Evaluation of Functional Data Clustering Algorithms on Renogram Curves to Aid in the Diagnosis of Kidney Obstruction

By

Kevin Park

Master of Science in Public Health

Department of Biostatistics

Evaluation of Functional Data Clustering Algorithms on Renogram Curves to Aid in the Diagnosis of Kidney Obstruction

By

Kevin Park

B.A in Mathematics
Emory University
2017

Thesis Committee Chair: Amita Manatunga, PHD

Abstract

Evaluation of Functional Data Clustering Algorithms on Renogram Curves
to Aid in the Diagnosis of Kidney Obstruction

By Kevin Park

Kidney obstruction is a serious condition where the body’s urinary system develops resistance to urine outflow. Radionuclide renal imaging plays a major role in evaluating a kidney with suspected obstruction. Current practices model 99mTc-mercaptoacetyltriglycine (MAG3) gamma tracer concentration across time through a renogram curve to help in the diagnosis of kidney obstruction. One important issue that exists in interpreting kidney obstruction is that there are high misclassification rates of whether a kidney is obstructed or not because of the lack of training in diuretic renography among radiologists. The objective of this work is to provide another perspective on the statistical and computer-assisted diagnosis for kidney obstruction by assessing functional data clustering methods in the classification of renogram curves. We considered seven existing algorithms with the training dataset (N=147) where 23.80% of the renogram curves were from obstructed kidneys. We first use the training dataset to evaluate the unsupervised clustering algorithms against the consensus on kidney obstruction from three experts. We then evaluate the accuracy of prediction by using another dataset where we predict the obstruction status for each kidney from clustering methods developed with the training data. We also assess the performance of the best performing clustering methods in a group of kidneys which are difficult to interpret. The clustering algorithms of fscm, waveclust, and itersubspace provide reasonable results for the training set with a kappa of 0.7917, 0.7322, and 0.522 respectively. These three methods resulted with a sensitivity of 82.86%, 77.14%, and 91.43% and specificity of 95.54%, 94.64%, and 74.11% in the training set respectively. With the validation set, we find that only the two algorithms of fscm and waveclust perform strongly with a kappa of 0.7273 and 0.6957, sensitivity of 75.00% and 100.00%, and specificity of 95.00% and 80.00% respectively. With the difficult renogram curves, we saw that fscm and waveclust gave ratings most similar to expert one while often not aligning with the majority expert rating. These two algorithms have shown potential to separate obstructed and unobstructed kidneys. Studies with larger sample sizes may provide further insight to the success of these algorithms in assisting kidney interpretations.

Evaluation of Functional Data Clustering Algorithms on Renogram Curves to Aid in the Diagnosis of Kidney Obstruction

By

Kevin Park

B.A in Mathematics
Emory University
2017

Thesis Committee Chair: Amita Manatunga, PHD

A thesis submitted to the Faculty of the
Rollins School of Public Health of Emory University
in partial fulfillment of the requirements for the degree of
Master of Science in Public Health
in Biostatistics
2018

Acknowledgements

This thesis would not have been possible without the support of my thesis advisor Dr. Amita Manatunga so I acknowledge and thank her very much for her time and help. I also want to express sincere gratitude to Emory PHD candidate Jeong Hoon Jang who provided a tremendous amount of help by meeting with me multiple times throughout the year and by answering my questions throughout my analysis. I also want to express appreciation to Dr. Qi Long for being my reader for this thesis and helping me quickly and proactively.

This thesis is a reminder of a very important life lesson that is worth repeating that “no one achieves anything alone.” I owe the completion of this thesis and the completion of my graduate school career to my family, friends, Department of Biostatistics colleagues, and Emory University.

Table of Contents

Introduction

Five out of one thousand adults experience obstructed uropathy and hydronephrosis with underlying causes originating from an obstructed kidney (Mujoomdar & Dionne, 2012). A kidney biologically becomes obstructed when the body's urinary system develops resistance to urine outflow and produces a build-up of urine (National Kidney Foundation, 2017). Although the prevalence of obstructed kidneys may be relatively low, kidney obstruction is still a major concern as an untreated, obstructed kidney will irreversibly lose function and increase the risk of renal failure. Renal failure is a serious condition because the kidney is unable to filter the excess waste material found in blood (Urology Care Foundation, 2018) and may lead to the requirement of chronic dialysis or a renal transplant.

Radionuclide renal imaging plays a major role to help radiologists and nuclear medicine experts make a diagnosis for an obstructed kidney. The standard procedure to evaluate suspected kidney obstruction is to conduct baseline radionuclide imaging after the gamma emitting tracer, ^{99m}Tc-mercaptoacetyltriglycine (MAG3), is intravenously injected into the blood (O'Reilly, 1996). In a normal kidney, MAG3 is rapidly removed from the blood, flown into the bladder, and is then excreted in the urine. However, an obstructed kidney cannot adequately extract MAG3 from the blood. This biological observation provides motivation to model the concentration of the MAG3 tracer to help in the diagnosis of kidney obstruction. MAG3 concentration is specifically modeled by a time activity curve (renogram curve) where the number of photons from the MAG3 tracer can be counted over a 20 to 24 minute time period using a region of interest assigned around the kidney. If a patient is suspected to have an obstructed kidney from a baseline MAG3 imaging scan, an additional 20 minutes of radionuclide renal imaging are conducted with the administration of furosemide (O'Reilly, 1996). Current practices utilize the analysis of the renal images, renogram curves, and summary measures of the renogram curve to interpret the radionuclide renal scans.

Taylor (2014) address the major challenge of correctly interpreting renogram curves in the field of radionuclide renal imaging and diuresis renography. One issue that exists in diagnosing kidney obstruction is that there are high misclassification rates of whether a kidney is obstructed or not because of the lack of training and experience in diuretic renography among radiologists (Bao et al., 2011). Specifically, Taylor and Garcia (2014) note that time constraints contribute to the lack of training as radiologist residents spend only four months to assimilate a 36-month competency stated under the American Nuclear Medicine Board. In addition, the difficulty of obtaining consistent diagnosis of kidney obstruction increases as "an estimated 590,000 renal scans performed annually in the United States are interpreted at sites that perform fewer than 3 studies per week" (Taylor & Garcia, 2014). The nature of classifying an obstructed kidney is tricky as even professional nuclear medicine physicians disagree about 20% of all kidney cases (Bao et al., 2011). These limitations found in current clinical practices of interpreting renogram curves call for the need of a new mechanism to help radiologists and nuclear medicine experts make a better decision when diagnosing an obstructed kidney.

Several authors have developed a decision support system (RENEX) to help diagnose the obstruction of the kidney based on quantitative measures derived from the renogram curve (Taylor et al., 2008). Bao et al. (2011) proposes a regression framework in identifying important variables that are derived from renogram curves to predict a kidney's obstruction status. However, Taylor (2014) cautions against the practice of over relying on a few parameters in the renogram scans, such as the half-time point, when analyzing renogram curve for suspected obstruction. All these manuscripts use summary measures and variables derived from the renogram curve to conduct interpretation of kidney obstruction. However, they do not fully take into consideration of the functional nature of the renogram curve. Interestingly, Taylor (2014) mentions the importance of the renogram curve in providing functional data to assist in the diagnosis and management of patients. Hence, the purpose of this thesis is to provide another perspective on the statistical and computer-assisted diagnosis for kidney obstruction by assessing functional data clustering methods in the classification of renogram curves. Specifically, we plan to evaluate the capabilities of existing functional data clustering algorithms in separating the obstructed and non-obstructed renogram curves. The functional data clustering algorithms that will be investigated include k-means clustering (MacQueen, 1967), fitfclust (James & Sugar, 2003), distclust (Peng & Muller, 2008), iterSubspace (Chiou & Ling Li, 2007), funclust (Jacques & Preda, 2013), waveclust (Giacofci et al. , 2013), and fscm (Jiang & Serban, 2012). These functional data clustering methods will utilize the strengths of clustering analysis and adapt to analyze the functional renogram curves. All these algorithms are developed for unsupervised clustering, which means that the information regarding the kidney obstruction status is not used for clustering. These functional data clustering algorithms can be more successful than standard clustering procedures by utilizing complex mathematical splines or wavelength bases to account for the functional structure and shape of the renogram curves. Through the unsupervised nature and goal of clustering, we hope to find a functional data clustering method that will best separate obstructed and not obstructed kidney renogram curves while simultaneously ensuring that renogram curves in the same group are most similar to one another. If successful, these clustering algorithms can help radiologists and nuclear medicine experts make a more accurate diagnosis for suspected kidney obstruction.

To accomplish this goal, we first apply and evaluate the above seven algorithms to a dataset of 147 subjects where each subject's renogram curve was rated by three experts and resulted in a full consensus as obstructed or non-obstructed. In addition, we evaluate the accuracy rates of prediction by using another dataset of 28 randomly selected subjects where we predict the obstruction status for each kidney from clustering methods developed with the training data. The 28 subject's renogram curves also had three experts give a consensus rating as obstructed or not obstructed. Third, we apply the strongest performing algorithms to a special group of 28 subjects to determine how the algorithms perform with difficult renogram cases. These 28 subjects comprise the hard-to-interpret group as there was no consensus on obstructed or not obstructed among the three experts.

Data

The data that was utilized for this study contains both baseline and diuretic renogram curve measurements across 99 time points for 108 patient's right and left kidney. Hence, the 108 patients provide a dataset of 216 renogram curves. The patients also compose of 54 males and 54 females with a mean age of 57.16 years.

Specifically, MAG3 concentrations were measured across 59 time points for each patient's baseline renogram curve and furosemide concentrations were measured across 40 time points for the patient's diuretic renogram curves. Along with a patient identification system, each patient's renogram curve was examined and rated by three experts. The three experts were asked to score each patient's renogram curve from a scale of zero to two where zero indicates no kidney obstruction, one indicates an equivocal result, and two is a score to represent an obstructed kidney.

Methods

Seven different unsupervised clustering methods were evaluated in this study. Each clustering method requires a fixed a-priori number of clusters to be specified before analysis so two clusters were pre-specified in this study, one for obstruction and one for not obstructed classification. The seven clustering methods that were tested include k-means clustering (MacQueen, 1967), fitfclust (James & Sugar, 2003), distclust (Peng & Muller, 2008), iterSubspace (Chiou & Ling Li, 2007), funclust (Jacques & Preda, 2013), waveclust (Giacofci et al. , 2013), and fscm (Jiang & Serban, 2012) .

Clustering Method Backgrounds

I. K-Means Clustering (MacQueen, 1967)

The k-means clustering procedure is an unsupervised learning method that is highly used because of its relatively fast and simple procedure for clustering. After specifying the number of clusters (k) in k-means, k data points are placed onto the space and are assigned as the initial centroids. (Naik, 2010) Each data point is then classified onto each centroid with the calculation of the Euclidean distance. The data point is then assigned to the centroid that corresponds to the smallest Euclidean distance. After a data point has been assigned to a centroid, the centroid is recalculated with the new data points that are added and this process of assigning data points to centers repeats until all data points are utilized.

The function that is minimized during K-means can be shown as:

$$\text{Distance} = \sum_{j=1}^{k} \sum_{i=1}^{n} \left|\left|x_i^j - c_j\right|\right|^2$$

where i = 1,…n and j = 1,…,k and n is the sample size and k is the number of centers and $\left|\left| x_i^j - c_j \right|\right|^2$ is the Euclidean distance between a data point x_i^j and the cluster center c_j (Naik, 2010).

Although Macqueen states how k-means is “efficient in the sense of within-class variance” and is economically and computationally feasible, k-means does contain some shortcomings (1967). One limitation of this procedure is its sensitivity to the initial means and number of centers chosen after every iteration of k-means clustering (Ortega et al., 2009). This sensitivity affects the consistent accuracy of the k-means classification across different iterations. Another disadvantage of the K-means clustering algorithm is that it is not resistant to outliers and that overlapping data are not handled very well (Ortega et al., 2009). Under this limitation, k-means may cluster data inadequately because of the close proximity of one data point to another. K-means is also only applicable to numerical variables and is not robust to categorical variables (Ortega et al., 2009). The weaknesses present in the k-means procedure provide motivation to assess the functional data clustering procedure in the classification task.

II. Fitfclust (James & Sugar, 2003)

Fitfclust is a flexible model-based clustering procedure that is “particularly effective when the observations are sparse, irregularly spaced, or occur at different time points for each subject” (James & Sugar, 2003). In the fitfclust model, suppose $g_i(t)$ is the true value of an individual’s curve at time point t. Then the observed values of Y_i can then be expressed as:

$$Y_i = g_i + \varepsilon_i, \quad \text{for } i = 1, \ldots, n$$

where n is the total number of individuals, g_i is a vector of true values, and ε_i are measurement errors at time t (James & Sugar, 2003).

Each curve in fitfclust is modeled by using natural cubic splines to impose structure onto each curve. Hence, each curve can be expressed as:

$$g_i(t) = s(t)^T\gamma_i$$

where s(t) is a p-dimensional spline basis vector and γ_i is a vector of spline coefficients modeled using a Gaussian distribution (James & Sugar, 2003).

With this formulation, the fitfclust model can be expressed as:

$$Y_{i\,=}\ S_i(\mu_k + \gamma_i) + \varepsilon_i\ ,\ i = 1,\ \ldots,\ n,\ \varepsilon_i \sim N(0, R)\ ,\ \gamma_i \sim N(0, \Gamma)$$

where $\mu_k = \lambda_o + \Lambda\alpha_k$ where λ_o and α_k are p and h-dimensional vectors and Λ is a p x h matrix and where $S_i = (s_i(t_{i1}), \ldots , s(t_{in}))^T$ is the spline basis matrix for ith curve (James & Sugar, 2003).

To avoid confounding of the λ_o ,Λ, α_k parameters, a constraint is imposed such that

$$\Sigma\alpha_k = 0 \text{ and } \Lambda^T S^T \Sigma^{-1} S\Lambda = I$$ (James & Sugar, 2003).

One unique aspect of this clustering method is that a random-effects model for the coefficients is utilized "instead of treating the basis coefficients as parameters and fitting a separate spline curve for each individual" when using basis functions for dimension reduction (James & Sugar, 2003). Incorporating the random effects in this way "allows strength to be borrowed across curves, providing superior results for data containing" sparseness (James & Sugar, 2003). The second unique feature to this clustering method is the formulation of the mean with additional parameters of = λ_o , Λ, and α_k. Allowing the mean to be expressed with these parameters reduces the number of parameters that need to be estimated and allows for a low-dimensional representation of each curve to be clustered (James & Sugar, 2003).

Along with the strengths, this method still contains limitations with regards to model selection. As this model utilizes spline features, the choice of which spline basis to use is still an active area of research. Choosing the dimensions of the cluster mean space may also not be clear depending on the research question which can affect clustering in fitfclust (James & Sugar, 2003). However, the strength of fitfclust's clustering method to work with "sparse and irregularly spaced" functional data make it a robust method.

III. Itersubspace (Chiou & Ling Li, 2007)

An important assumption of the itersubspace procedure is that functions of the outcome are independently sampled from a "mixture of stochastic processes" and are "associated with a random cluster variable" (Chiou and Ling Li, 2013). Hence, the clusters centers under itersubspace "are stochastic structures consisting of the cluster means and eigenfunctions" that allow for the curve projection to fit onto the "functional principal component (FPC) subspaces of individual clusters" (Chiou and Ling Li, 2013).

The itersubspace clustering method uniquely "accounts for both the means and the modes of variation differentials between clusters by predicting cluster membership with a reclassification step." The reclassification step involves a "non-parametric iterative mean and covariance updating scheme" to "estimate cluster structures which then allows the cluster membership predictions" to be "based on a non-parametric random-effect model of the truncated Karhunen–Loève expansion" (Chiou and Ling Li, 2013).

Specifically, the clustering of the FPCs can provide the structure of the mean and covariance such that the predicted curve can be expressed with the non-parametric random-effects model:

$$\tilde{Y}^{(c)}(t) = \mu^{(c)}(t) + \sum_{j=1}^{M_c} \varepsilon_j^{(c)}(Y)\, p_j^{(c)}(t)$$

Where p_j are eigenfunctions, μ is the marginal mean of the mixture process, and ε_j is the random coefficients (Chiou and Ling Li, 2013).

Maximizing the conditional probability here

$$c^*(y) = \arg\max \{ P_{C|Y}(c|y)\} \text{ where } c\ \varepsilon\ \{1,\ldots.,k\}$$

determines the best predicted cluster membership for each curve (Chiou and Ling Li, 2013).

