Unsupervised Learning of Cancer Data

Connor Leipelt

Department of Mathematics and Statistics

University of Massachusetts Amherst

Amherst, Massachusetts

May 2024

Contents

List of Figures

1 Introduction

Cancer is one of the leading causes of death in America [3]. Because of this, it is important to research how it grows and how it can be treated. This project is focused on applying machine learning methods to cancer data to learn about the cancer growth process. We will be using unsupervised learning; a key area of machine learning. This project will be focused on a category of unsupervised learning called cluster methods.

Motivation for this project comes from the fact that cancer growth is notoriously difficult to generalize to large populations. Through clustering techniques one can research the correlation in each cluster group to find an explanation for specific growth patterns. We will investigate three methods of clustering on our data: Gaussian Mixture Models with the EM algorithm, Hierarchical Agglomerative clustering, and K means clustering. From here we will perform a qualitative evaluation of the methods (depending on the specified number of clusters) and see which method performs the best for each scenario.

The aim of this project is to cluster cancerous tumors based on their growth rates before and after receiving chemotherapy treatment. From this action, we hope to look into the specific clusters and investigate the correlation between *other* features of the patients that could give reasons as to why these patients have these specific growth rate behaviors.

2 Background on Data and Formulation of Problem

2.1 Data Specifics

2.1.1 Data Origin

This project will be using a set of data given by Professor Wang of Union College. The data and experiment origin comes from the following article: *Model-Based Tumor Growth*

Dynamics and Therapy Response in a Mouse Model of "de Novo" Carcinogenesis; all referenced experiment information comes from this paper [10]. The data is of cancerous tumors that came from Chinese hamster V79 cells which are cells taken from the lung of a male Chinese hamster in 1958 that exhibit fibroblast morphology [15]. These cells were transferred to immunocompetent E6/E7 double transgenic mice that were treated with DMBA/TPA. This means that the response of the mice's immune system was taken into account, and the cancer was initiated by 12-dimethylbenzanthracene (DMBA), a polycyclic aromatic hydrocarbon (PAH) and a carcinogen that initiates and promotes tumorigenesis [5]. DMBA was then paired with 12-O-tetradecanoylphorbol 13-acetate (TPA), a tumor promoter that stimulates cell proliferation [11]. This process allows for *de novo* carcinogenesis capturing the tumor initialization phase. We also note that the mice data had tumor progression and maintanence *in vivo* [10].

2.1.2 Important Data Features

This experiment divided the mice above into two important groups, those that received chemotherapy treatment and those that did not. Note that the experiment also had a third group which was not treated with DMBA/TPA that of course resulted in no tumors. Each mouse that grew tumors had three tumors measured for the experiment. These tumors were measured in only two dimensions therefore to calculate the volume of the tumors the third length was averaged from the other two lengths and an ellipsoid volume was computed (Note: this computation was a main component of another paper I've done; please inquire if interested). Once the tumors reached a length of 3mm in diameter, the mice were given one of two chemotherapy treatments, Imiquimod or 5-Fluorouracil [14], [1]. The experiment also gave two different dosages for each treatment; 50mg/kg or 100mg/kg. The mice were given these doses every 3-4 days and measured every 3-7 days. Note that if a tumor reached the size of 1cm the mouse was euthanized.

A key component to this project is the fact that we have the specific starting dates of injection which allows the creation of two main time periods, initial tumor growth and tumor growth after chemotherapy treatment was implemented.

From this structure we see there are five different categories of mice:

- No treatment
- Imiquimod 50mg/kg treatment
- Imiquimod 100mg/kg treatment
- 5-Fluorouracil 50mg/kg treatment
- 5-Fluorouracil 100mg/kg treatment

These five groups suggest that five clusters should arise from the data.

2.2 Formulation of Data for Unsupervised Learning

Important distinction: This data set contain 45 mice and 137 tumors (two mice had 4 tumors). This project will be ran on two data sets: (1) the data of all tumors and (2) the data of all mice. The data for mice is calculated through averaging the data for each of its tumors.

