
write.csv(Unknown7RunID,file= "Unknown7RunID.csv") #Save as .csv
Unknown7RunID<-read.csv(file= "Unknown7RunID.csv") #Read in .csv
Unknown7RunID$X<-NULL #Remove column 'X' due to read/write .csv

Unknown8FILL<- UnknownMatrix8 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Unknown8FILLNoNA<-Unknown8FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Unknown8RunID<-add_column(Unknown8FILLNoNA,runID=8) #Assign runID
```

```
write.csv(Unknown8RunID,file= "Unknown8RunID.csv") #Save as .csv
Unknown8RunID<-read.csv(file= "Unknown8RunID.csv") #Read in as .csv
Unknown8RunID$X<-NULL #Remove column 'X' due to read/write .csv

Unknown9FILL<- UnknownMatrix9 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Unknown9FILLNoNA<-Unknown9FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Unknown9RunID<-add_column(Unknown9FILLNoNA,runID=9) #Assign runID
write.csv(Unknown9RunID,file= "Unknown9RunID.csv") #Save as .csv
Unknown9RunID<-read.csv(file= "Unknown9RunID.csv") #Read in as .csv
Unknown9RunID$X<-NULL #Remove column 'X' due to read/write .csv

Unknown10FILL<- UnknownMatrix10 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Unknown10FILLNoNA<-Unknown10FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Unknown10RunID<-add_column(Unknown10FILLNoNA,runID=10) #Assign runID
write.csv(Unknown10RunID,file= "Unknown10RunID.csv") #Save as .csv
Unknown10RunID<-read.csv(file= "Unknown10RunID.csv") #Read in as .csv
Unknown10RunID$X<-NULL #Remove column 'X' due to read/write .csv

Unknown11FILL<- UnknownMatrix11 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Unknown11FILLNoNA<-Unknown11FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Unknown11RunID<-add_column(Unknown11FILLNoNA,runID=11) #Assign runID
write.csv(Unknown11RunID,file= "Unknown11RunID.csv") #Save as .csv
Unknown11RunID<-read.csv(file= "Unknown11RunID.csv") #Read in as .csv
Unknown11RunID$X<-NULL #Remove column 'X' due to read/write .csv

Unknown12FILL<- UnknownMatrix12 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Unknown12FILLNoNA<-Unknown12FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Unknown12RunID<-add_column(Unknown12FILLNoNA,runID=12) #Assign runID
write.csv(Unknown12RunID,file= "Unknown12RunID.csv") #Save as .csv
Unknown12RunID<-read.csv(file= "Unknown12RunID.csv") #Read in as .csv
Unknown12RunID$X<-NULL #Remove column 'X' due to read/write .csv

Unknown13FILL<- UnknownMatrix13 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Unknown13FILLNoNA<-Unknown13FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Unknown13RunID<-add_column(Unknown13FILLNoNA,runID=13) #Assign runID
write.csv(Unknown13RunID,file= "Unknown13RunID.csv") #Save as .csv
Unknown13RunID<-read.csv(file= "Unknown13RunID.csv") #Read in as .csv
Unknown13RunID$X<-NULL #Remove column 'X' due to read/write .csv

Unknown14FILL<- UnknownMatrix14 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Unknown14FILLNoNA<-Unknown14FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Unknown14RunID<-add_column(Unknown14FILLNoNA,runID=14) #Assign runID
write.csv(Unknown14RunID,file= "Unknown14RunID.csv") #Save as .csv
Unknown14RunID<-read.csv(file= "Unknown14RunID.csv") #Read in as .csv
Unknown14RunID$X<-NULL #Remove column 'X' due to read/write .csv

Unknown15FILL<- UnknownMatrix15 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Unknown15FILLNoNA<-Unknown15FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Unknown15RunID<-add_column(Unknown15FILLNoNA,runID=15) #Assign runID
write.csv(Unknown15RunID,file= "Unknown15RunID.csv") #Save as .csv
Unknown15RunID<-read.csv(file= "Unknown15RunID.csv") #Read in as .csv
Unknown15RunID$X<-NULL #Remove column 'X' due to read/write as .csv

Unknown16FILL<- UnknownMatrix16 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Unknown16FILLNoNA<-Unknown16FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Unknown16RunID<-add_column(Unknown16FILLNoNA,runID=16) #Assign runID
write.csv(Unknown16RunID,file= "Unknown16RunID.csv") #Save as .csv
Unknown16RunID<-read.csv(file= "Unknown16RunID.csv") #Read in as .csv
Unknown16RunID$X<-NULL #Remove column 'X' due to read/write as .csv