One important assumption that is made however is equal within-cluster variation. Unlike other clustering methods which utilize basis functions when modeling the curves, the itersubspace method uses "cluster eigenbases for process expansion" which is estimated by the data. The utilization of the cluster eigenbases allows for the maximum percentage of total variation to be explained by a few eigencomponents (Chiou and Ling Li, 2013). A major strength of this clustering analysis procedure is that the incorporation of "the mean and covariance functions of each cluster" allows "additional insight into cluster structures which facilitates functional cluster analysis (Chiou and Ling Li, 2013)."

One main weakness of the itersubspace approach is that it can be computationally intensive. The majority of the computation time is present because of "cluster covariance estimation via two-dimensional smoothing (Chiou and Ling Li, 2013)." Another weakness of the itersubspace method is its sensitivity to the initial means. According to Chiou and Ling Li, this area of sensitivity still requires further investigation. Lastly, the itersubspace method can also run the risk of inconsistent estimation of the cluster mean and covariance function when the "number of curves in clusters is relatively small compared with random variation or when the measurement errors in observed curves are dominant" (Chiou and Ling Li, 2013). With these weaknesses, the itersubspace algorithm may need to be used with caution to an application with a small sample size.

IV. Distclust (Peng & Muller, 2008)

Distclust is another clustering method that is proposed to help with the analysis of functional data that contains unequally spaced measurements and sparseness "with additional measurement errors" from a stochastic process (Peng & Mueller, 2008). The motivation for this method was to help analyze data common in longitudinal studies and online bidding sales data (Peng & Mueller, 2008). Under "a square integrable stochastic process $\{X(t) : t \in T\}$" across time points, observations from the realization of this stochastic process can be expressed as:

$$Y_{il} = X_i(T_{il}) + \varepsilon_{il} \, ,$$

where $\{\varepsilon_{il}\}$ are independently and identically distributed with mean of zero and variance σ^2 and $\{Y_{il} : 1 \leq l \leq n_i \, ; 1 \leq i \leq n\}$ (Peng & Mueller, 2008).

It follows that by Mercer's theorem, $X_i(t)$ can be expanded with eigenfunctions of a positive semidefinite kernel because X is a "square integrable stochastic process"

$$Xi(t) = \mu(t) + \sum_{k=1}^{\infty} \xi_{ik}\varphi_k(t)$$

where $\mu(t)$ is the mean function "and the random variables $\{\xi_{ik} : k \geq 1\}$ for each i are uncorrelated with zero mean and variance," and $\lambda_k; \lambda 1 \geq \lambda 2 \geq \cdot \cdot \cdot \geq 0$ are the eigenvalues of the positive semidefinite kernel and φ_k are the corresponding orthonormal eigenfunctions (Peng & Mueller, 2008).

The distance that is proposed for their use in their clustering analysis is to "use the conditional expectation of the L2 distance between these two curves, given the data" (Peng & Mueller, 2008). Given curves Y_i and Y_j based on the observed data $X_i = (X_{i1},\ldots,X_{in})$ and $X_j = (X_{j1},\ldots,X_{jn})$, the L2 distance is defined as:

$$D(i,j) = \left\{\int_{\tau} \left(Y_i(t) - Y_j(t) \right)^2 \, dt \right\}^{1/2} \text{(Peng \& Mueller, 2008).}$$

The original definition of L2 distance is not applicable to observed data that is sparse so Peng and Mueller propose "to use the conditional expectation of $D^2(i,j)$ given" Y_i and Y_j as the squared distance between X_i and X_j. Specifically, this can be expressed by:

$$\overline{D}(i,j) = \{ E(D^2(i,j) | X_i , X_j) \}^{1/2} \quad 1 \leq i, j \leq n \text{ (Peng \& Mueller, 2008).}$$

Utilizing Parzeval's identity, the L2 distance can then be expressed as:

$$\overline{D}(i,j) = E\left(\sum_{k=1}^{\infty}(\varepsilon_{ik} - \varepsilon_{jk})^2 \mid Xi , Xj \right) \text{ (Peng \& Mueller, 2008).}$$

One limitation of the distclust procedure is that it has the most computational time in comparison to the other clustering methods evaluated in this thesis. Another limitation of this clustering algorithm is that it is not as powerful to curve data that is ideally equally spaced and not irregular.

V. Funclust (Jacques & Preda, 2013)

Inspired by the itersubspace method and functional random variable estimation, funclust proposes a parametric mixture model to aid in the clustering task. Although "the notion of probability density for functional random variables" is not generally specified, Jacques and Preda utilize the Karhunen-Loeve expansion (principal component analysis) of a stochastic process to get an “approximation for the density of functional variables” (2013). With that expansion, the best approximation of the underlying functional data density under the mean squares criterion is expressed as:

$$X^{(q)}(t) = \mu(t) + \sum_{j=1}^{q} C_j \psi_j(t)$$

where “μ is the mean function of X, C_j are zero-mean random variables (principal components), and ψj 's form an orthonormal system of eigenfunctions of the covariance operator of X (Jacques and Preda, 2013).”

By clever manipulation of probability and under the conditions that when “X is a Gaussian process, the principal components C_j are Gaussian and independent” the density of $f_X^{(q)}$ can be expressed as:

$$f_X^{(q)}(x) = \prod_{j=1}^{q} f_{Cj}\left(c_j(x)\right)$$

where f_{Cj} is the Gaussian centered density of variance λ” (Jacques and Preda, 2013).

With these two equations, Jacques and Preda describes how these “results justify at least theoretically, the use of the principal component densities f_{Cj} to approximate the notion of probability density” of curve X.

Based on this density approximation, a parametric mixture model is proposed. With this parametric mixture model, parameter estimation is then performed by incorporating an expectation maximizing like algorithm. Specifically, the algorithm maximizes the pseudo-completed log-likelihood to obtain an estimate:

$$L_C^{(q)}(\Theta;\, X,Z) = \sum_{i=1}^{n} \sum_{g=1}^{K} Z_{i,g}\,(\log \pi_g + \sum_{j=1}^{q_g} \log f_{Cj,g}\left(C_{i,j,g}\right))$$

where Z_i are group labels for each curve X_i, $\pi_{1,\ldots,}\pi_g$ are mixing probabilities, and f_{C_j} is the probability density function of the principal components (Jacques & Preda, 2013).

With the maximization, the maximum a posteriori rule provides the clusters for analysis.

A strength of the funclust procedure is that it takes group-specific subspaces into consideration through the assumption of the Gaussian mixture model for the coefficients of the eigen-function expansion (Jacques & Preda, 2013). One limitation of this procedure is the choice of the group specific dimension is still not fully clear and is an active area of research (Jacques & Preda, 2013).

VI. Waveclust (Giacofci et al. , 2013)

Addressing the limitations of splines to deal with high dimensional data and irregular curve shapes, waveclust strategizes to manage individual inter-variability in high dimensional curve clustering. A strength of using wavelet decomposition is that it accounts for the signal of both fixed and random effects. Giacofci et al. state specifically that "using a wavelet representation of this model allows us to characterize different types of smoothness conditions assumed on the response curves by the mean of their wavelet coefficients (2013)." Specifically, this method allows for the correlation within curves to have variability across groups and positions. It is noted though that in waveclust, orthonormal wavelet basis will be utilized as defined as:

$$\{ \varphi_{j0k}(t), k = 0,1,\ldots 2^{j0} - 1; \psi_{jk}(t), j \geq j_0, k = 0, \ldots, 2^j - 1\}$$

where each basis was calculated from a "father wavelet φ and a mother wavelet ψ of regularity $(r>0)$" and j is a time index (Giacofici et al., 2013).

With the orthonormal wavelet basis, each response curve, $Y_i(t)$, can be expressed as:

$$Y_i(t) = \sum_{k=0}^{2^{j_0}-1} c^*_{i,j0k}\varphi_{j0k}(t) + \sum_{j \geq j_0}\sum_{k=0}^{2^j-1} d^*_{i,jk}\psi_{jk}(t)$$

where c_i and d_i are wavelet coefficients of an individual curve (Giacofici et al., 2013).

After defining the response curve, waveclust "proposes an efficient dimension reduction step based on wavelet thresholding adapted to multiple curves" (Giacofci et al., 2013). Then, by accounting for the "appropriate structure for the random effect variance", it can mathematically be shown that "both fixed and random effects lie in the same functional space even when dealing with irregular functions that belong to Besov spaces" (2013). Giacofci et al. refer another source about Bezov's space but they utilize that Bezov's parameters of the function's number of derivatives to help define a function's regularity (2013).

Once in the wavelet domain, waveclust becomes a linear mixed-effects model that can be used for clustering. Utilizing an expectation maximization algorithm for maximum likelihood estimation, the expectation maximization algorithm maximizes this likelihood

$$\log \Upsilon\,(\,c, d, \upsilon, \theta, \varsigma; \pi, \alpha, \beta, G, \sigma^2_\varepsilon) = \log \Upsilon\,(c, d|\,\upsilon, \theta, \varsigma; \pi, \alpha, \beta, G, \sigma^2_\varepsilon)$$

$$+ \log \Upsilon\,(\upsilon, \theta|\,\varsigma; G)$$

$$+ \log \Upsilon\,(\,\varsigma\,;\,\pi\,)$$

where Υ and random effects (υ, θ) are label variables that are unobserved, α and β are the estimators of the mean curve coefficients, G is the variance of the random wavelet coefficients, and σ^2_ε is the noise measurement error (Giacofci et al., 2013).

It follows that the posterior probabilities for cluster l are then specified as:

$$\tau_{il}^{[h+1]} = \frac{\pi_l^{[h]} f(c_i, d_i;\ \alpha_l^{[h]}, \beta_l^{[h]}, G^{[h]} +\ \sigma_\varepsilon^{2[h]} I)}{\sum_p \pi_p^{[h]} f(\, c_i, d_i;\ \alpha_p^{[h]}, \beta_p^{[h]}, G^{[h]} +\ \sigma_\varepsilon^{2[h]} I)}$$

where f is the probability density function of the Gaussian distribution.

A major strength of the waveclust procedure is that its account for functional random effects in the model allows for "a better identification of the informative structures" for clustering (Giacofci et al., 2013). In contrast to other methods that rely on splines, Giacofci. et al. show that the waveclust method has a better estimation of the residuals. Another strength of the waveclust algorithm is that it has better computation time than spline-based functional clustering algorithms (Giacofci et al., 2013).

VII. FSCM (Jiang & Serban, 2012)

The functional spatial clustering method (fscm) uniquely specializes with handling spatially independent functions that are time varying.

Mathematically, this can be expressed as

$$Y_{ij} = f_{sj}(t_{ij}) + \sigma_\varepsilon \varepsilon_{ij}\ ,\ j = 1, \ldots, n$$

where the cluster information will enter in the model with different f_{sj} (t) for each cluster (Huijing & Serban, 2012).

Mathematically, fscm "models the spatial dependence in the joint distribution of (Y_j , Z_j), j = 1, . . . , n by assuming spatial dependence on the cluster membership Z_j , j = 1, . . . , n and on the conditional distribution of $Y_j|Z_j$, j = 1, . . . , n" (Huijing & Serban, 2012). A major advantage of specifying "spatial dependence in the joint distribution (Y_j , Z_j)" is that it "allows borrowing information across curves corresponding to nearby locations yet maintaining local resolution which will lead to enhanced estimation accuracy of the cluster patterns and of the cluster membership" (Huijing & Serban, 2012).

Interestingly, a nonparametric model with spatially correlated errors is used as the underlying clustering model. Fscm also uniquely "assumes that the clustering membership is a realization from a Markov random field" (Huijing & Serban, 2012). Under the assumption of Markov, the probability a curve belongs to its nearest neighbors depends on the state of the cluster membership and fscm uses k-nearest neighborhoods to describe the structure of the neighborhood.

By borrowing information across functions and stating these assumptions, "enhanced estimation accuracy of the cluster effects and of the cluster membership" is a result.

Because of the difficulty in maximizing a likelihood under spatial dependence, Serban and Huijing propose using "a pseudo-likelihood imputation for Z1, . . . , Zn and Monte-Carlo approximations in the imputation of the latent variables" (2012).

Therefore they can define the joint distribution of Z1, …, Zn for curves $Y_1, \ldots, Y_n$ such that:

$$\widehat{z_{jk}} \approx \frac{1}{M} \sum_{m=1}^{M} E\left[Z_{jk} \middle| Y_1, \ldots, Y_n, Y^m\right]$$

$$\approx \frac{1}{M} \sum_{m=1}^{M} E\left[Z_{jk} \middle| Y_{1,} \ldots, Y_n, \hat{Y}\right] = E\left[z_{jk} \middle| Y_1, \ldots, Y_n, \hat{Y}\right]$$

A major strength of fscm's method is its account for spatial dependence. One weakness that is important to note is that the conditional independence assumption for $Y_j|Z_j$ for computational feasibility has been a disagreement among other work (Huijing & Serban , 2012).

Methods of Analysis

For this analysis, the R package of "funcy" was utilized to perform functional data cluster analysis. The funcit function in "funcy" was used to perform fitfclust (James & Sugar, 2003), distclust (Peng & Muller, 2008), iterSubspace (Chiou & Ling Li, 2007), funclust (Jacques & Preda, 2013), waveclust (Giacofci et al. , 2013), and fscm (Jiang & Serban, 2012). K-Means clustering was performed using the kmeans.fd R function. To use these procedures, the data also needed to be converted into a functional data object in R. This was accomplished by formatFuncy function in "funcy" under the Format1 argument. The statistical measures of agreement were also calculated using the confusionMatrix function in the "caret" and "e1071" packages in R. Weighted kappa was additionally calculated using the wkappa function in the "psych" package in R. The seeds 12345 and 123456 were also used for the reproducibility of the analysis.

It is also important to note that each clustering procedure by default outputs clusters labeled as 1 and 2 when specified to cluster the data into two groups. To assess agreement between the proposed clusters and expert scores, the proposed clusters under each clustering procedure were re-labeled to correspond to the expert score. This relabeling technique was clinically driven as individuals with obstructed kidneys had higher concentrations of MAG3 across most time points including the final time points. Hence, this can mathematically be evaluated by comparing the sum of the cluster center across the final time points. Mathematically, we relabeled cluster 1 from the clustering algorithm as obstructed if the sum of the last five centers of cluster 1 were greater than the last five centers of cluster 2. It follows that cluster 2 in that case would be labeled as the not-obstructed group. Moreover, cluster 2 from the clustering algorithm was relabeled as obstructed if the sum of the last five centers of cluster 2 were greater than the last five centers of cluster 1. Then, cluster 1 would be relabeled as the not obstructed group.

Training Set Construction

Initial analysis was done only on the training set in order to gain a better understanding of the performance of the unsupervised clustering algorithms. The training set is a subset of the 216 renogram curves and only contains renogram curve data for individuals that had all three experts give the same score for obstructed kidney (2) or not obstructed kidney (0). For example, an individual where all experts gave a score of 2 were included in the training set. Individuals with all equivocal scores were not included in this dataset to ensure proper evaluation of each clustering method on well-defined renogram curves. Although there are a total of 175 curves that fit this criterion, the training set contains data for 147 renogram curves. The 147 renogram curves were randomly chosen from the 175 total renogram curves with the sample function in R to randomly choose 147 patient id's. The remaining 28 curves were used as the validation set for the second part of the analysis. The 147 curve observations were measured across 99 time points where 35 curves were scored as obstructed and 112 curves were scored as not obstructed.

Training Set Procedure

K-means and the functional data clustering algorithms were evaluated with the training set. Two-by-two count tables were constructed after each clustering procedure to cross-tabulate the number of renogram curves with respect to the proposed cluster classification under each clustering procedure and the expert consensus score (see Appendix, Figure A). Agreement was assessed through the calculation of statistical measures of Kappa, Sensitivity, Specificity, Positive Predictive Value, and Negative Predictive Value (Agresti, 2002).