For this project to work we need to be able to compare the data. We are interested in the relationship between initial growth rate and growth rate after treatment is started. This means that we must calculate these two growth rates. **Also** note that some tumors did not start to form until after the treatment was given; for these tumors their initial growth rate is 0.

2.2.1 Growth Rate

The initial data was given in two dimensions which we generalized to volume. The next goal is to compare the change in these volumes. For this project the annual average growth rate, g, for each tumor was calculated. This rate assumes that the next day's volume is dependent on the growth of the previous at the same rate for each day. This can be calculated as follows:

$$g := (v_n/v_0)^{\frac{1}{n}} - 1,$$

where v_0 = initial volume and v_n = volume at day n. Here is the proof of the average annual growth rate:

Proof. Define the following given v_i = volume at day i:

$$v_1 = v_0 + g \cdot v_0.$$

From here we see that we can write $v_1 = v_0(1+g)$. Now if we define v_2 in the same fashion we have:

$$v_2 = v_1 + g \cdot v_1 = v_1(1+g) = v_0(1+g)^2.$$

Following this pattern inductively we see that:

$$v_n = v_0(1+g)^n.$$

Solving for g then gives us the annual average growth rate:

$$(v_n/v_0)^{\frac{1}{n}} - 1 = g.$$

This concludes the proof. □

2.2.2 Coding of the Growth Rate

Now that we have determined our chosen way of calculating growth rate we must implement this into code. To start, the data of the tumor volumes and dates of measurements were imported (note that the volume calculation was done for another paper and the code is quite long so it will not appear in the appendix because it was not made specifically for this project). The dates of injections were hand checked and programmed into the code to work with the dates of the measurements.

Once the data was imported, the PADCAT function in MATLAB was used to create a matrix containing all of the volume information and a matrix containing all of the specific measuring dates for each mouse [13]. From here the initial growth rate and treatment growth rate were calculated similarly.

Create a for loop that takes the initial volume, the final volume (volume at final date), the initial date, and the final date to compute the annual average growth rate. For the initial growth rate the final date is the date before treatment started and for treatment growth rate the initial date was the first day recorded after treatment was given and the final date was the final dates given in the data. Once the growth rate for every tumor has been computed, compute the growth rate for each mouse by taking all of its tumors' growth rates and averaging them; this action is how we calculated the data for mice.

The code for this can be found in appendix (A.1).

3 Unsupervised Learning Methods

Now that we have formatted the data into growth rates we can perform Unsupervised Learning. Each of the methods require that we specify the number of clusters before proceeding.

As stated earlier we have five different groups: no treatment, Imiquimod 50mg/kg, Imiquimod 100mg/kg, 5-Fluorouracil 50mg/kg, and 5-Fluorouracil 100mg/kg. This structure lends us to believe that specifying five clusters is optimal. For the sake of the project and investigation we let the number of clusters, k, range from 2 to 5 in our methods.

As said earlier, these processes will be used on **two** sets of data, the first being the data of **all tumors**, and the second being the data of **mice**.

Note that all of these methods can be found in *Probabilistic Machine Learning: An Introduction* by Murphy and a great explanation for these methods and further topics related to them can be found in the following citation [12] [17].

3.1 Gaussian Mixture Models and the EM Algorithm

The first clustering method that we are investigating is the use of Gaussian Mixture Models and the EM algorithm. A published explanation of this method can be found in *Probabilistic Machine Learning: An Introduction* by Murphy (specifically section 21.4.1) [12].

This method creates a generative model of our data. It creates a mixed Gaussian by combining Gaussians with specific weights labeled π_k:

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k).$$

To draw a sample from $\boldsymbol{x}$ from $p(\boldsymbol{x})$ we start by choosing a mixture component with probability π_k:

$$p(z = k) = \pi_k.$$

Then, given $z = k$ we draw the probability of x from:

$$p(x|z = k) = \mathcal{N}(x|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k),$$

which is the sample from the component's Gaussian.

This variable z is called the "latent variable" as it is not specifically observed, but still used to compute the probability of x. For this method to being we have our k Gaussians start with random π_k, $\boldsymbol{\mu}_k$, and $\boldsymbol{\Sigma}_k$ values.