Unknown17FILL<- UnknownMatrix17 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Unknown17FILLNoNA<-Unknown17FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Unknown17RunID<-add_column(Unknown17FILLNoNA,runID=17) #Assign runID
write.csv(Unknown17RunID,file= "Unknown17RunID.csv") #Save as .csv
Unknown17RunID<-read.csv(file= "Unknown17RunID.csv") #Read in as .csv
Unknown17RunID$X<-NULL #Remove column 'X' due to read/write as .csv

Unknown18FILL<- UnknownMatrix18 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Unknown18FILLNoNA<-Unknown18FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Unknown18RunID<-add_column(Unknown18FILLNoNA,runID=18) #Assign runID
write.csv(Unknown18RunID,file= "Unknown18RunID.csv") #Save as .csv
Unknown18RunID<-read.csv(file= "Unknown18RunID.csv") #Read in as .csv
Unknown18RunID$X<-NULL #Remove column 'X' due to read/write as .csv

Unknown19FILL<- UnknownMatrix19 %>%
  arrange(channel)%>%
  mutate(channel = as.numeric(channel))%>%
  complete(channel=seq(from=48,to=16383,by=1))
Unknown19FILLNoNA<-Unknown19FILL %>% mutate_if(is.numeric , replace_na, replace = 0)
Unknown19RunID<-add_column(Unknown19FILLNoNA,runID=19) #aAssign runID
write.csv(Unknown19RunID,file= "Unknown19RunID.csv") #Save as .csv
Unknown19RunID<-read.csv(file= "Unknown19RunID.csv") #Read in as .csv
Unknown19RunID$X<-NULL #Remove column 'X' due to read/write as .csv
##################################################################################
#Make a list of the files you want to load and create an empty dataframe
##################################################################################
Unknowndata2 <- c("Unknown1RunID", "Unknown2RunID", "Unknown3RunID", "Unknown4RunID",
             "Unknown5RunID", "Unknown6RunID", "Unknown7RunID", "Unknown8RunID",
             "Unknown9RunID", "Unknown10RunID", "Unknown11RunID", "Unknown12RunID",
             "Unknown13RunID","Unknown14RunID", "Unknown15RunID","Unknown16RunID",
             "Unknown17RunID", "Unknown18RunID", "Unknown19RunID")
datUnknown2014 <- data.frame() #Create an empty data frame to fill
# Read csv, add a column referring to the runID
# Then combine them into one data folder
for (runID in Unknowndata2) {
  filename = paste(runID, ".csv", sep="")
  t <- read.csv(filename)
  t$runID <- runID
  datUnknown2014<- rbind(datUnknown2014, t)
}
#head(datUnknown2014) #Overview of dataframe as a check
rownames(datUnknown2014)<-NULL #Delete rownames
```