Figure 1 - Diagram of Validation Set Construction

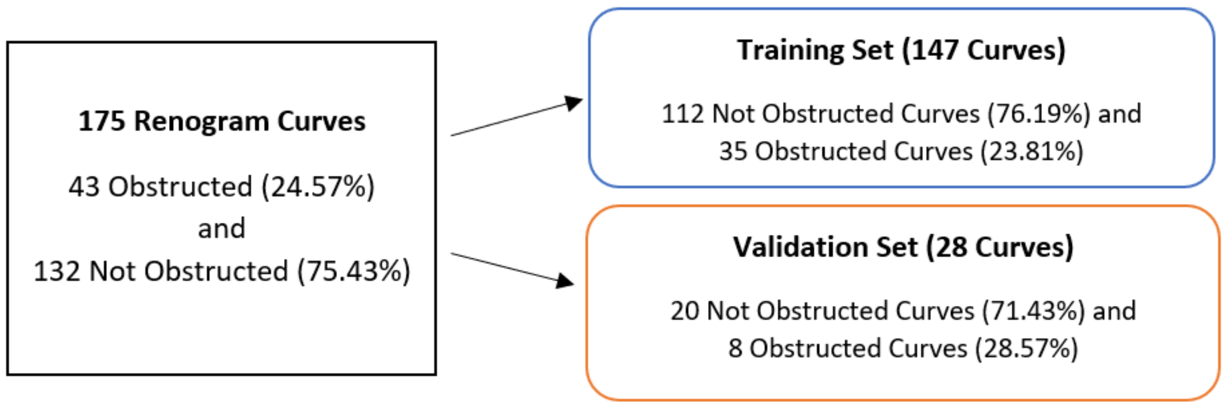

175 Renogram Curves
43 Obstructed (24.57%)
and
132 Not Obstructed (75.43%)
Training Set (147 Curves)
112 Not Obstructed Curves (76.19%) and
35 Obstructed Curves (23.81%)
Validation Set (28 Curves)
20 Not Obstructed Curves (71.43%) and
8 Obstructed Curves (28.57%)

Figure 1 illustrates how the training set and validation set were constructed from the 175 renogram curves that had a consensus across the three experts.

Validation Set Construction

The validation set is a subset of the 175 renogram curves where there was a consensus of expert score ratings across the three experts. Specifically, the validation set contains 28 curves were 20 of the curves were rated by all three experts as not obstructed and eight of the curves were rated by all three experts as obstructed. The validation set consist of the remaining curves in the 175 renogram curves dataset that were not selected by the random sampling that was utilized in the training set.

Validation Set Procedure

The validation set was the starting point for the analysis to validate the clustering algorithms. Each renogram curve data (one row of the validation set), was added onto the training set data as shown in figure B in the appendix. Hence, after one renogram curve from the validation set was added onto the 147 observations in the training set, a new data set of 148 observations was used in the analysis in order to obtain the predicted obstruction status for the newly added kidney. This "include-one-prediction" procedure was repeated to create 28 different datasets so that each kidney’s renogram curve in the validation set can be tested one at a time.

Each dataset underwent the seven functional data clustering method that were evaluated. For this analysis, the k-Means and the six functional clustering algorithms procedure were run. After each clustering algorithm was run, every renogram curve in the validation set was given a predicted obstruction status by each clustering algorithm. Then, cross-tabulation tables of the number of renogram curve were made with respect to each clustering method’s proposed classification of the curves and the expert’s consensus score of the curve. Agreement was assessed through the calculation of statistical measures of Kappa, Sensitivity, Specificity, Positive Predictive Value, and Negative Predictive Value (Agresti, 2002).

Hard-to-Interpret Set Construction

Figure 2 – Diagram of Hard-To-Interpret Set Construction

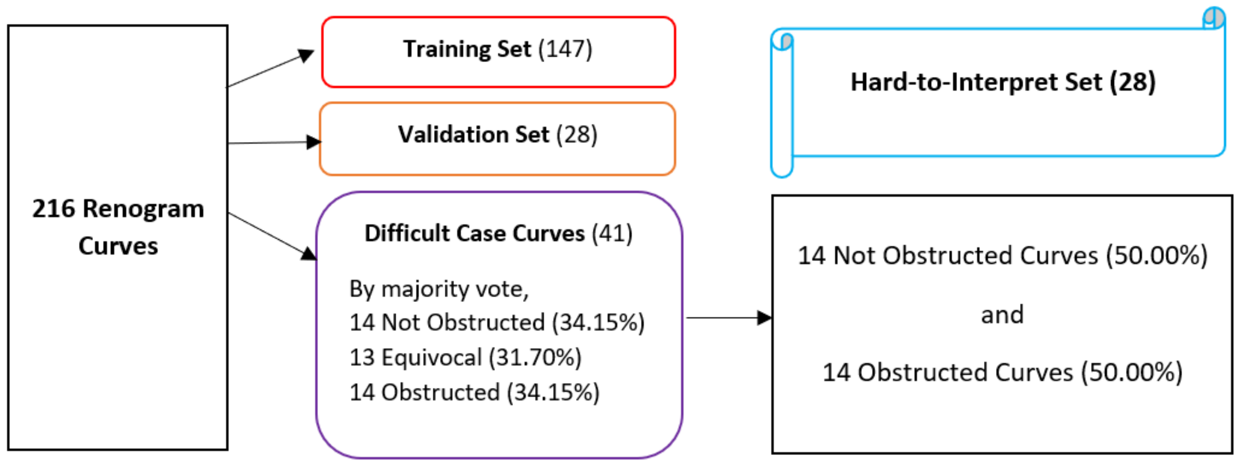

216 Renogram Curves
Training Set (147)
Validation Set (28)
Difficult Case Curves (41)
By majority vote,
14 Not Obstructed (34.15%)
13 Equivocal (31.70%)
14 Obstructed (34.15%)
Hard-to-Interpret Set (28)
14 Not Obstructed Curves (50.00%)
and
14 Obstructed Curves (50.00%)

Figure two illustrates how the hard-to-interpret set was constructed from difficult renogram curves.

The hard-to-interpret set contains renogram curve data that are considered as “difficult cases.” A renogram curve is considered a “difficult case” when the three experts did not all give the same rating. There were a total of 41 renogram curves that were “difficult cases” where 14 of the curves had two out of three experts give a not obstructed rating, 13 of the curves had two out of three experts give an equivocal rating, and 14 of the curves had two out of three experts give an obstructed rating. This analysis only focused on the curves where a majority of experts gave a not obstructed and obstructed score so only 28 curves were contained in the hard-to-interpret set.

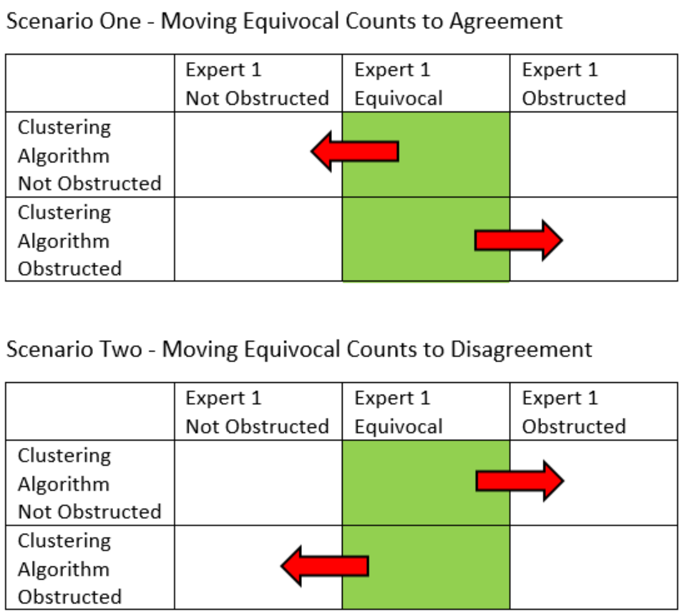
Hard-To-Interpret Set Procedure

The hard-to-interpret set was the starting point to see how the strongest performing clustering algorithms handle difficult renogram curves. Similar to the validation set procedure as displayed in figure B of the appendix, each renogram curve data (row of the hard-to-interpret set), was added onto the training set data. Hence, after one renogram curve from the second test set was added onto the consensus training set, a new data set of 148 observations was used in the analysis. This procedure was repeated to create 28 different datasets so that one hard-to-interpret set curve can be tested at a time

Each dataset underwent the specific functional data clustering method that was evaluated. For the this part of the analysis, the strongest performing clustering algorithms from the validation set analysis of fscm (Jiang & Serban, 2012) and waveclust (Giacofci et al. , 2013) procedure were evaluated. After each clustering algorithm run, cross-tabulations count tables of the renogram curves were made with respect to each clustering method’s proposed classification of the curves and each expert’s rating. Agreement was assessed through the calculation of weighted kappa (Agresti, 2002).

Handling Equivocal Cases in Hard-To-Interpret Set

Figure 3 – Illustration of the Two Scenarios of Handling Equivocal Cases



Scenario One - Moving Equivocal Counts to Agreement

	Expert 1 Not Obstructed	Expert 1 Equivocal	Expert 1 Obstructed
Clustering Algorithm Not Obstructed		← 🟩	
Clustering Algorithm Obstructed		🟩 →	

Scenario Two - Moving Equivocal Counts to Disagreement

	Expert 1 Not Obstructed	Expert 1 Equivocal	Expert 1 Obstructed
Clustering Algorithm Not Obstructed		🟩 →	
Clustering Algorithm Obstructed		← 🟩	

Figure three provides a visual of how equivocal counts were examined under two frameworks.

We considered two scenarios to deal with equivocal expert ratings when comparing the proposed classification by the clustering algorithm with a single expert. This modification was needed to calculate weighted kappa in order to keep the scoring the same with the two levels of (0-Not Obstructed, 2- Obstructed) on both sides of the 2 x 2 table. As shown in figure 3 above, one scenario that took place is that equivocal case counts of renogram curves were moved and combined with the counts that agree with the cluster algorithm classification. In other words, the equivocal expert opinion will always be treated as concordant to whichever of the two group assignments (obstructed or non-obstructed) each clustering algorithm provides. This takes into account that the equivocal rating curves can actually agree with the true biological outcome of the kidney being truly obstructed or not obstructed. As shown in figure 3, the second framework that was used was to move and combine equivocal rating counts that disagree with the cluster algorithm classification. In other words, the equivocal expert opinion will always be treated as discordant to whichever of the two group assignments (obstructed or non-obstructed) each clustering algorithm provides This takes into account that the equivocal rating curve can actually disagree with the true biological outcome of the kidney being truly obstructed or not obstructed. By evaluating both cases, this method takes into consideration that the equivocal rating can truly mean biologically that a kidney is obstructed or not obstructed kidney. The modifications were only applied to expert one and expert three because expert two gave no equivocal ratings on any curves in testing set two. All combinations of the three expert scores were also compared and evaluated. For each 2x2 table, weighted kappa was calculated (Agresti, 2002).

Results

I. Training Set Results

Figure 4- Illustration of K-Means Clustering Procedure

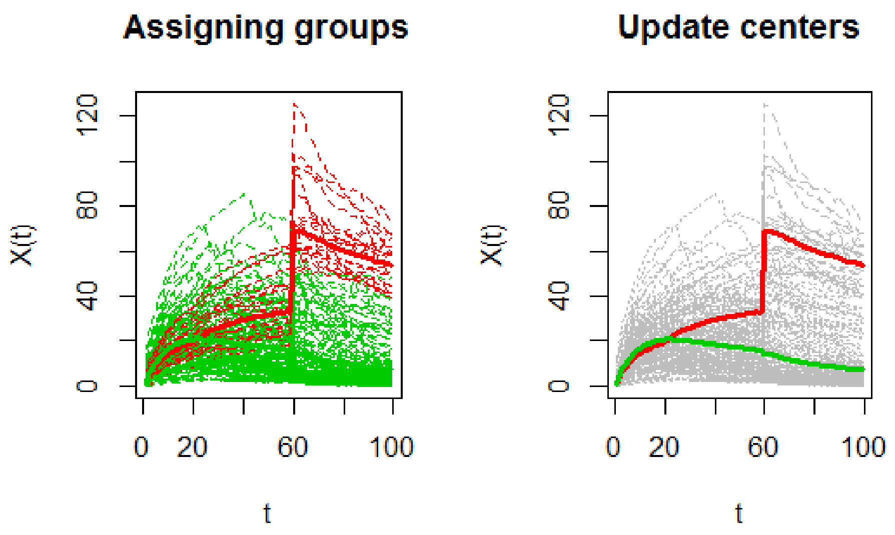


Figure four visually shows how the k-means procedure assigned each curve into 2 clusters (green- not obstructed, red- obstructed). The second panel illustrates the center curve for each of the two clusters.

Figure 5 - Illustration of Functional Clustering Procedure

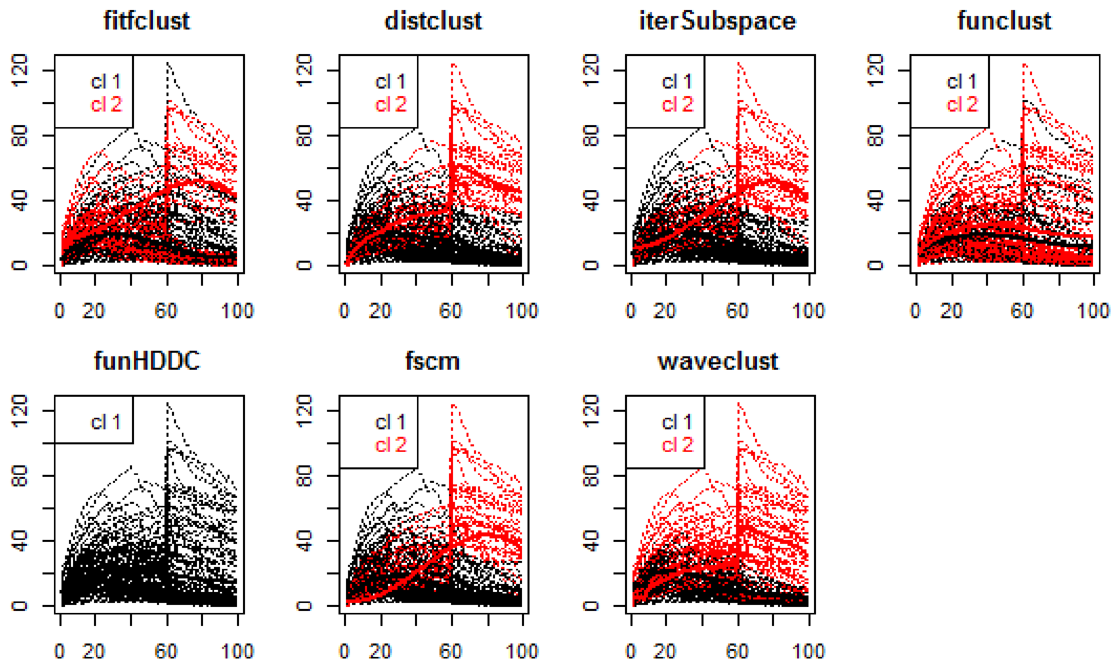


Figure five illustrates how each of the clustering algorithms places each curve into two clusters (black{cl 1}- not obstructed, red{cl 2}-obstructed).

Figure 6- Illustration of Functional Clustering Algorithm Cluster Centers

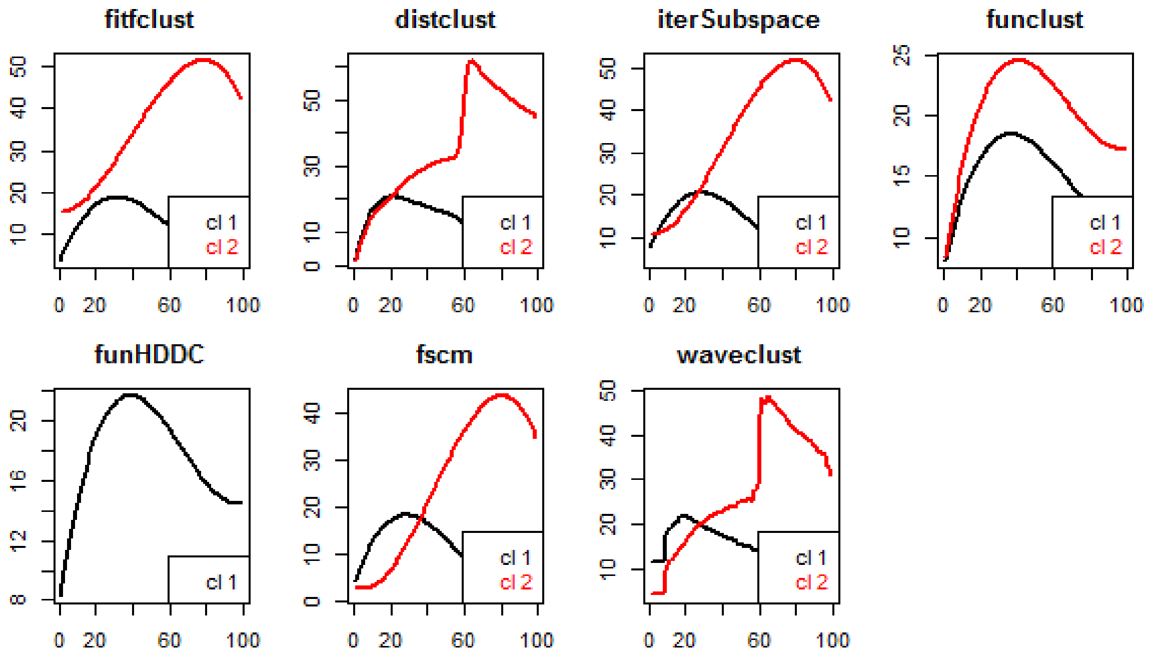


Figure six illustrates the cluster centers proposed by each functional data clustering algorithm. (Black – Not Obstructed, Red – Obstructed)

Table 1 - Agreement Measures for Training Set Analysis

N = 147

	KAPPA	SENSITIVITY	SPECIFICITY	POSITIVE PREDICTIVE VALUE	NEGATIVE PREDICTIVE VALUE
K-MEANS	0.5333	42.86%	100.00%	100.00%	84.85%
FITFCLUST	0.096	51.43%	60.71%	29.03%	80.00%
DISTCLUST	0.6877	71.43%	94.64%	80.65%	91.38%
ITERSUBSPACE	0.7322	77.14%	94.64%	81.82%	92.98%
FUNCLUST	0.0444	71.43%	35.71%	25.77%	80.00%
FSCM	0.7917	82.86%	95.54%	85.29%	94.69%
WAVECLUST	0.522	91.43%	74.11%	52.46%	96.51%

Note: In table 1, sensitivity is defined as the probability that a clustering algorithm will indicate obstructed among those who truly have an obstructed kidney. Specificity is the probability that a clustering algorithm will indicate not obstructed out of those who truly do not have an obstructed kidney. Positive predictive value is defined here as the probability that subjects classified as obstructed by the clustering algorithm truly have an obstructed kidney. Negative predictive value here is the probability that subjects classified as not obstructed by the clustering algorithm truly do not have an obstructed kidney.