Now that we understand this we can move on to the EM algorithm:

The first step, E (Expectation), is performed by starting with clusters defined by our Gaussians and then for each data point, we compute the responsibility of the point as follows (for data point $\boldsymbol{x}_n$):

$$r_{n,k} := \frac{\pi_k \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{k'=1}^{K} \pi_{k'} \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_{k'}, \boldsymbol{\Sigma}_{k'})} = \frac{p(z_n = k|\boldsymbol{\theta})p(\boldsymbol{x}_n|z_n = k, \boldsymbol{\theta})}{\sum_{k'=1}^{K} p(z_n = k'|\boldsymbol{\theta})p(\boldsymbol{x}_n|z_n = k'|\boldsymbol{\theta})}.$$

This responsibility of a point tells you whether or not you should be in a specific cluster. If x_n is likely to be in class k, the $r_{n,k}$ will be large (percentage wise) and if it is not likely for x_n to be in a certain k the $r_{n,k}$ for that class will be small (percentage wise). It is important to note that $r_{n,k}$ sums to 1!

Once you've computed the responsibilities for each point for each cluster, you perform the second step, M (Maximization). This step fixes the responsibility values and updates the parameters π_k, $\boldsymbol{\mu}_k$, and $\boldsymbol{\Sigma}_k$ for each cluster (Gaussian $z = k$). For π_k we compute a variable m_k that we can call the total responsibility allocated to cluster k:

$$m_k = \sum_n r_{n,k}.$$

From here we update π_k by making it equal $\frac{m_k}{m}$ where $m = \sum_k m_k$:

$$\pi_k = \frac{m_k}{m}.$$

This is really calculating the fraction of data point probabilities assigned to cluster k.

We then define the new $\boldsymbol{\mu}_k$ to be the weighted average (mean) of the data:

$$\boldsymbol{\mu}_k = \frac{1}{m_k} \sum_n r_{n,k} x_n.$$

Lastly, we updated the new $\boldsymbol{\Sigma}_k$ to be the weighted covariance of the assigned data using the new $\boldsymbol{\mu}_k$:

$$\frac{1}{m_k} \sum_n r_{n,k} (x_n - \boldsymbol{\mu}_k)^T (x_n - \boldsymbol{\mu}_k).$$

If we then go back to step E and iterate through this algorithm we see that each step increases the log-likelihood of our model which increases the model's fit to the data [16]. This means that the method will converge given some tolerance criterion!

The main problem with this method is that it can converge to local optima so it is important to initialize the Gaussians well. This is the essence of the model and the code for this model can be found in appendix (A.2).

Note that the explanation of this method was also made clear through the following citation [16].

3.2 Hierarchical Agglomerative Clustering Methods

Our next Unsupervised Learning clustering method is called Hierarchical Agglomerative clustering. A published explanation of this method can be found in *Probabilistic Machine*

Learning: An Introduction by Murphy (specifically section 21.2) [12].

This method is different then Gaussian Mixture Models because it starts with *every data point* being its own cluster. The key idea of these methods is that certain clusters will merge depending on the criteria of the specific Hierarchical Agglomerative clustering method.

To start one specifies how many clusters they want, k. Then the method iteratively merges clusters until there are k clusters left.

Note that each method can be visualized through a dendrogram! These graphs are to be read from the bottom up and explain which clusters merge together (they will be shown in section 4).

In this project we test four types of Hierarchical Agglomerative Clustering methods:

- Single Linkage
- Complete Linkage
- Average Linkage
- Ward Linkage

The first three methods can be found in Murphy and the last method is discussed in the following citation [12] [17].

3.2.1 Single Linkage

Single Linkage clustering, also sometimes called nearest neighbor clustering, calculates the distance between all data points. Which ever two points (clusters) are closest merge and become one new cluster. Once this has been done, the *minimum* distance between all members of different clusters (example shown for two groups G and H)

$$d_{SL}(G,H) = \min_{i \in G, i' \in H} d_{i,i'},$$

is computed once more and whichever two clusters have the closest minimum distance merge. This method of merging is performed until the specified number of clusters is reached [12]. The code for this method can be found in appendix (A.3).