```
datUnknown2014$X<-NULL #Delete column X (channel numbers)
##################################################################################
#Sliding window of 5. This sums the counts every five rows for every cycle and
    creates a new dataframe. Sliding window of 5 presented as is was found to be the
    best for material identification. Easily adapted to any window size.
##################################################################################
newtestdatSW5<-datUnknown2014 #Create copy of the dataframe (keeps dataframes
    separate)
cyclesSW5<-unique(newtestdatSW5$runID) #Creates a vector of all cycles
get_countsSW5<-function(x){
  Unknown<- newtestdatSW5 %>%
    filter(runID==x) %>%  #Select cycle
    arrange(channel)      #Make sure in ascending order based on channel
  counts1<-roll_sum(Unknown$X1,n=5)  #Sliding window of five rows
  counts2<-roll_sum(Unknown$X2,n=5)
  counts3<-roll_sum(Unknown$X3,n=5)
  counts4<-roll_sum(Unknown$X4,n=5)
  counts5<-roll_sum(Unknown$X5,n=5)
  counts6<-roll_sum(Unknown$X6,n=5)
  counts7<-roll_sum(Unknown$X7,n=5)
  counts8<-roll_sum(Unknown$X8,n=5)
  counts9<-roll_sum(Unknown$X9,n=5)
  counts10<-roll_sum(Unknown$X10,n=5)
 df<-data.frame(X1=counts1,X2=counts2,X3=counts3, X4=counts4,
                X5=counts5, X6=counts6, X7=counts7, X8=counts8,
                X9=counts9, X10=counts10,runID=x) #Creates a dataframe
}
SW5<-map_df(cyclesSW5,get_countsSW5) #Apply function to all cycles
##################################################################################
#Function for getting new channel labels. Get new channels for sliding window of 5
##################################################################################
get_channelsSW5<-function(x){
  Unknown3<- newtestdatSW5 %>%
    filter(runID==x) %>%   #Select cycle
    arrange(channel)
  test4<-data.frame(Unknown3$channel, new_channel=dplyr::lead(Unknown3$channel,2)) #
      Names channels with middle number
}
NC5<-map_df(cyclesSW5,get_channelsSW5)
##################################################################################
#Filter extra values. Filters NC5 so that the extra 80 values are removed and bound
    with main dataframe. This needs to be edited if using a sliding window other than
     5. More or less channels need to be filtered depending on size of window. Refer
    to this section in Pu/HEU data pre-processing codes for example of a window of 3
##################################################################################
Filter5<-NC5 %>%
  filter(!is.na(new_channel)) %>%
  subset(new_channel!=16383) %>%
  subset(new_channel!=16382) %>%
  select(-c(Unknown3.channel))
Final5<-cbind(SW5,Filter5)#Binds the two dataframes
##################################################################################
#Feature generation
##################################################################################
Final5$mean<-rowMeans(Final5[,1:10]) #Create feature "mean"
Final5<-Final5 %>%
  rowwise() %>%
  mutate(median = median(c(X1,X2,X3,X4,X5,X6,X7,X8,X9,X10), na.rm = TRUE)) #Create
      feature 'median'
Final5$rho<-apply(Final5[,1:10],1,SpearmansScore,t1) #Create feature 'rho'
#Adds column of chunked rho scores as an additional feature
Final5<- Final5 %>%
  mutate(rho_cat=case_when(rho >=0.7000000 & rho<=0.9999999 ~ 'StrongPos',
                           rho >=0.5000000 & rho <=0.6999999 ~ 'ModeratePos',
                           rho >= 0.3000000 & rho <= 0.4999999 ~ 'WeakPos',
                           rho >= 0.0000000 & rho <=0.2999999 ~ 'NoRelationship',
                           rho <=0.0000000 & rho >= -0.2999999~ 'NoRelationship',
                           rho <= -0.3000000 & rho >= -0.4999999~ 'WeakNeg',
                           rho <=-0.5000000 & rho >= -0.6999999~ 'ModerateNeg',
                           rho <= -0.7000000 & rho>=-0.9999999~ 'StrongNeg',
                           rho == "1" ~'PerfectPos',
                           rho == "-1" ~ 'PerfectNeg'))
FinalUnknown5<-Final5[c(12,11,13:16,1:10)] #Reorder the data frame after feature
    generation is complete
##################################################################################
#This portion changes based on what channels are deemed significant (ie: based on p-
    value threshold). See code "Significant Channels" for determing channels for
    input. Each cycle was filtered as a subset from larger dataframe (1-19). Labeled
    'Pu' just for validation so that user could keep track of the two unknowns. In
    practice, this would truly be an unknown data set utilizing channels pre-
    determined based on p-value threshold.Channel determination of unknown data set
    would be based off of current energy calibration (using reference channels)
#These channels were chosen based on typical rounding protocol from energy
    calibration
FinalUnknownPu<-FinalUnknown5 %>%
  subset(runID=="Unknown19RunID") %>%
  filter(new_channel %in% c("143","144","145","146","241","313","314",
                           "315","909", "910","911", "1003","1004","1005"))
colnames(FinalUnknownPu)[colnames(FinalUnknownPu)=="runID"] <- "class"
FinalUnknownPu$class<-"?" #Strip instance of class label
write.csv(FinalUnknownPu,file="FinalUnknown5Pu19.csv")
```

F Relative Mass Content Estimator