The results presented in table one illustrate that certain clustering algorithms perform well while others perform poorly in the classification of obstructed and not obstructed renogram curves. The agreement measures were calculated from the 2x2 tables one through seven found in the appendix. The functional data clustering algorithm of fscm (Jiang & Serban, 2012) performs the best in contrast to the other clustering methods with its kappa of 0.7917, sensitivity of 82.86%, specificity of 95.54%, positive predictive value of 85.29%, and negative predictive value of 94.69%. The second highest performing clustering algorithm is itersubspace (Chiou & Ling Li, 2007) with a kappa of 0.7322. Fitfclust (James & Sugar, 2003) and funclust (Jacques & Preda, 2013) performed the weakest in terms of agreement with a kappa of 0.096 and 0.0444 respectively. With a kappa value greater than 0.7, there is statistically significant evidence to suggest a strong agreement of the obstructed and not obstructed kidney rating of the renogram curves between the expert consensus score and the proposed clusters from fscm (Jiang & Serban, 2012) and itersubspace (Chiou & Ling Li, 2007) respectively.

II. Validation Set Results

Table 2 – Agreement Measures for the Validation Set

28 Samples of N = 148

	KAPPA	SENSITIVITY	SPECIFICITY	POSITIVE PREDICTIVE VALUE	NEGATIVE PREDICTIVE VALUE
K-MEANS	0.4615	37.50%	100.00%	100.00%	80.00%
FITFCLUST	-0.1818	12.50%	70.00%	14.29%	66.67%
DISTCLUST	0.6216	62.50%	95.00%	83.33%	86.36%
ITERSUBSPACE	0.6216	62.50%	95.00%	83.33%	86.36%
FUNCLUST	0.0292	100.00%	5.00%	29.63%	100.00%
FSCM	0.7273	75.00%	95.00%	85.71%	90.48%
WAVECLUST	0.6957	100.00%	80.00%	66.67%	100.00%

Note: In table 2, sensitivity is defined as the probability that a clustering algorithm will indicate obstructed among those who truly have an obstructed kidney. Specificity is the probability that a clustering algorithm will indicate not obstructed out of those who truly do not have an obstructed kidney. Positive predictive value is defined here as the probability that subjects classified as obstructed by the clustering algorithm truly have an obstructed kidney. Negative predictive value here is the probability that subjects classified as not obstructed by the clustering algorithm truly do not have an obstructed kidney.

Table 2 presents the validation results of each functional data clustering algorithm after the inclusion of a single new renogram curve for 28 different curves. The agreement measures in table 2 were calculated from the 2x2 tables found in tables eight to 14 in the appendix. Like in the training set analysis, fscm (Jiang & Serban, 2012) performs the best with respect to agreement with a kappa value of 0.7273, sensitivity of 75.00%, specificity of 95.00%, positive predictive value of 85.71%, and negative predictive value of 90.48%. The second strongest clustering method in the validation set analysis is waveclust with a kappa of 0.6957. With a kappa value greater than 0.7, there is statistically significant evidence to suggest a strong agreement of obstructed and not obstructed kidney rating between the expert consensus score and the proposed clusters from fscm (Jiang & Serban, 2012). Unlike in aim one, itersubspace (Chiou & Ling Li, 2007) is not the second strongest method in producing the best agreement between the expert rating score and the proposed clusters. Similar to the consensus training set analysis as well, the two weakest performing clustering algorithm include funclust (Jacques &

Preda, 2013), and fitfclust (James & Sugar, 2003) with a kappa of 0.0292 and -0.1818 respectively.

III. Hard-To-Interpret Set Results

Table 3 – Weighted Kappa Measures for Hard-To-Interpret Set

N = 28

TWO WAY AGREEMENT COMPARISON	WEIGHTED KAPPA (EQUIVOCAL MOVED TO AGREEMENT)	WEIGHTED KAPPA (EQUIVOCAL MOVED TO DISAGREEMENT)
CASE 1: FSCM AND EXPERT ONE	0.5602	-0.0616
CASE 2: FSCM AND EXPERT TWO	-0.2912	NA
CASE 3: FSCM AND EXPERT THREE	0.0718	-0.2315
CASE 4: EXPERT ONE AND EXPERT TWO	0.3368	-0.3434
CASE 5: EXPERT ONE AND EXPERT THREE	-0.05514	NA
CASE 6: EXPERT TWO AND EXPERT THREE	0.4948	0.0000
CASE 7: WAVECLUST AND EXPERT ONE	0.5692	-0.1487
CASE 8: WAVECLUST AND EXPERT TWO	0.0618	NA
CASE 9: WAVECLUST AND EXPERT THREE	0.0714	-0.4358

Table 4 – Difficult Cases Ratings Across Experts and Clustering Algorithms

Case Number	**Expert 1 (AT.L)**	**Expert 2 (ED.L)**	**Expert 3 (RH.L)**	**FSCM Clustering**	**Waveclust Clustering**
1	🟥	🟥	🟨	🟩	🟩
2	🟥	🟥	🟨	🟩	🟥
3	🟥	🟥	🟨	🟩	🟥
4	🟥	🟥	🟨	🟩	🟩
5	🟨	🟥	🟥	🟩	🟩
6	🟨	🟥	🟥	🟥	🟥
7	🟨	🟥	🟥	🟩	🟥
8	🟨	🟥	🟥	🟥	🟥
9	🟨	🟥	🟥	🟩	🟥
10	🟨	🟥	🟥	🟩	🟩
11	🟨	🟥	🟥	🟩	🟩
12	🟩	🟩	🟨	🟩	🟩
13	🟩	🟩	🟨	🟩	🟥
14	🟩	🟩	🟨	🟩	🟩
15	🟨	🟩	🟩	🟩	🟩
16	🟨	🟩	🟩	🟩	🟩
17	🟨	🟩	🟩	🟥	🟥

18	🟩	🟩	🟥	🟥	🟩
19	🟩	🟥	🟩	🟩	🟥
20	🟩	🟥	🟩	🟩	🟩
21	🟩	🟥	🟩	🟩	🟥
22	🟥	🟩	🟩	🟥	🟥
23	🟥	🟩	🟩	🟥	🟥
24	🟥	🟩	🟩	🟥	🟥
25	🟥	🟥	🟩	🟥	🟥
26	🟥	🟩	🟥	🟥	🟥
27	🟥	🟩	🟥	🟩	🟩
28	🟩	🟥	🟥	🟩	🟩

Key: **Red** - Obstructed Rating (2); **Yellow** - Equivocal Rating (1); **Green** - Not Obstructed Rating (0)

From table 3, both fscm (Jiang & Serban, 2012) and waveclust (Giacofci et al., 2013) have the greatest agreement (kappa of 0.5602 and 0.5692 respectively) when compared with expert one when equivocal cases were moved under the agreement scenario. Even when the equivocal cases were moved under the disagreement scenario, fscm (Jiang & Serban, 2012) and waveclust (Giacofci et al., 2013) still had the greatest agreement with expert one. This illustrates that the clustering algorithm proposed classifications were most similar to expert one. Interestingly, expert one gave the most equivocal results which demonstrates that the clustering algorithm does not align with the majority score (two out of three experts agree) for "difficult case" renogram curves. This can be seen in table four as fscm (Jiang & Serban, 2012) and waveclust (Giacofci et al., 2013) agreed with the majority expert score only 42.86% (12/28) and 50.00% (14/28) of the time respectively. It is also interesting to note that fscm (Jiang & Serban, 2012) tended to be more conservative and give a higher percentage of 67.86% (19/28) of not obstructed ratings when compared to waveclust (Giacofci et al., 2013) giving a not obstructed rating 46.43% (13/28) of the time. It follows that waveclust (Giacofci et al., 2013) was not as conservative by giving a greater percentage of 53.57% (15/28) of obstructed ratings when compared to fscm's 32.14% (9/28) provision of obstructed ratings.

Discussion

Fscm (Jiang & Serban, 2012) has been one of the strongest performing clustering algorithm across the analysis. Fscm's strong performance encourage further investigation of its assumption of spatial dependence between the joint distribution of the curves and cluster membership. This assumption allows the fscm method to borrow information from other curves in the classification of renogram curves. Waveclust's good performance of the "difficult case" curves with the hard-to-interpret set motivate the importance of further research of the potential of wavelet basis to account for functional random effects, rather than splines, to help with prediction and classification of renogram curves.

On the other hand, fitfclust (James & Sugar, 2003) has consistently been one of the weakest performing clustering algorithms. Fitfclust's advantage to deal with sparse, irregular time point curves is not effective here as our curve measurements were equally spaced and not sparse. Fitfclust's incorporation of spline methods may also not be superior in the classification of renogram curves.

A strong assumption of the independence between each patient's right and left kidney is one limitation of this analysis. In this analysis, a patient's right and left kidney were treated as independent, separate observations. However, these measurements may actually be dependent since these two biological measurements were taken on the same individual. Future work can see how the dependent nature of these measurements may affect the results of the analysis.

Another limitation of the functional data clustering algorithms lies with respect to the handling of abnormal, outlier renogram curves. As many clustering algorithms are sensitive to cluster centers, abnormal renogram curves may produce inconsistent results in classifying renogram curves as obstructed or not obstructed. This sensitivity to the cluster center motivated our methods to consider the sensitivity by adding “one difficult case” curve at a time in our validation set and hard-to-interpret set analysis. Future work can look at how to properly increase the robustness of these functional data clustering methods with respect to outliers.

A final limitation of this analysis is the small sample size. With more data, unsupervised methods can become more powerful as there are more data to train each method. With only 216 curves, one of our test sets only had 28 curves that were used to assess for the classification task.

References

Agresti, A. *Categorical Data Analysis*. 2. New York, NY: Wiley; 2002. P 275-277

Bao J, Manatunga A, Binongo JNG, Taylor A (2011) *Key Variables for interpreting MAG3 diuretic scans: development and validation of a predictive model*. AJR ;197:325-333. PMCID: PMC3179984

Chiou, J.-M. and Li, P.-L. (2007), *Functional clustering and identifying substructures of longitudinal data*. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 69: 679–699. doi:10.1111/j.1467-9868.2007.00605.x (IterSubspace)

Christina Yassouridis (2017). funcy: Functional Clustering Algorithms. R package version 0.8.6. https://CRAN.R-project.org/package=funcy

David Meyer, Evgenia Dimitriadou, Kurt Hornik, Andreas Weingessel and Friedrich Leisch (2017). e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien. R package version 1.6-8. https://CRAN.R-project.org/package=e1071

Huijing Jiang & Nicoleta Serban (2012) Clustering Random Curves Under Spatial Interdependence With Application to Service Accessibility, Technometrics, 54:2, 108-119, DOI: 10.1080/00401706.2012.657106 (FSCM)

James, Gareth M., and Catherine A. Sugar. “Clustering for Sparsely Sampled Functional Data.” *Journal of the American Statistical Association*, vol. 98, no. 462, 2003, pp. 397–408. *JSTOR*, JSTOR, www.jstor.org/stable/30045249. (Fitfclust)

Jie Peng and Hans-Georg Muller, Distance-based clustering of sparsely observed stochastic processes, with applications to online auctions, Ann. Appl. Stat 2008. Volume 2, Number 3, 1056-1077. (Distclust)

J. B. MacQueen (1967): "Some Methods for classification and Analysis of Multivariate Observations, *Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability"*, Berkeley, University of California Press, 1:281-297

Julien Jacques, Cristian Preda. Funclust: a curves clustering method using functional random variables density approximation. *Neurocomputing*, Elsevier, 2013, 112, pp.164-171(fun)

Madison Giacofci, Sophie Lambert-Lacroix, Guillemette Marot, Franck Picard. Wavelet-based clustering for mixed-effects functional models in high dimension. *Biometrics*, Wiley, 2013, 69 (1), pp.31-40. (Waveclust)

Max Kuhn. Contributions from Jed Wing, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, Brenton Kenkel, the R Core Team, Michael Benesty, Reynald Lescarbeau, Andrew Ziem, Luca Scrucca,Yuan Tang, Can Candan and Tyler Hunt. (2017). caret: Classification and Regression Training. R package version 6.0-78. https://CRAN.R-project.org/package=caret

Mujoomdar M, Russell E, Dionne F, et al. Ottawa (ON): *Canadian Agency for Drugs and Technologies in Heath*; 2012. https://www.ncbi.nih.gov/books/NBK174850/

Naik, Azad. “k-Means Clustering Algorithm - Data Clustering Algorithms.” Data Clustering Algorithms, 2010, sites.google.com/site/dataclusteringalgorithms/k-means-clusteringalgorithm

National Kidney Foundation. *Hydronephrosis*. 3 Feb. 2017, www.kidney.org/atoz/content/hydronephrosis.

O’Reilly P, Aurell M, Britton K, Kletter K, Rosenthal L, Testa T. *Consensus on diuresis renography for investigating the dilated upper urinary tract*. J Nucl Med. 1996; 37:1872-1876. [PubMed:8917195]

Ortega, Joaquín & Del Rocío, Ma & Rojas, Boone & J Somodevilla García, María. (2009). *Research issues on K-means Algorithm: An Experimental Trial Using Matlab*. CEUR Workshop Proceedings. 534.

Revelle, W. (2017) *psych: Procedures for Personality and Psychological Research*, Northwestern University, Evanston, Illinois, USA, https://CRAN.R-project.org/package=psych Version = 1.7.8.

Taylor, Andrew T. “Radionuclides in Nephrourology, Part 2: Pitfalls and Diagnostic

Applications.” *Journal of nuclear medicine : official publication, Society of Nuclear Medicine* 55.5 (2014): 786–798. *PMC*. Web. 30 Mar. 2018.

Taylor, Andrew T, and Ernest V Garcia. “Computer Assisted Diagnosis in Renal Nuclear Medicine: Rationale, Methodology and Interpretative Criteria for Diuretic Renography.” *Seminars in nuclear medicine* 44.2 (2014): 146–158. *PMC*. Web. 30 Mar. 2018.

Taylor, Andrew, Amita Manatunga, and Ernest V. Garcia. “Decision Support Systems in Diuresis Renography.” *Seminars in nuclear medicine* 38.1 (2008): 67–81. *PMC*. Web. 30 Mar. 2018.

Urology Care Foundation. *What Is Kidney (Renal) Failure?* American Urological Association, 2018, www.urologyhealth.org/urologic-conditions/kidney-(renal)-failure.