3.2.2 Complete Linkage

Complete Linkage clustering, sometimes called furthest neighbor clustering, computes the *farthest* distance between two different clusters by computing the distance between all points and then computing the largest distance between a point in one cluster to a point in another cluster. Example shown for two groups G and H:

$$d_{CL}(G,H) = \max_{i \in G, i' \in H} d_{i,i'}.$$

From this, the methods merges the two clusters that have the minimum farthest distance. Once merged the method recalculates this specific distance for all clusters and merges again. This method continues until the specified number of clusters is reached [12].
The code for this method can be found in appendix (A.3).

3.2.3 Average Linkage

Average Linkage is a method that computes the average distance between all elements of a cluster with all elements of another cluster. This average distance is computed between all clusters as follows (example given for two groups G and H)

$$d_{avg}(G,H) = \frac{1}{n_G n_H} \sum_{i \in G} \sum_{i' \in H} d_{i,i'},$$

where n_G and n_H are the number of elements in groups G and H [12]. Once this distance is computed for each cluster compared to all other clusters, the two clusters with the smallest

average distance are merged. This method iterates through these steps until the specified number of clusters is reached.
The code for this method can be found in the appendix (A.3).

3.2.4 Ward Linkage

Ward Linkage computes the hypothetical in cluster variance if two clusters merged. This new in cluster variance is measured between all clusters and whichever two clusters has the smallest in cluster variance once merging become merged. Here is an explanation of the computation between two clusters G and H:

$$\begin{aligned} d_{ward}(G,H) &= \sum_{i \in G \cup H} \|\vec{x}_i - \vec{m}_{G \cup H}\|^2 - \sum_{i \in G} \|\vec{x}_i - \vec{m}_G\|^2 - \sum_{i \in H} \|\vec{x}_i - \vec{m}_H\|^2 \\ &= \frac{n_G n_H}{n_G + n_H} \|\vec{m}_G - \vec{m}_H\|^2, \end{aligned}$$

where $\vec{m}_j$ is the center of the cluster j, and n_j is the number of points in cluster j [2]. Once the new cluster is formed the in cluster variance is computed again for every combination of two groups and the two clusters with the smallest distance merge again. These steps are iterated until there are the specified number of clusters remaining [17].
The code for this method can be found in the appendix (A.3).

3.3 K means Method

The final clustering method of this paper is the K means method. A published explanation of this method can be found in *Probabilistic Machine Learning: An Introduction* by Murphy (specifically section 21.3) [12].
This algorithm starts by requiring the number of clusters you wish the data to be sorted into. Once you have defined this number, k, the methods starts by picking k random points

from your data set and labels them as the cluster centers $\boldsymbol{\mu}_k$. From here each data point, $\boldsymbol{x}_n$ computes its distance to the cluster centers:

$$z_n^* = \arg\min_k \|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|_2^2.$$

Once this has been done, the data point is assigned to whichever cluster that produced the minimum distance when computing the distance between the data point and the cluster center.

Once all of the points have been assigned a cluster, each cluster updates its center by the following:

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n:z_n=k} \boldsymbol{x}_n,$$

where N_k is the total number of data points assigned to that specific cluster. This action is taking all of the points assigned to a cluster and finding the average location of these clusters.

We can iterate these steps to convergence because this algorithm has a cost function that can be minimized [12]. One can choose to run this method until the new average $\boldsymbol{\mu}_k$ is the same as the previous for all clusters, or once the difference between the current and new $\boldsymbol{\mu}_k$ is below a given criterion for all clusters. One drawback of this method is that all of your clusters will be circle shaped and that the clusters are always Voronoi-diagrams of centers [17].

The code for this method can be found in the appendix (A.4).

4 Findings of the Unsupervised Learning Methods

This section of the paper is to show the findings of the three main clustering methods. **Note:** These methods were tested on **two** data sets, one being all tumors, 137 total, and the other being all mice data, 45 total; two mice had four tumors. The mice data was created by averaging the data for the tumors on each mouse.
Each method will test for number of clusters, k, ranging from two up to 5 ($k = 2, 3, 4, 5$).