```
#Takes in the integrated spectra for an unknown and the pure reference materials.
    Outputs the percent mass of each material and the percent error. Input values for
     RawUnk, RawA and RawB are in the format: c(1,2,3,4,5,6,7,8,9,10). Could be any
    number of bins/intervals
##############################################################
RawUnk <- c() #Reads in values of unknown spectrum to variable RawUnk
RawA <-c() #Reads in values of first reference spectrum to variable RawA
RawB <- c()  #Reads in values of first reference spectrum to variable RawA
##############################################################
N <- cbind(RawA,RawB) #Binds the two raw ref spectra into one nx2 matrix (n time bins
    )
Tr <-10# length(RawA) #Sets the reference time to the last time bin to maximize
    counting stats
TempA <- (RawA) / RawA[Tr] #Temporal spectra for ref A by dividing each entry by ref
    time value
TempB <- (RawB) / RawB[Tr] #Temporal spectra for ref B by dividing each entry by ref
    time value
TempU <- (RawUnk) / RawUnk[Tr] #Temporal spectra for unknown by dividing each entry
    by ref time value
dA <- (TempA*sqrt((sqrt(RawA)/RawA)^2 + (sqrt(RawA[Tr])/RawA[Tr])^2)) #Propagates
    error for TempA
dB <- (TempB*sqrt((sqrt(RawB)/RawB)^2 + (sqrt(RawB[Tr])/RawB[Tr])^2)) #Propagates
    error for TempB
dU <- (TempU*sqrt((sqrt(RawUnk)/RawUnk)^2 + (sqrt(RawUnk[Tr])/RawUnk[Tr])^2)) #Error
    for TempU
dA[is.na(dA)] <- 0 #Setting 0/0 values to 0 instead of NaN, which can kill a
    calculation
dB[is.na(dB)] <- 0 #Setting 0/0 values to 0 instead of NaN, which can kill a
    calculation
dU[is.na(dU)] <- 0 #Setting 0/0 values to 0 instead of NaN, which can kill a
    calculation
S <- cbind(TempA,TempB) #Binds the two temporal vectors into one nx2 matrix (n time
    bins)
s <- svd(S) #Singular Value Decomposition of the source matrix S
DA <- diag(dA) #Creates a diagonal covariance matrix for the error in RawA
DSigA <- t(s$u) %*% DA %*% s$u #Converts covariance into A orthogonal space for
    material A
DB <- diag(dB) #Creates a diagonal covariance matrix for the error in RawB
DSigB <- t(s$u) %*% DB %*% s$u #Converts covariance into A orthogonal space for
    material B
DU <- diag(dU) #Creates a diagonal covariance matrix for the error in RawUnk
DSigU <- t(s$u) %*% DU %*% s$u #Converts covariance into A orthogonal space for
    unknown material
A <- t(s$u) %*% S #Assembling A from the right orthonal matrix from SVD of S
W <- diag(c(1,1)) #Assembling the composition matrix, 100% U and 100% Pu
w <- svd(W) #Taking the SVD of W to use to get the pseudo-inverse of W, Wstar
WDstar <- diag(1/w$d) #Intermediate step creating the inverse eigenvector of W
    ######(these never change?)
Wstar <- w$v %*% WDstar %*% t(w$u) #The pseudo-inverse of W
C <- A %*% Wstar #Assembing transformation matrix C
DSigCA <- DSigA %*% Wstar #Propagating the error from A to C, treating Wstar as a
    constant
DSigCB <- DSigB %*% Wstar #Propagating the error from A to C, treating Wstar as a
    constant
c <- svd(C) #Taking SVD of C to eventually find pseudo-inverse of C, Cstar
CDstar <- diag(1/c$d) #Intermediate step creating the inverse eigenvector of c
Cstar <- c$v %*% CDstar %*% t(c$u) #Assembling the pseudo-inverse of C, Cstar
DSigdCA <- t(c$u) %*% DSigCA %*% c$v #Propagating the error from C to the eigenvector
     of C
DSigdCB <- t(c$u) %*% DSigCB %*% c$v #Propagating the error from C to the eigenvector
     of C
DSigdCAstar <- -CDstar %*% DSigdCA %*% CDstar #Propagating error to inverse of
    eigenvector of C
DSigdCBstar <- -CDstar %*% DSigdCB %*% CDstar #Propagating error to inverse of
```

```
    eigenvector of C
DCAstar <- c$v %*% DSigdCAstar %*% t(c$u) #Propagaing error to the pseudo-inverse of
    C, Cstar
DCBstar <- c$v %*% DSigdCBstar %*% t(c$u) #Propagaing error to the pseudo-inverse of
    C, Cstar
au <- t(s$u) %*% TempU #Bringing the unknown material temporal spectrum into the
    orthogonal space
WUnk <- Cstar %*% au #Calculating the estimated fissile material content
DWUnk <- cbind(DCAstar %*% au,DCBstar %*% au) + Cstar %*% DSigU #Propagating error to
     final estimate
dWUnk <- abs(mean(DWUnk)) #Taking expectated value of propagated covariance matrix to
     find error
Material_1 <- (WUnk[1] * N[Tr,2])/(N[Tr,1]-WUnk[1]*N[Tr,1]+WUnk[1]*N[Tr,2]) #
    Unpacking estimate for M1
Material_2 <- 1 - Material_1 #Simple calculation of M2, assuming M1 + M2 = 1
Weights <- cbind(Material_1,Material_2)*100 #Turning it into a percentage
sigmaN2 <- (WUnk[1] * N[Tr,2])^2 * ((dWUnk/WUnk[1])^2 + N[Tr,2]/(N[Tr,2])^2) #
    Nominator of error prop
sigmaD2 <- sigmaN2 + (WUnk[2] * N[Tr,1])^2 * ((dWUnk/WUnk[2])^2 + N[Tr,1]/(N[Tr,1])
    ^2) #Denominator
Sigma <- Material_1 * sqrt(sigmaN2/(WUnk[1] *
                                     N[Tr,2])^2 + sigmaD2/(WUnk[1] * N[Tr,2] + WUnk
                                         [2] * N[Tr,1])^2)

Error <- Sigma*100 #Turning the error value into a percentage as well
print (c(Weights, Error)) #Printing out the final values:  Material 1   Material 2
    Error
```