Appendix

Figure A – Diagram of Consensus Training Set Procedure

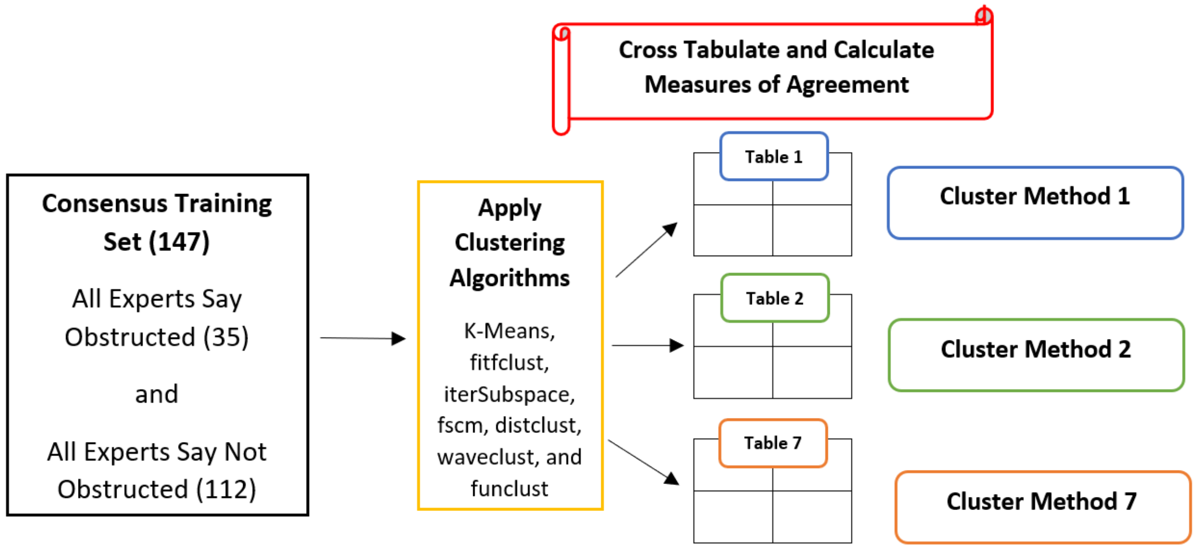

Cross Tabulate and Calculate Measures of Agreement
Consensus Training Set (147)
All Experts Say Obstructed (35)
and
All Experts Say Not Obstructed (112)
Apply Clustering Algorithms
K-Means, fitfclust, iterSubspace, fscm, distclust, waveclust, and funclust
Table 1
Table 2
Table 7
Cluster Method 1
Cluster Method 2
Cluster Method 7

Figure B - Diagram of Validation Set and Hard-To-Interpret Set Procedure

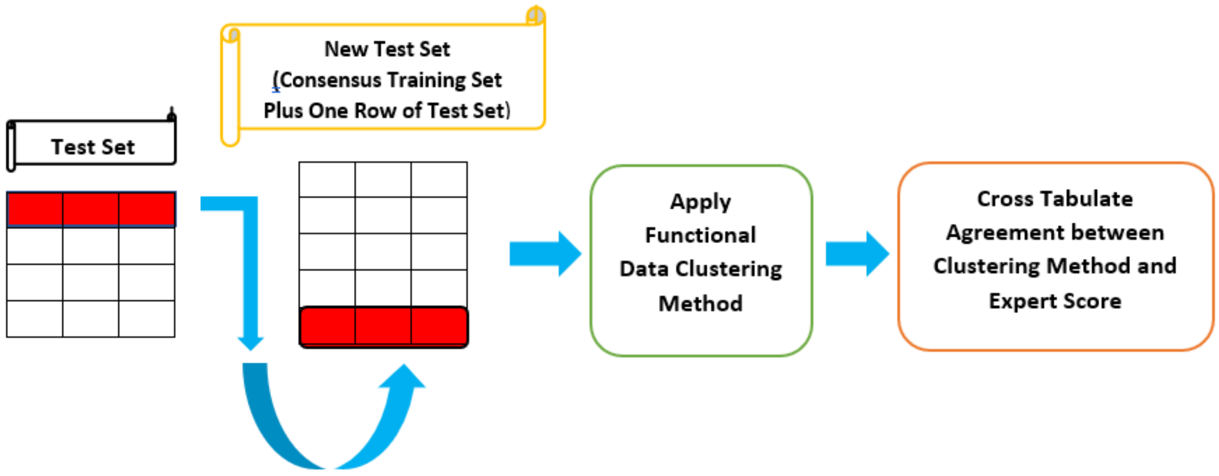

New Test Set (Consensus Training Set Plus One Row of Test Set)
Test Set
Apply Functional Data Clustering Method
Cross Tabulate Agreement between Clustering Method and Expert Score

Table 1 - Cross Tabulation of K-Means Classification And Expert Consensus Score, N = 147

	OBSTRUCTED (EXPERT – SCORE 2)	NOT OBSTRUCTED (EXPERT – SCORE 0)
K-MEANS OBSTRUCTED	15 (42.86%)	0 (0.00%)
K-MEANS NON-OBSTRUCTED	20 (57.14%)	112 (100.00%)

Table 2 - Cross Tabulation of Fitfclust Classification And Expert Consensus Score, N = 147

	OBSTRUCTED (EXPERT – SCORE 2)	**NOT OBSTRUCTED (EXPERT – SCORE 0)**
FITFCLUST OBSTRUCTED	18 (51.43%)	44 (39.29%)
FITFCLUST NON-OBSTRUCTED	17 (48.57%)	68 (60.71%)

Table 3 - Cross Tabulation of Distclust Classification And Expert Consensus Score, N = 147

	OBSTRUCTED (EXPERT – SCORE 2)	**NOT OBSTRUCTED (EXPERT – SCORE 0)**
DISTCLUST OBSTRUCTED	25 (71.43%)	6 (5.36%)
DISTCLUST NON-OBSTRUCTED	10 (28.57%)	106 (94.64%)

Table 4 - Cross Tabulation of IterSubspace Classification And Expert Consensus Score, N = 147

	OBSTRUCTED (EXPERT – SCORE 2)	**NOT OBSTRUCTED (EXPERT – SCORE 0)**
ITERSUBSPACE OBSTRUCTED	27 (77.14%)	6 (5.36%)
ITERSUBSPACE NON-OBSTRUCTED	8 (22.86%)	106 (94.64%)

Table 5 - Cross Tabulation of Funclust Classification And Expert Consensus Score, N = 147

	OBSTRUCTED (EXPERT – SCORE 2)	**NOT OBSTRUCTED (EXPERT – SCORE 0)**
FUNCLUST OBSTRUCTED	25 (71.43%)	72 (64.29%)
FUNCLUST NON-OBSTRUCTED	10 (28.57%)	40 (35.71%)

Table 6 - Cross Tabulation of Waveclust Classification And Expert Consensus Score, N = 147

	OBSTRUCTED (EXPERT – SCORE 2)	**NOT OBSTRUCTED (EXPERT – SCORE 0)**
WAVECLUST OBSTRUCTED	32 (91.43%)	29 (25.89%)
WAVECLUST NON-OBSTRUCTED	3 (8.57%)	83 (74.11%)

Table 7 - Cross Tabulation of Fscm Classification And Expert Consensus Score, N = 147

	OBSTRUCTED (EXPERT – SCORE 2)	NOT OBSTRUCTED (EXPERT – SCORE 0)
FSCM OBSTRUCTED	29 (82.86%)	5 (4.46%)
FSCM NON-OBSTRUCTED	6 (17.14%)	107 (95.54%)

Table 8 - Cross Tabulation of K-Means Classification And Expert Consensus Score, N = 28

	OBSTRUCTED (EXPERT – SCORE 2)	NOT OBSTRUCTED (EXPERT – SCORE 0)
K-MEANS OBSTRUCTED	3 (37.50%)	0 (0.00%)
K-MEANS NON-OBSTRUCTED	5 (62.50%)	20 (100.00%)

Table 9 - Cross Tabulation of IterSubspace Classification And Expert Consensus Score, N = 28

	OBSTRUCTED (EXPERT – SCORE 2)	NOT OBSTRUCTED (EXPERT – SCORE 0)
ITERSUBSPACE OBSTRUCTED	5 (62.50%)	1 (5.00%)
ITERSUBSPACE NON-OBSTRUCTED	3 (37.50%)	19 (95.00%)

Table 10 - Cross Tabulation of Fscm Classification And Expert Consensus Score, N = 28

	OBSTRUCTED (EXPERT – SCORE 2)	NOT OBSTRUCTED (EXPERT – SCORE 0)
FSCM OBSTRUCTED	6 (75.00%)	1 (5.00%)
FSCM NON-OBSTRUCTED	2 (25.00%)	19 (95.00%)

Table 11- Cross Tabulation of Distclust Classification And Expert Consensus Score, N = 28

	OBSTRUCTED (EXPERT – SCORE 2)	NOT OBSTRUCTED (EXPERT – SCORE 0)
DISTCLUST OBSTRUCTED	5 (62.50%)	1 (5.00%)
DISTCLUST NON-OBSTRUCTED	3 (37.50%)	19 (95.00%)

Table 12- Cross Tabulation of Waveclust Classification And Expert Consensus Score, N = 28

	OBSTRUCTED (EXPERT – SCORE 2)	NOT OBSTRUCTED (EXPERT – SCORE 0)
WAVECLUST OBSTRUCTED	8 (100.00%)	4 (20.00%)
WAVECLUST NON-OBSTRUCTED	0 (0.00%)	16 (80.00%)

Table 13 - Cross Tabulation of Fitfclust Classification And Expert Consensus Score , N = 28

	OBSTRUCTED (EXPERT – SCORE 2)	NOT OBSTRUCTED (EXPERT – SCORE 0)
FITFCLUST OBSTRUCTED	1 (12.50%)	6 (30.00%)
FITFCLUST NON-OBSTRUCTED	7 (87.50%)	14 (70.00%)

Table 14 - Cross Tabulation of Funclust Classification And Expert Consensus Score , N = 28

	OBSTRUCTED (EXPERT – SCORE 2)	NOT OBSTRUCTED (EXPERT – SCORE 0)
FUNCLUST OBSTRUCTED	8 (100.00%)	19 (95.00%)
FUNCLUST NON-OBSTRUCTED	0 (0.00%)	1 (5.00%)

Table 15 - Cross Tabulation of Fscm Classification and Expert One Score , N = 28

	Not Obstructed (Expert – Score 0)	Equivocal (Expert – Score 1)	Obstructed (Expert – Score 2)
Fscm Not Obstructed	7	7	5
Fscm Obstructed	1	3	5

Table 16 – Agreed Table of Fscm Classification and Expert One Score

	Not Obstructed (Expert – Score 0)	Obstructed (Expert – Score 2)
Fscm Not Obstructed	14	5
Fscm Obstructed	1	8

Table 17 – Disagreed Table of Fscm Classification and Expert One Score

	Not Obstructed (Expert – Score 0)	Obstructed (Expert – Score 2)
Fscm Not Obstructed	7	12
Fscm Obstructed	4	5

Table 18 - Cross Tabulation of Fscm Classification and Expert Two Score , N = 28

	Not Obstructed (Expert – Score 0)	Obstructed (Expert – Score 2)
Fscm Not Obstructed	6	13
Fscm Obstructed	6	3

Table 19 - Cross Tabulation of Fscm Classification and Expert Three Score , N = 28

	Not Obstructed (Expert – Score 0)	Equivocal (Expert – Score 1)	Obstructed (Expert – Score 2)
Fscm Not Obstructed	5	7	7
Fscm Obstructed	5	0	4

Table 20 – Agreed Table of Fscm Classification and Expert Three Score

	Not Obstructed (Expert – Score 0)	Obstructed (Expert – Score 2)
Fscm Not Obstructed	12	7
Fscm Obstructed	5	4

Table 21 – Disagreed Table of Fscm Classification and Expert Three Score

	Not Obstructed (Expert – Score 0)	Obstructed (Expert – Score 2)
Fscm Not Obstructed	5	14
Fscm Obstructed	5	4

Table 22 - Cross Tabulation of Expert One and Expert Two Score, N = 28

	Expert One Not Obstructed	Expert One Equivocal	Expert One Obstructed
Expert Two Not Obstructed	4	3	5
Expert Two Obstructed	4	7	5

Table 23 – Agreed Table of Expert One and Expert Two Score

	Expert One Not Obstructed	Expert One Obstructed
Expert Two Not Obstructed	7	5
Expert Two Obstructed	4	12

Table 264– Disagreed Table of Expert One and Expert Two Score

	Expert One Not Obstructed	Expert One Obstructed
Expert Two Not Obstructed	4	8
Expert Two Obstructed	11	5

Table 25 - Cross Tabulation of Expert Two and Expert Three Score, N = 28

	Expert One Not Obstructed	Expert One Equivocal	Expert One Obstructed
Expert Two Not Obstructed	6	3	3
Expert Two Obstructed	4	4	8

Table 26 – Agreed Table of Expert Two and Expert Three Score

	Expert One Not Obstructed	Expert One Obstructed
Expert Two Not Obstructed	9	3
Expert Two Obstructed	4	12

Table 27 – Disagreed Table of Expert Two and Expert Three Score

	Expert One Not Obstructed	Expert One Obstructed
Expert Two Not Obstructed	6	6
Expert Two Obstructed	8	8

Table 28- Cross Tabulation of Expert One and Expert Three Score, N = 28

	Expert Three Not Obstructed	Expert Three Equivocal	Expert Three Obstructed
Expert One Not Obstructed	3	3	2
Expert One Equivocal	3	0	7
Expert Two Obstructed	4	4	2

Table 29 - Cross Tabulation of Expert One and Waveclust, N = 28

	Obstructed (Expert 1– Score 2)	Equivocal (Expert 1– Score 1)	Not Obstructed (Expert 1 – Score 0)
Waveclust Not Obstructed	5	5	3
Waveclust Obstructed	3	5	7

Table 30 – Agreed Table of Expert Two and Expert Three Score

	Obstructed (Expert– Score 2)	Not Obstructed (Expert – Score 0)
Expert Two Obstructed	10	3
Expert Two Non-Obstructed	3	12

Table 31 – Disagreed Table of Expert Two and Expert Three Score

	Obstructed (Expert – Score 2)	Not Obstructed (Expert – Score 0)
Expert Two Obstructed	5	8
Expert two Non-Obstructed	8	7

Table 32 - Cross Tabulation of Waveclust Classification and Expert Two Score , N = 28

	Not Obstructed (Expert – Score 0)	Obstructed (Expert – Score 2)
Waveclust Not Obstructed	6	7
Waveclust Obstructed	6	9

Table 33 - Cross Tabulation of Expert Three and Waveclust, N = 28

	Obstructed (Expert 3– Score 2)	Equivocal (Expert 3– Score 1)	Not Obstructed (Expert 3 – Score 0)
Waveclust Not Obstructed	3	4	6
Waveclust Obstructed	7	3	5

Table 34 – Agreed Table of Waveclust and Expert Three Score

	Obstructed (Expert– Score 2)	Not Obstructed (Expert – Score 0)
Expert Two Obstructed	7	6
Expert Two Non-Obstructed	7	8

Table 35 – Disagreed Table of Waveclust and Expert Three Score

	Obstructed (Expert – Score 2)	Not Obstructed (Expert – Score 0)
Expert Two Obstructed	3	10
Expert two Non-Obstructed	10	5

Table 36 - Rand Index Across Clustering Methods for Curves

Methods	fitfclust	distclust	iterSubspace	funclust	funHDDC	fscm	waveclust
fitfclust	NA	0.6819	0.6819	0.0074	0.0000	0.6639	0.2203
distclust	0.6819	NA	0.9591	0.0013	0.0000	0.9836	0.1788
iterSubspace	0.6819	0.9591	NA	-0.0090	0.0000	0.9794	0.1788
funclust	0.0074	0.0013	-0.0090	NA	0.0000	-0.0063	-0.0015
funHDDC	0.0000	0.0000	0.0000	0.0000	NA	0.0000	0.0000
fscm	0.6639	0.9386	0.9794	-0.0063	0.0000	NA	0.1710
waveclust	0.2203	0.1788	0.1788	-0.0015	0.0000	0.1710	NA

Table 37 - Elapsed Computational Time Across Methods for Combined Baseline and Diuretic Curves

Clustering Method	Time (Minutes)
Fitfclust	2.78
Distclust	113.03
iterSubspace	0.24
funclust	24.37
funHDDC	0.05
fscm	5.45
waveclust	20.54

Table 38 – Distribution of Gender Across Groups

	Group 1	Group 2	Group 3	Group 4	All Groups
Male	31	23	11	23	54 (Left or Right), 108 (Left and Right)
Female	18	26	35	26	54 (Left or Right), 108 (Left and Right)

Table 39 – Distribution of Age Across Groups

	Group 1	Group 2	Group 3	Group 4	All Groups
Mean	58.35	58.67	58.67	56.15	57.16
Minimum	25	18	18	18	18
First Quartile	43	48	48	44	41.75
Median	61	61	61	57	59.50
Third Quartile	70	71	71	67.75	70.00
Max	87	83	83	83	87.00

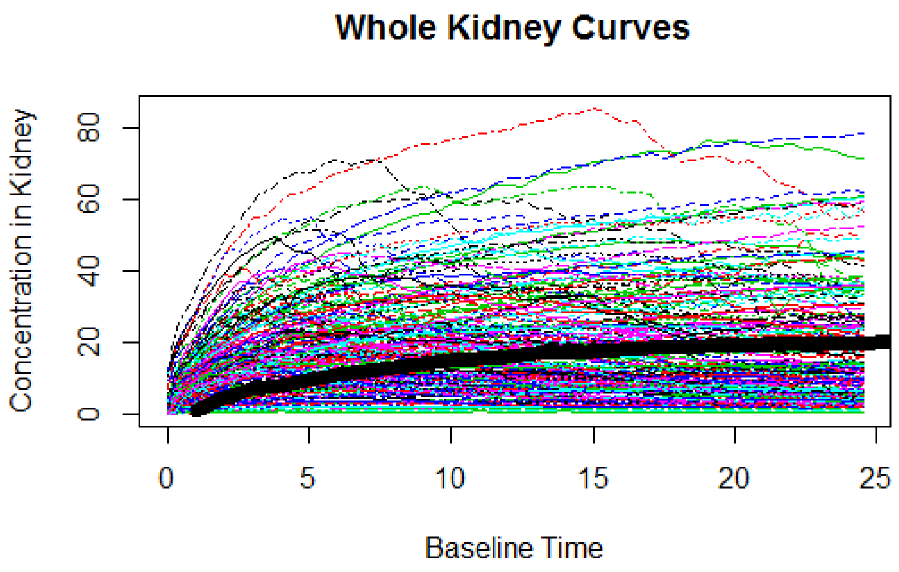
Table 40 – Number of Curves Across Groups

	Group 1	Group 2	Group 3	Group 4	All Groups
Count (Left)	91	82	12	40	108

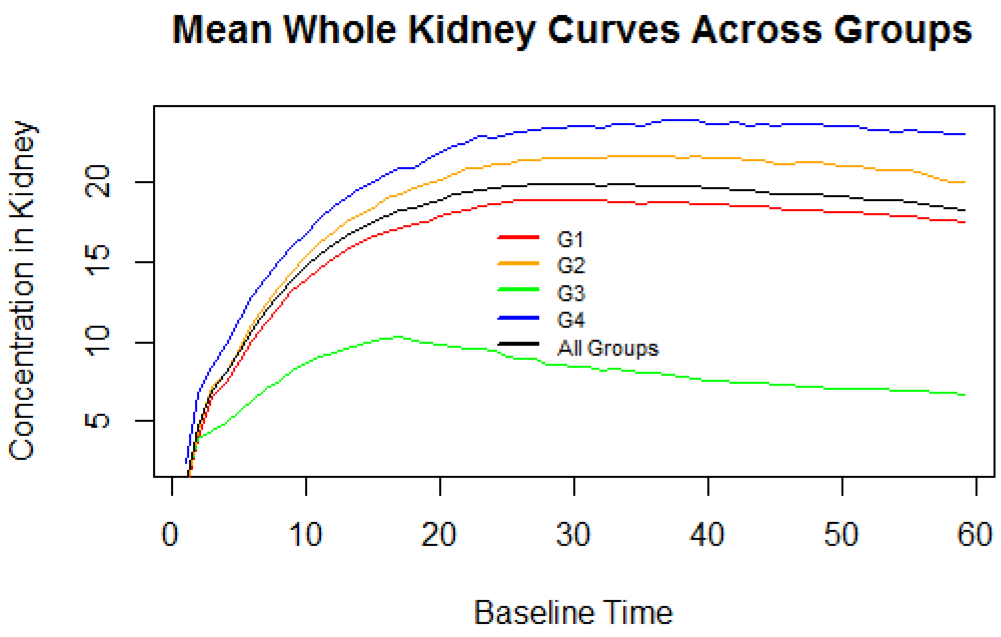
Table 41 – Number of Curves Across Diagnosis

	Obstructed	Equivocal	Non-Obstructed
Curves	58	10	145

Plot A- Plot of Whole Kidney Curves



Plot B – Mean Whole Kidney Curves



R Code

```
y1l <- group1234_Left_curve_expert_data  %>% select(V6:V64) # Extract Baseline curves for left
kidney

y1r <- group1234_Right_curve_expert_data  %>% select(V6:V64) # Extract Baseline curves for right
kidney

y1 <- t(rbind(y1l,y1r))/1000 # Stacking left and right kidney baseline curves with scale divided by 1000

y2l <- group1234_Left_curve_expert_data  %>% select(a1:a40) # Extract second curves for left kidney

y2r <- group1234_Right_curve_expert_data  %>% select(a1:a40) # Extract second curves for right
kidney

y2 <- t(rbind(y2l,y2r))/1000 # Stacking left and right kidney second curves with scale divided by 1000


Xl <-  group1234_Left_curve_expert_data  %>% select(AGE, Gender)

Xr <-  group1234_Right_curve_expert_data  %>% select(AGE, Gender)

X <- rbind(Xl,Xr)

X[,1] <- (X[,1] - mean(X[,1]))/sd(X[,1])

X <- matrix(as.numeric(t(X)),nrow=2)


Wl <- as.matrix(group1234_Left_curve_expert_data[,c(6,7,8)]) # Three sets of Expert ratings for left
kidney

Wr <- as.matrix(group1234_Right_curve_expert_data[,c(6,7,8)]) # Three sets of Expert ratings for right
kidney

W <- t(rbind(Wl, Wr)) # Stacking three sets of expert ratings for left and right kidneys


yf1 <- fdata(t(y1), argvals = Baseline_Time_Interval) # Registered baseline curves

yf2 <- fdata(t(y2), argvals = Diuretic_Time_Interval) # Registered second curves


ave_y1 <- c(rep(NULL,59))


for(i in 1:59) {

  ave_y1[i] <- mean( t(y1)[,i] )

}
```

```
plot(yf1, main = "Whole Kidney Curves", xlab = "Baseline Time", ylab = "Concentration in Kidney") #
Plot them

lines(ave_y1, type = "l", lwd = 8, col = "black") #Plot the average curve for all groups


ave_y2 <- c(rep(NULL,40))


for(i in 1:40) {
  ave_y2[i] <- mean( t(y2)[,i] )
}


plot(yf2, main = "Diuretic Whole Kidney Curves", xlab = "Diuretic Time" , ylab = "Concentration in
Kidney")

lines(ave_y2, type = "l", lwd = 8, col = "black") #Plot the average curve for all groups


#Majority Voting Coding


Exp_Majority <- t(W)
Exp_Majority <- as.data.frame(Exp_Majority)
nrow(Exp_Majority)
Exp_Majority[1,2]
Majority <- rep(0, 216)
Exp_Majority <- cbind(Exp_Majority, Majority)


for (i in 1:nrow(Exp_Majority)) {
  #If all experts agree, give majority the same vote
  if((Exp_Majority[i,1] == Exp_Majority[i,2]) && (Exp_Majority[i,1] == Exp_Majority[i,3]) &&
(Exp_Majority[i,2] == Exp_Majority[i,3])) {
    Exp_Majority[i,4] = Exp_Majority[i,2]
  }
  #If all experts disagree, code it as equivocal
  if((Exp_Majority[i,1] != Exp_Majority[i,2]) & (Exp_Majority[i,1] != Exp_Majority[i,3]) &
(Exp_Majority[i,2] != Exp_Majority[i,3])) {
    Exp_Majority[i,4] = 1
  }
  #If 2 experts agree, that will be the majority vote
  else {
    mode <- unique(Exp_Majority[i,])
    Exp_Majority[i,4] = mode[which.max(tabulate(match(Exp_Majority[i,], mode)))]
  }
}

count(Exp_Majority, vars = c("AT.L") )
count(Exp_Majority, vars = c("ED.L") )
count(Exp_Majority, vars = c("RH.L") )
count(Exp_Majority, vars = c("Majority") )

#To use functional data clustering methods
install.packages("funcy")
library(funcy)
#Set seed for reproducible research
set.seed(123456)
#Set data to right format for fdata object
baseline_format <- formatFuncy(data = y1, format = "Format1")

#New Analysis Plan
#2/26/18

#Obtain consenesus training set
#Where all experts say obstructed(2) or all experts say not obstructed(0)
two_curves_a <- as.data.frame(two_curves)
```

```
two_curves_new <- t(two_curves_a)

#Attach expert ratings onto the two curves so we can subset by decision status
two_curves_new2 <- cbind(two_curves_new, Exp_Majority[,-4])
two_curves_new2all <- cbind(two_curves_new, Exp_Majority)

#Only want Where all experts say obstructed(2) or all experts say not obstructed(0)
two_curves_2 <- subset(two_curves_new2, AT.L == 2 & ED.L == 2 & RH.L == 2)
two_curves_0 <- subset(two_curves_new2, AT.L == 0 & ED.L == 0 & RH.L == 0)

#All obstructed and not obstructed with expert ratings
two_curves02e <- rbind(two_curves_0,two_curves_2)

#All obstructed and not obstructed with no expert ratings
two_curves_02 <- two_curves02e[,-c(100:102)]

#Testing Set Construction
two_curves_no1 <- subset(two_curves_new2all, AT.L == 2 & ED.L == 2 & RH.L == 1) #4
two_curves_no2 <- subset(two_curves_new2all, AT.L == 2 & ED.L == 1 & RH.L == 2) #0
two_curves_no3 <- subset(two_curves_new2all, AT.L == 1 & ED.L == 2 & RH.L == 2) #7

two_curves_no4 <- subset(two_curves_new2all, AT.L == 1 & ED.L == 1 & RH.L == 2) #3
two_curves_no5 <- subset(two_curves_new2all, AT.L == 1 & ED.L == 1 & RH.L == 0) #0
two_curves_no6 <- subset(two_curves_new2all, AT.L == 1 & ED.L == 2 & RH.L == 1) #1
two_curves_no7 <- subset(two_curves_new2all, AT.L == 2 & ED.L == 1 & RH.L == 1) #1
two_curves_no8 <- subset(two_curves_new2all, AT.L == 1 & ED.L == 0 & RH.L == 1) #1
two_curves_no9 <- subset(two_curves_new2all, AT.L == 0 & ED.L == 1 & RH.L == 1) #0

two_curves_no10 <- subset(two_curves_new2all, AT.L == 0 & ED.L == 0 & RH.L == 1) #3
two_curves_no11 <- subset(two_curves_new2all, AT.L == 0 & ED.L == 1 & RH.L == 0) #0
```

```
two_curves_no12 <- subset(two_curves_new2all, AT.L == 1 & ED.L == 0 & RH.L == 0) #3

two_curves_no13 <- subset(two_curves_new2all, AT.L == 0 & ED.L == 0 & RH.L == 2) #1

two_curves_no14 <- subset(two_curves_new2all, AT.L == 0 & ED.L == 2 & RH.L == 0) #3

two_curves_no15 <- subset(two_curves_new2all, AT.L == 2 & ED.L == 0 & RH.L == 0) #3


two_curves_no16 <- subset(two_curves_new2all, AT.L == 2 & ED.L == 2 & RH.L == 0) #1

two_curves_no17 <- subset(two_curves_new2all, AT.L == 2 & ED.L == 0 & RH.L == 2) #2

two_curves_no18 <- subset(two_curves_new2all, AT.L == 0 & ED.L == 2 & RH.L == 2) #1

two_curves_no19 <- subset(two_curves_new2all, AT.L == 1 & ED.L == 1 & RH.L == 1) #4


two_curves_no20 <- subset(two_curves_new2all, AT.L == 0 & ED.L == 1 & RH.L == 2) #0

two_curves_no21 <- subset(two_curves_new2all, AT.L == 0 & ED.L == 2 & RH.L == 1) #0

two_curves_no22 <- subset(two_curves_new2all, AT.L == 1 & ED.L == 2 & RH.L == 0) #0

two_curves_no23 <- subset(two_curves_new2all, AT.L == 1 & ED.L == 0 & RH.L == 2) #2

two_curves_no24 <- subset(two_curves_new2all, AT.L == 2 & ED.L == 0 & RH.L == 1) #1

two_curves_no25 <- subset(two_curves_new2all, AT.L == 2 & ED.L == 1 & RH.L == 0) #0


#Test Set, 41 curves

two_curves_test <- rbind(two_curves_no1, two_curves_no3, two_curves_no4,
two_curves_no6,two_curves_no7, two_curves_no8, two_curves_no10,two_curves_no12,
two_curves_no13, two_curves_no14, two_curves_no15, two_curves_no16, two_curves_no17,
two_curves_no18, two_curves_no19, two_curves_no23, two_curves_no24)

count(two_curves_test,var = "Majority")


#We want test set with no equivoval majority, just obstructed and not obstructed

#Get 28 curves now with expert ratings

two_curves_test2 <- subset(two_curves_test, Majority == 0 | Majority == 2)


#Test set with no expert ratings

two_curves_test2_noexp <- two_curves_test2[, -c(100:103)]
```

```
#Apply K-Means clustering on this dataset


set.seed(12345)

k_twocurves <- kmeans.fd(two_curves_02,ncl = 2)

#ncl = number of clusters (groups) to classify

k_twocurves


k_twocurves_center <- k_twocurves$centers[["data"]]


par(mfrow=c(1,1))

plot(k_twocurves_center[1,], type = "l", col = "red", main = "K-Mean Curves for Combined Kidney
Curves", xlab = "Time Points", ylab = "Concentration in Kidney") #Curve 1

lines(k_twocurves_center[2,], col = "blue")

legend("topleft", legend = c("Obstructed", "Not Obstructed"), col = c("red","blue"), lwd = 1, bty = "n")



#Determine how many curves belong to each cluster

twocurves_cluster <- k_twocurves$cluster

table(twocurves_cluster)

length(twocurves_cluster)

#1- Obstructed

#2- Not Obstructed


#Expert and K-Means Cluster Group

twocurves_kmeans_score <- cbind(two_curves02e[,c(100:102)],twocurves_cluster)


count(twocurves_kmeans_score, vars = c("AT.L", "twocurves_cluster") )
```

```
#Functional Data Analysis Clustering
library(funcy)

#Get data into fdata object
twocurves_format <- formatFuncy(data = as.matrix(t(two_curves_02)), format = "Format1")

set.seed(123456)
twocurvesfit <- funcit(twocurves_format , k = 2, methods = c(1:7))

summary(twocurvesfit)

#Plot the clustered curves
plot(twocurvesfit, legendPlace = "topleft")
plot(twocurvesfit, type = "centers", legendPlace = "bottomright")
#Cluster 1- Not Obstructed
#Cluster 2 - Obstructed

#Collect all the cluster information
twocurves_clusters <- as.data.frame(Cluster(twocurvesfit))

#Attach expert ratings onto this cluster data frame
twocurves_clusters <- cbind(two_curves02e[,c(100:102)],twocurves_clusters)

#Get Cross Tabulation Counts for tables
count(twocurves_clusters, vars = c("AT.L", "fitfclust") )
count(twocurves_clusters, vars = c("AT.L", "distclust") )
count(twocurves_clusters, vars = c("AT.L", "iterSubspace") )
count(twocurves_clusters, vars = c("AT.L", "funclust") )
count(twocurves_clusters, vars = c("AT.L", "waveclust") )
count(twocurves_clusters, vars = c("AT.L", "fscm") )
```

```
#Calculate kappa, sensitivity, specificity, NPV, and PPV for each 2x2 clustering table
install.packages("caret")
install.packages("e1071")
library(caret)
library(e1071)

?confusionMatrix

#K-Means Table
kmean_pred <- matrix(data = c(22,0,21,132),nrow = 2, ncol = 2, byrow = TRUE)
kmean_table <- as.table(kmean_pred)
confusionMatrix(data = kmean_table, positive ="A")

#Fitfclust Table
fitf_pred <- matrix(data = c(21,51,22,81),nrow = 2, ncol = 2, byrow = TRUE)
fitf_table <- as.table(fitf_pred)
confusionMatrix(data = fitf_table, positive ="A")

#Distclust Table
dist_pred <- matrix(data = c(30,7,13,125),nrow = 2, ncol = 2, byrow = TRUE)
dist_table <- as.table(dist_pred)
confusionMatrix(data = dist_table, positive ="A")

#iterSubspace Table
iter_pred <- matrix(data = c(32,7,11,125),nrow = 2, ncol = 2, byrow = TRUE)
iter_table <- as.table(iter_pred)
confusionMatrix(data = iter_table, positive ="A")
```

```
#funclust Table
fun_pred <- matrix(data = c(18,62,25,70),nrow = 2, ncol = 2, byrow = TRUE)
fun_table <- as.table(fun_pred)
confusionMatrix(data = fun_table, positive ="A")

#fscm Table
fscm_pred <- matrix(data = c(35,6,8,126),nrow = 2, ncol = 2, byrow = TRUE)
fscm_table <- as.table(fscm_pred)
confusionMatrix(data = fscm_table, positive ="A")

#waveClust Table
wave_pred <- matrix(data = c(41,53,2,79),nrow = 2, ncol = 2, byrow = TRUE)
wave_table <- as.table(wave_pred)
confusionMatrix(data = wave_table, positive ="A")

#Generate 147 random row numbers to choose rows for consensus set
set.seed(12345)
rownum <- sample(1:175, size = 147)
ts1_ids <- setdiff(1:175, rownum) #Get id that were not chosen in rownum

#Make a consensus training set of 147 with expert ratings
consensus_e <- two_curves02e[rownum,]
#35 Obstructed Curves
#112 Not Obstructed Curves

#Make a consensus training set of 147 with no expert ratings
consensus <- two_curves_02[rownum,]



#Perform K-Means on Consensus Training Set of size 147
```

```
#Apply K-Means clustering on this dataset


set.seed(12345)

k_twocurves <- kmeans.fd(consensus,ncl = 2)

#ncl = number of clusters (groups) to classify

k_twocurves


k_twocurves_center <- k_twocurves$centers[["data"]]


par(mfrow=c(1,1))

plot(k_twocurves_center[1,], type = "l", col = "red", main = "K-Mean Curves for Combined Kidney
Curves", xlab = "Time Points", ylab = "Concentration in Kidney") #Curve 1

lines(k_twocurves_center[2,], col = "blue")

legend("topleft", legend = c("Obstructed", "Not Obstructed"), col = c("red","blue"), lwd = 1, bty = "n")



#Determine how many curves belong to each cluster

twocurves_cluster <- k_twocurves$cluster

table(twocurves_cluster)

length(twocurves_cluster)

#1- Obstructed

#2- Not Obstructed


#Expert and K-Means Cluster Group

twocurves_kmeans_score <- cbind(consensus_e[,c(100:102)],twocurves_cluster)


count(twocurves_kmeans_score, vars = c("AT.L", "twocurves_cluster") )


#Functional Data Analysis Clustering

library(funcy)
```

```
#Get data into fdata object
twocurves_format <- formatFuncy(data = as.matrix(t(consensus)), format = "Format1")

set.seed(123456)
twocurvesfit <- funcit(twocurves_format , k = 2, methods = c(1:7))

summary(twocurvesfit)

#Plot the clustered curves
plot(twocurvesfit, legendPlace = "topleft")
plot(twocurvesfit, type = "centers", legendPlace = "bottomright")
#Cluster 1- Not Obstructed
#Cluster 2 - Obstructed

#Collect all the cluster information
twocurves_clusters <- as.data.frame(Cluster(twocurvesfit))

#Attach expert ratings onto this cluster data frame
twocurves_clusters <- cbind(consensus_e[,c(100:102)],twocurves_clusters)

#Get Cross Tabulation Counts for tables
count(twocurves_clusters, vars = c("AT.L", "fitfclust") )
count(twocurves_clusters, vars = c("AT.L", "distclust") )
count(twocurves_clusters, vars = c("AT.L", "iterSubspace") )
count(twocurves_clusters, vars = c("AT.L", "funclust") )
count(twocurves_clusters, vars = c("AT.L", "waveclust") )
count(twocurves_clusters, vars = c("AT.L", "fscm") )

#Calculate kappa, sensitivity, specificity, NPV, and PPV for each 2x2 clustering table
```

```
install.packages("caret")

install.packages("e1071")

library(caret)

library(e1071)


?confusionMatrix


#K-Means Table

kmean_pred <- matrix(data = c(15,0,20,112),nrow = 2, ncol = 2, byrow = TRUE)

kmean_table <- as.table(kmean_pred)

confusionMatrix(data = kmean_table, positive ="A")


#Fitfclust Table

fitf_pred <- matrix(data = c(18,44,17,68),nrow = 2, ncol = 2, byrow = TRUE)

fitf_table <- as.table(fitf_pred)

confusionMatrix(data = fitf_table, positive ="A")


#Distclust Table

dist_pred <- matrix(data = c(25,6,10,106),nrow = 2, ncol = 2, byrow = TRUE)

dist_table <- as.table(dist_pred)

confusionMatrix(data = dist_table, positive ="A")


#iterSubspace Table

iter_pred <- matrix(data = c(27,6,8,106),nrow = 2, ncol = 2, byrow = TRUE)

iter_table <- as.table(iter_pred)

confusionMatrix(data = iter_table, positive ="A")


#funclust Table

fun_pred <- matrix(data = c(25,72,10,40),nrow = 2, ncol = 2, byrow = TRUE)

fun_table <- as.table(fun_pred)
```

```
confusionMatrix(data = fun_table, positive ="A")

#fscm Table
fscm_pred <- matrix(data = c(29,5,6,107),nrow = 2, ncol = 2, byrow = TRUE)
fscm_table <- as.table(fscm_pred)
confusionMatrix(data = fscm_table, positive ="A")

#waveClust Table
wave_pred <- matrix(data = c(32,29,3,83),nrow = 2, ncol = 2, byrow = TRUE)
wave_table <- as.table(wave_pred)
confusionMatrix(data = wave_table, positive ="A")




#Test Set 1 Procedure

#Constructing Test Set 1
test1e <- two_curves02e[ts1_ids,]  #With expert ratings
test1 <- two_curves_02[ts1_ids,] #No expert ratings

count(test1e, vars = c("AT.L"))
#8 Obstructed
#20 Not Obstructed

set.seed(12345)
#Include one curve at a time procedure
kmeans_test <- list()

#For every iteration, perform k-means
#With every iteration, only add one test set curve info. to the test set
```

```
for(i in 1: nrow(test1)) {


  data_i <- rbind(consensus, test1[i,])  #Make the data for clustering procedure
  kmeans_test[[i]] <- kmeans.fd(data_i,ncl = 2)


}


#Re-labeling the clustering groups to match with the expert score scale
results <- c()
for(i in 1:nrow(test1)) {
  if (sum(kmeans_test[[i]][["centers"]]$data[1,95:99]) <
      sum(kmeans_test[[i]][["centers"]]$data[2,95:99])) {
    if (kmeans_test[[i]][["cluster"]][148] == 1) {
      results[i] <- 0
    } else if (kmeans_test[[i]][["cluster"]][148] == 2) {
      results[i] <- 2
    }
  }
  else if (sum(kmeans_test[[i]][["centers"]]$data[1,95:99]) >=
           sum(kmeans_test[[i]][["centers"]]$data[2,95:99])) {
    if (kmeans_test[[i]][["cluster"]][148] == 1) {
      results[i] <- 2
    } else if (kmeans_test[[i]][["cluster"]][148] == 2) {
      results[i] <- 0
    }
  }
}


table(results, test1e[,"AT.L"])
table(test1e[,"AT.L"], results)
```

```
#K-Means Table
kmean_pred <- matrix(data = c(3,0,5,20),nrow = 2, ncol = 2, byrow = TRUE)
kmean_table <- as.table(kmean_pred)
confusionMatrix(data = kmean_table, positive ="A")

#iterSubspace Time
library(funcy)

#Include one curve at a time procedure

iter_test <- list() #Initialize empty list to store itersubspace results
set.seed(12345)
#For every iteration, perform k-means
#With every iteration, only add one test set curve info. to the test set
for(i in 1: nrow(test1)) {

  data <- rbind(consensus, test1[i,])  #Make the data for clustering procedure
  data_format <- formatFuncy(data = as.matrix(t(data)), format = "Format1") #Format data to fdata object
  iter_test[[i]] <- funcit(data_format, k = 2, methods = "iterSubspace")

}

#Re-labeling the clustering groups to match with the expert score scale
results <- c()
for(i in 1:nrow(test1)) {
  if (sum(Center(iter_test[[i]])[95:99,1]) <
      sum(Center(iter_test[[i]])[95:99,2])) {
    if (Cluster(iter_test[[i]])[148] == 1) {
      results[i] <- 0
    } else if (Cluster(iter_test[[i]])[148] == 2) {
      results[i] <- 2
    }
  }
  else if (sum(Center(iter_test[[i]])[95:99,1]) >=
           sum(Center(iter_test[[i]])[95:99,2])) {
    if (Cluster(iter_test[[i]])[148] == 1) {
      results[i] <- 2
    } else if (Cluster(iter_test[[i]])[148] == 2) {
      results[i] <- 0
    }
  }
}

table(results, test1e[,"AT.L"])
table(test1e[,"AT.L"], results)

#Iter Table
iter_pred <- matrix(data = c(5,1,3,19),nrow = 2, ncol = 2, byrow = TRUE)
iter_table <- as.table(iter_pred)
confusionMatrix(data = iter_table, positive ="A")

#FSCM Time

#Include one curve at a time procedure

fscm_test <- list() #Initialize empty list to store fscm results

set.seed(12345)
```

```
#For every iteration, perform k-means
#With every iteration, only add one test set curve info. to the test set
for(i in 1: nrow(test1)) {

  data <- rbind(consensus, test1[i,])  #Make the data for clustering procedure
  data_format <- formatFuncy(data = as.matrix(t(data)), format = "Format1") #Format data to fdata object
  fscm_test[[i]] <- funcit(data_format, k = 2, methods = "fscm")

}

#Re-labeling the clustering groups to match with the expert score scale
results <- c()
for(i in 1:nrow(test1)) {
  if (sum(Center(fscm_test[[i]])[95:99,1]) <
      sum(Center(fscm_test[[i]])[95:99,2])) {
    if (Cluster(fscm_test[[i]])[148] == 1) {
      results[i] <- 0
    } else if (Cluster(fscm_test[[i]])[148] == 2) {
      results[i] <- 2
    }
  }
  else if (sum(Center(fscm_test[[i]])[95:99,1]) >=
           sum(Center(fscm_test[[i]])[95:99,2])) {
    if (Cluster(fscm_test[[i]])[148] == 1) {
      results[i] <- 2
    } else if (Cluster(fscm_test[[i]])[148] == 2) {
      results[i] <- 0
    }
  }
}
```

```
table(results, test1e[,"AT.L"])
table(test1e[,"AT.L"], results)

#FSCM Table
fscm_pred <- matrix(data = c(6,1,2,19),nrow = 2, ncol = 2, byrow = TRUE)
fscm_table <- as.table(fscm_pred)
confusionMatrix(data = fscm_table, positive ="A")



#DistClust Procedure
#Include one curve at a time procedure

dist_test <- list() #Initialize empty list to store fscm results

set.seed(12345)
#For every iteration, perform k-means
#With every iteration, only add one test set curve info. to the test set
for(i in 1: nrow(test1)) {

  data <- rbind(consensus, test1[i,])  #Make the data for clustering procedure
  data_format <- formatFuncy(data = as.matrix(t(data)), format = "Format1") #Format data to fdata object
  dist_test[[i]] <- funcit(data_format, k = 2, methods = "distclust")

}

#Re-labeling the clustering groups to match with the expert score scale
results <- c()
for(i in 1:nrow(test1)) {
  if (sum(Center(dist_test[[i]])[95:99,1]) <
      sum(Center(dist_test[[i]])[95:99,2])) {
    if (Cluster(dist_test[[i]])[148] == 1) {
      results[i] <- 0
    } else if (Cluster(dist_test[[i]])[148] == 2) {
      results[i] <- 2
    }
  }
  else if (sum(Center(dist_test[[i]])[95:99,1]) >=
           sum(Center(dist_test[[i]])[95:99,2])) {
    if (Cluster(dist_test[[i]])[148] == 1) {
      results[i] <- 2
    } else if (Cluster(dist_test[[i]])[148] == 2) {
      results[i] <- 0
    }
  }
}

table(results, test1e[,"AT.L"])
table(test1e[,"AT.L"], results)

#Distclust Table
dist_pred <- matrix(data = c(5,1,3,19),nrow = 2, ncol = 2, byrow = TRUE)
dist_table <- as.table(dist_pred)
confusionMatrix(data = dist_table, positive ="A")



#Waveclust Procedure
#Include one curve at a time procedure

wave_test <- list() #Initialize empty list to store fscm results
```

```
set.seed(12345)
#For every iteration, perform waveclust
#With every iteration, only add one test set curve info. to the test set
for(i in 1: nrow(test1)) {

  data <- rbind(consensus, test1[i,])  #Make the data for clustering procedure
  data_format <- formatFuncy(data = as.matrix(t(data)), format = "Format1") #Format data to fdata object
  wave_test[[i]] <- funcit(data_format, k = 2, methods = "waveclust")

}

#Re-labeling the clustering groups to match with the expert score scale
results <- c()
for(i in 1:nrow(test1)) {
  if (sum(Center(wave_test[[i]])[95:99,1]) <
      sum(Center(wave_test[[i]])[95:99,2])) {
    if (Cluster(wave_test[[i]])[148] == 1) {
      results[i] <- 0
    } else if (Cluster(wave_test[[i]])[148] == 2) {
      results[i] <- 2
    }
  }
  else if (sum(Center(wave_test[[i]])[95:99,1]) >=
           sum(Center(wave_test[[i]])[95:99,2])) {
    if (Cluster(wave_test[[i]])[148] == 1) {
      results[i] <- 2
    } else if (Cluster(wave_test[[i]])[148] == 2) {
      results[i] <- 0
    }
  }
}

table(results, test1e[,"AT.L"])
table(test1e[,"AT.L"], results)

#Waveclust Table
wave_pred <- matrix(data = c(8,4,0,16),nrow = 2, ncol = 2, byrow = TRUE)
wave_table <- as.table(wave_pred)
confusionMatrix(data = wave_table, positive ="A")




#Fitfclust Procedure
#Include one curve at a time procedure
fitf_test <- list() #Initialize empty list to store fscm results

set.seed(12345)
#For every iteration, perform waveclust
#With every iteration, only add one test set curve info. to the test set
for(i in 1: nrow(test1)) {

  data <- rbind(consensus, test1[i,])  #Make the data for clustering procedure
  data_format <- formatFuncy(data = as.matrix(t(data)), format = "Format1") #Format data to fdata object
  fitf_test[[i]] <- funcit(data_format, k = 2, methods = "fitfclust")

}

#Re-labeling the clustering groups to match with the expert score scale
```

```
results <- c()
for(i in 1:nrow(test1)) {
  if (sum(Center(fitf_test[[i]])[95:99,1]) <
      sum(Center(fitf_test[[i]])[95:99,2])) {
    if (Cluster(fitf_test[[i]])[148] == 1) {
      results[i] <- 0
    } else if (Cluster(fitf_test[[i]])[148] == 2) {
      results[i] <- 2
    }
  }
  else if (sum(Center(fitf_test[[i]])[95:99,1]) >=
           sum(Center(fitf_test[[i]])[95:99,2])) {
    if (Cluster(fitf_test[[i]])[148] == 1) {
      results[i] <- 2
    } else if (Cluster(fitf_test[[i]])[148] == 2) {
      results[i] <- 0
    }
  }
}

table(results, test1e[,"AT.L"])
table(test1e[,"AT.L"], results)

#Fitfclust Table
fitf_pred <- matrix(data = c(1,6,7,14),nrow = 2, ncol = 2, byrow = TRUE)
fitf_table <- as.table(fitf_pred)
confusionMatrix(data = fitf_table, positive ="A")
```

```
#Funclust Procedure
#Include one curve at a time procedure
fun_test <- list() #Initialize empty list to store fscm results

set.seed(12345)
#For every iteration, perform waveclust
#With every iteration, only add one test set curve info. to the test set
for(i in 1: nrow(test1)) {

  data <- rbind(consensus, test1[i,])  #Make the data for clustering procedure
  data_format <- formatFuncy(data = as.matrix(t(data)), format = "Format1") #Format data to fdata object
  fun_test[[i]] <- funcit(data_format, k = 2, methods = "funclust")

}

#Re-labeling the clustering groups to match with the expert score scale
results <- c()
for(i in 1:nrow(test1)) {
  if (sum(Center(fun_test[[i]])[95:99,1]) <
      sum(Center(fun_test[[i]])[95:99,2])) {
    if (Cluster(fun_test[[i]])[148] == 1) {
      results[i] <- 0
    } else if (Cluster(fun_test[[i]])[148] == 2) {
      results[i] <- 2
    }
  }
  else if (sum(Center(fun_test[[i]])[95:99,1]) >=
           sum(Center(fun_test[[i]])[95:99,2])) {
    if (Cluster(fun_test[[i]])[148] == 1) {
      results[i] <- 2
    } else if (Cluster(fun_test[[i]])[148] == 2) {
      results[i] <- 0
    }
  }
}

table(results, test1e[,"AT.L"])
table(test1e[,"AT.L"], results)

#Funclust Table
fun_pred <- matrix(data = c(8,19,0,1),nrow = 2, ncol = 2, byrow = TRUE)
fun_table <- as.table(fun_pred)
confusionMatrix(data = fun_table, positive ="A")



#Test Set 2 Construction
#Testing Set Construction
two_curves_no1 <- subset(two_curves_new2all, AT.L == 2 & ED.L == 2 & RH.L == 1) #4
two_curves_no2 <- subset(two_curves_new2all, AT.L == 2 & ED.L == 1 & RH.L == 2) #0
two_curves_no3 <- subset(two_curves_new2all, AT.L == 1 & ED.L == 2 & RH.L == 2) #7

two_curves_no4 <- subset(two_curves_new2all, AT.L == 1 & ED.L == 1 & RH.L == 2) #3
two_curves_no5 <- subset(two_curves_new2all, AT.L == 1 & ED.L == 1 & RH.L == 0) #0
two_curves_no6 <- subset(two_curves_new2all, AT.L == 1 & ED.L == 2 & RH.L == 1) #1
two_curves_no7 <- subset(two_curves_new2all, AT.L == 2 & ED.L == 1 & RH.L == 1) #1
two_curves_no8 <- subset(two_curves_new2all, AT.L == 1 & ED.L == 0 & RH.L == 1) #1
two_curves_no9 <- subset(two_curves_new2all, AT.L == 0 & ED.L == 1 & RH.L == 1) #0

two_curves_no10 <- subset(two_curves_new2all, AT.L == 0 & ED.L == 0 & RH.L == 1) #3
two_curves_no11 <- subset(two_curves_new2all, AT.L == 0 & ED.L == 1 & RH.L == 0) #0
```

```
two_curves_no12 <- subset(two_curves_new2all, AT.L == 1 & ED.L == 0 & RH.L == 0) #3

two_curves_no13 <- subset(two_curves_new2all, AT.L == 0 & ED.L == 0 & RH.L == 2) #1

two_curves_no14 <- subset(two_curves_new2all, AT.L == 0 & ED.L == 2 & RH.L == 0) #3

two_curves_no15 <- subset(two_curves_new2all, AT.L == 2 & ED.L == 0 & RH.L == 0) #3


two_curves_no16 <- subset(two_curves_new2all, AT.L == 2 & ED.L == 2 & RH.L == 0) #1

two_curves_no17 <- subset(two_curves_new2all, AT.L == 2 & ED.L == 0 & RH.L == 2) #2

two_curves_no18 <- subset(two_curves_new2all, AT.L == 0 & ED.L == 2 & RH.L == 2) #1

two_curves_no19 <- subset(two_curves_new2all, AT.L == 1 & ED.L == 1 & RH.L == 1) #4


two_curves_no20 <- subset(two_curves_new2all, AT.L == 0 & ED.L == 1 & RH.L == 2) #0

two_curves_no21 <- subset(two_curves_new2all, AT.L == 0 & ED.L == 2 & RH.L == 1) #0

two_curves_no22 <- subset(two_curves_new2all, AT.L == 1 & ED.L == 2 & RH.L == 0) #0

two_curves_no23 <- subset(two_curves_new2all, AT.L == 1 & ED.L == 0 & RH.L == 2) #2

two_curves_no24 <- subset(two_curves_new2all, AT.L == 2 & ED.L == 0 & RH.L == 1) #1

two_curves_no25 <- subset(two_curves_new2all, AT.L == 2 & ED.L == 1 & RH.L == 0) #0


#Test Set, 41 curves

two_curves_test <- rbind(two_curves_no1, two_curves_no3, two_curves_no4,
two_curves_no6,two_curves_no7, two_curves_no8, two_curves_no10,two_curves_no12,
two_curves_no13, two_curves_no14, two_curves_no15, two_curves_no16, two_curves_no17,
two_curves_no18, two_curves_no19, two_curves_no23, two_curves_no24)

count(two_curves_test,var = "Majority")


#We want test set with no equivoval majority, just obstructed and not obstructed

#Get 28 curves now with expert ratings

two_curves_test2 <- subset(two_curves_test, Majority == 0 | Majority == 2)


#Test set with no expert ratings

two_curves_test2_noexp <- two_curves_test2[, -c(100:103)]
```

```
#Test Set 2 Analysis
#Include one curve at a time procedure
kmeans_test <- list()

#For every iteration, perform k-means
#With every iteration, only add one test set curve info. to the test set
for(i in 1: nrow(two_curves_test2)) {

  data_i <- rbind(consensus, two_curves_test2_noexp[i,])  #Make the data for clustering procedure
  kmeans_test[[i]] <- kmeans.fd(data_i,ncl = 2)

}

#Re-labeling the clustering groups to match with the expert score scale
results <- c()
for(i in 1:nrow(two_curves_test2)) {
  if (sum(kmeans_test[[i]][["centers"]]$data[1,95:99]) <
      sum(kmeans_test[[i]][["centers"]]$data[2,95:99])) {
    if (kmeans_test[[i]][["cluster"]][148] == 1) {
      results[i] <- 0
    } else if (kmeans_test[[i]][["cluster"]][148] == 2) {
      results[i] <- 2
    }
  }
  else if (sum(kmeans_test[[i]][["centers"]]$data[1,95:99]) >=
           sum(kmeans_test[[i]][["centers"]]$data[2,95:99])) {
    if (kmeans_test[[i]][["cluster"]][148] == 1) {
      results[i] <- 2
    } else if (kmeans_test[[i]][["cluster"]][148] == 2) {
      results[i] <- 0
    }
  }
}

table(results, two_curves_test2[,"Majority"])
table(two_curves_test2[,"Majority"], results)

#K-Means Table
kmean_pred <- matrix(data = c(2,1,12,13),nrow = 2, ncol = 2, byrow = TRUE)
kmean_table <- as.table(kmean_pred)
confusionMatrix(data = kmean_table, positive ="A")



#iterSubspace Time
library(funcy)

#Include one curve at a time procedure

iter_test <- list() #Initialize empty list to store itersubspace results
set.seed(12345)
#For every iteration, perform k-means
#With every iteration, only add one test set curve info. to the test set
for(i in 1: nrow(two_curves_test2)) {

  data <- rbind(consensus, two_curves_test2_noexp[i,])  #Make the data for clustering procedure
  data_format <- formatFuncy(data = as.matrix(t(data)), format = "Format1") #Format data to fdata object
  iter_test[[i]] <- funcit(data_format, k = 2, methods = "iterSubspace")

}
```

```
#Re-labeling the clustering groups to match with the expert score scale
results <- c()
for(i in 1:nrow(two_curves_test2)) {
  if (sum(Center(iter_test[[i]])[95:99,1]) <
      sum(Center(iter_test[[i]])[95:99,2])) {
    if (Cluster(iter_test[[i]])[148] == 1) {
      results[i] <- 0
    } else if (Cluster(iter_test[[i]])[148] == 2) {
      results[i] <- 2
    }
  }
  else if (sum(Center(iter_test[[i]])[95:99,1]) >=
           sum(Center(iter_test[[i]])[95:99,2])) {
    if (Cluster(iter_test[[i]])[148] == 1) {
      results[i] <- 2
    } else if (Cluster(iter_test[[i]])[148] == 2) {
      results[i] <- 0
    }
  }
}

table(results, two_curves_test2[,"Majority"])
table(two_curves_test2[,"Majority"], results)


#Iter Table
iter_pred <- matrix(data = c(4,5,10,9),nrow = 2, ncol = 2, byrow = TRUE)
iter_table <- as.table(iter_pred)
confusionMatrix(data = iter_table, positive ="A")
```

```
#FSCM Time

#Include one curve at a time procedure

fscm_test <- list() #Initialize empty list to store fscm results

set.seed(12345)
#For every iteration, perform k-means
#With every iteration, only add one test set curve info. to the test set
for(i in 1: nrow(two_curves_test2)) {

  data <- rbind(consensus, two_curves_test2_noexp[i,])  #Make the data for clustering procedure
  data_format <- formatFuncy(data = as.matrix(t(data)), format = "Format1") #Format data to fdata object
  fscm_test[[i]] <- funcit(data_format, k = 2, methods = "fscm")

}

#Re-labeling the clustering groups to match with the expert score scale
results <- c()
for(i in 1:nrow(two_curves_test2)) {
  if (sum(Center(fscm_test[[i]])[95:99,1]) <
      sum(Center(fscm_test[[i]])[95:99,2])) {
    if (Cluster(fscm_test[[i]])[148] == 1) {
      results[i] <- 0
    } else if (Cluster(fscm_test[[i]])[148] == 2) {
      results[i] <- 2
    }
  }
  else if (sum(Center(fscm_test[[i]])[95:99,1]) >=
           sum(Center(fscm_test[[i]])[95:99,2])) {
    if (Cluster(fscm_test[[i]])[148] == 1) {
      results[i] <- 2
    } else if (Cluster(fscm_test[[i]])[148] == 2) {
      results[i] <- 0
    }
  }
}

table(results, two_curves_test2[,"Majority"])
table(two_curves_test2[,"Majority"], results)

#FSCM Table
fscm_pred <- matrix(data = c(4,5,10,9),nrow = 2, ncol = 2, byrow = TRUE)
fscm_table <- as.table(fscm_pred)
confusionMatrix(data = fscm_table, positive ="A")



#DistClust Procedure
#Include one curve at a time procedure

dist_test <- list() #Initialize empty list to store fscm results

set.seed(12345)
#For every iteration, perform k-means
#With every iteration, only add one test set curve info. to the test set
for(i in 1: nrow(two_curves_test2)) {
  data <- rbind(consensus, two_curves_test2_noexp[i,])  #Make the data for clustering procedure
  data_format <- formatFuncy(data = as.matrix(t(data)), format = "Format1") #Format data to fdata object
  dist_test[[i]] <- funcit(data_format, k = 2, methods = "distclust")

}

#Re-labeling the clustering groups to match with the expert score scale
results <- c()
for(i in 1:nrow(two_curves_test2)) {
  if (sum(Center(dist_test[[i]])[95:99,1]) <
      sum(Center(dist_test[[i]])[95:99,2])) {
    if (Cluster(dist_test[[i]])[148] == 1) {
      results[i] <- 0
    } else if (Cluster(dist_test[[i]])[148] == 2) {
      results[i] <- 2
    }
  }
  else if (sum(Center(dist_test[[i]])[95:99,1]) >=
           sum(Center(dist_test[[i]])[95:99,2])) {
    if (Cluster(dist_test[[i]])[148] == 1) {
      results[i] <- 2
    } else if (Cluster(dist_test[[i]])[148] == 2) {
      results[i] <- 0
    }
  }
}

table(results, two_curves_test2[,"Majority"])
table(two_curves_test2[,"Majority"], results)
```

```
#Distclust Table
dist_pred <- matrix(data = c(4,10,5,9),nrow = 2, ncol = 2, byrow = TRUE)
dist_table <- as.table(dist_pred)
confusionMatrix(data = dist_table, positive ="A")



#Waveclust Procedure
#Include one curve at a time procedure

wave_test <- list() #Initialize empty list to store fscm results

set.seed(12345)
#For every iteration, perform waveclust
#With every iteration, only add one test set curve info. to the test set
for(i in 1: nrow(two_curves_test2)) {

  data <- rbind(consensus, two_curves_test2_noexp[i,])  #Make the data for clustering procedure
  data_format <- formatFuncy(data = as.matrix(t(data)), format = "Format1") #Format data to fdata object
  wave_test[[i]] <- funcit(data_format, k = 2, methods = "waveclust")

}

#Re-labeling the clustering groups to match with the expert score scale
results <- c()
for(i in 1:nrow(two_curves_test2)) {
  if (sum(Center(wave_test[[i]])[95:99,1]) <
      sum(Center(wave_test[[i]])[95:99,2])) {
    if (Cluster(wave_test[[i]])[148] == 1) {
      results[i] <- 0
    } else if (Cluster(wave_test[[i]])[148] == 2) {
      results[i] <- 2
    }
  }
  else if (sum(Center(wave_test[[i]])[95:99,1]) >=
           sum(Center(wave_test[[i]])[95:99,2])) {
    if (Cluster(wave_test[[i]])[148] == 1) {
      results[i] <- 2
    } else if (Cluster(wave_test[[i]])[148] == 2) {
      results[i] <- 0
    }
  }
}

table(results, two_curves_test2[,"Majority"])
table(two_curves_test2[,"Majority"], results)

#Waveclust Table
wave_pred <- matrix(data = c(8,7,6,7),nrow = 2, ncol = 2, byrow = TRUE)
wave_table <- as.table(wave_pred)
confusionMatrix(data = wave_table, positive ="A")



#Fitfclust Procedure
#Include one curve at a time procedure
fitf_test <- list() #Initialize empty list to store fscm results

set.seed(12345)
#For every iteration, perform waveclust
#With every iteration, only add one test set curve info. to the test set
for(i in 1: nrow(two_curves_test2)) {
  data <- rbind(consensus, two_curves_test2_noexp[i,])  #Make the data for clustering procedure
  data_format <- formatFuncy(data = as.matrix(t(data)), format = "Format1") #Format data to fdata object
  fitf_test[[i]] <- funcit(data_format, k = 2, methods = "fitfclust")

}

#Re-labeling the clustering groups to match with the expert score scale
results <- c()
for(i in 1:nrow(two_curves_test2)) {
  if (sum(Center(fitf_test[[i]])[95:99,1]) <
      sum(Center(fitf_test[[i]])[95:99,2])) {
    if (Cluster(fitf_test[[i]])[148] == 1) {
      results[i] <- 0
    } else if (Cluster(fitf_test[[i]])[148] == 2) {
      results[i] <- 2
    }
  }
  else if (sum(Center(fitf_test[[i]])[95:99,1]) >=
           sum(Center(fitf_test[[i]])[95:99,2])) {
    if (Cluster(fitf_test[[i]])[148] == 1) {
      results[i] <- 2
    } else if (Cluster(fitf_test[[i]])[148] == 2) {
      results[i] <- 0
    }
  }
}

table(results, two_curves_test2[,"Majority"])
table(two_curves_test2[,"Majority"], results)
```

```
#Fitfclust Table
fitf_pred <- matrix(data = c(5,1,9,13),nrow = 2, ncol = 2, byrow = TRUE)
fitf_table <- as.table(fitf_pred)
confusionMatrix(data = fitf_table, positive ="A")



#Funclust Procedure
#Include one curve at a time procedure
fun_test <- list() #Initialize empty list to store fscm results

set.seed(12345)
#For every iteration, perform waveclust
#With every iteration, only add one test set curve info. to the test set
for(i in 1: nrow(two_curves_test2)) {

  data <- rbind(consensus, two_curves_test2_noexp[i,])  #Make the data for clustering procedure
  data_format <- formatFuncy(data = as.matrix(t(data)), format = "Format1") #Format data to fdata object
  fun_test[[i]] <- funcit(data_format, k = 2, methods = "funclust")

}

#Re-labeling the clustering groups to match with the expert score scale
results <- c()
for(i in 1:nrow(two_curves_test2)) {
  if (sum(Center(fun_test[[i]])[95:99,1]) <
      sum(Center(fun_test[[i]])[95:99,2])) {
    if (Cluster(fun_test[[i]])[148] == 1) {
      results[i] <- 0
    } else if (Cluster(fun_test[[i]])[148] == 2) {
      results[i] <- 2
    }
  }
  else if (sum(Center(fun_test[[i]])[95:99,1]) >=
           sum(Center(fun_test[[i]])[95:99,2])) {
    if (Cluster(fun_test[[i]])[148] == 1) {
      results[i] <- 2
    } else if (Cluster(fun_test[[i]])[148] == 2) {
      results[i] <- 0
    }
  }
}

table(results, two_curves_test2[,"Majority"])
table(two_curves_test2[,"Majority"], results)

#Funclust Table
fun_pred <- matrix(data = c(13,12,1,2),nrow = 2, ncol = 2, byrow = TRUE)
fun_table <- as.table(fun_pred)
confusionMatrix(data = fun_table, positive ="A")



#Aim Three
install.packages("psych")
library(psych)
citation("psych")

#Calculate kappa and weighted kappa for fscm
fscm_pred <- matrix(data = c(4,5,10,9),nrow = 2, ncol = 2, byrow = TRUE)
weights <- matrix(data = c(0,1,1,0), nrow = 2, ncol = 2, byrow= TRUE)
```

```
weights
wkappa(fscm_pred, w = weights)

#Calculate kappa and weighted kappa for waveclust
wave_pred <- matrix(data = c(8,7,6,7),nrow = 2, ncol = 2, byrow = TRUE)
wkappa(wave_pred, w = weights)

#Calculate kappa and weighted kappa for k-means
kmean_pred <- matrix(data = c(2,1,12,13),nrow = 2, ncol = 2, byrow = TRUE)
wkappa(kmean_pred, w = weights)
```