4.1 Gaussian Mixture Models and EM Algorithm Findings

This Gaussian Mixture Models and EM algorithm code has **four** possible combinations of parameters that we vary in the code. These parameters are whether or not the covariance matrix is *diagonal* or *full* and whether or not the covariance matrix is shared meaning that all components have the same covariance matrix or they each have a personalized covariance matrix. The decision for this will be labeled *true* or *false* depending on if it is shared or not. Note that not all combinations are possible due to an ill-conditioned covariance matrix.
The code for this can be found in appendix (A.2), these methods were written with help from the following citation [6]. Note that for $k = 5$ the diag-true combination for the mice data gives an index error instead of an ill-conditioned error.

4.1.1 Gaussian Mixture Models Tumor Data

- The first set of graphs is for the Tumor Data, $k = 2$.

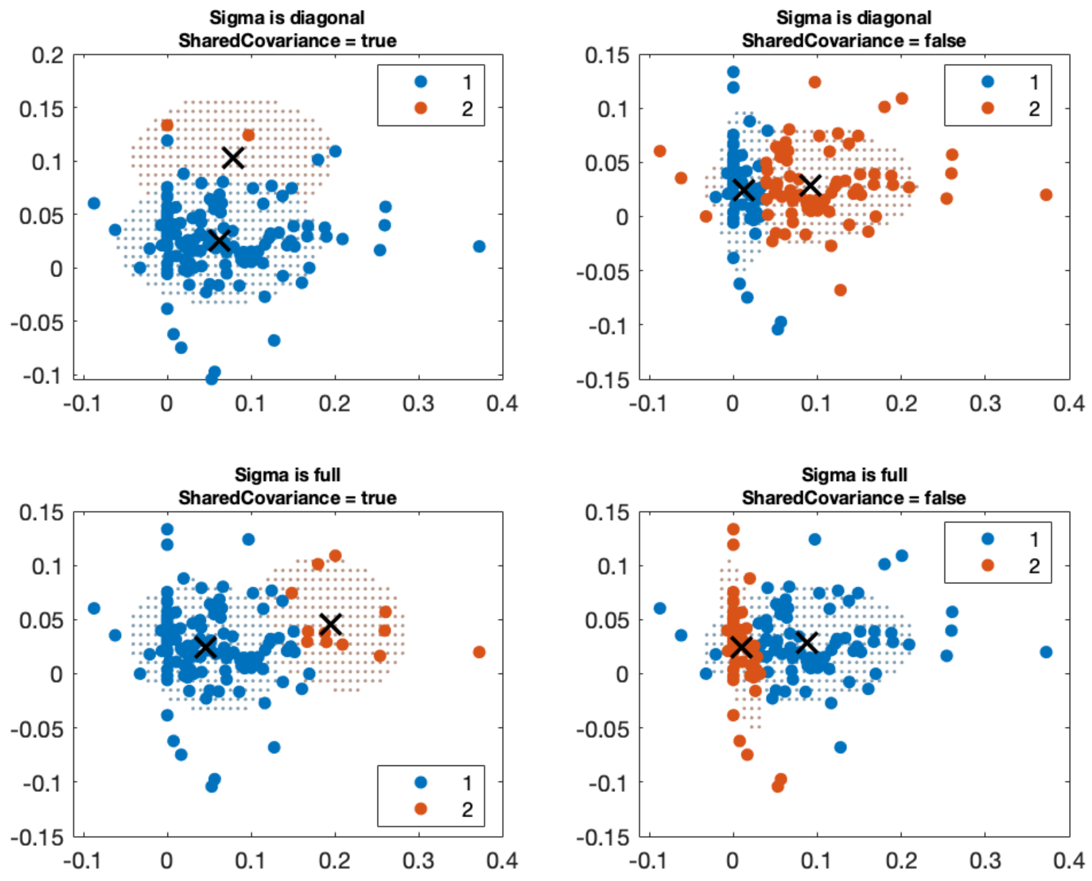


Figure 1: GMM Method Tumor Data k=2 All Combinations

- Tumor Data, $k = 3$

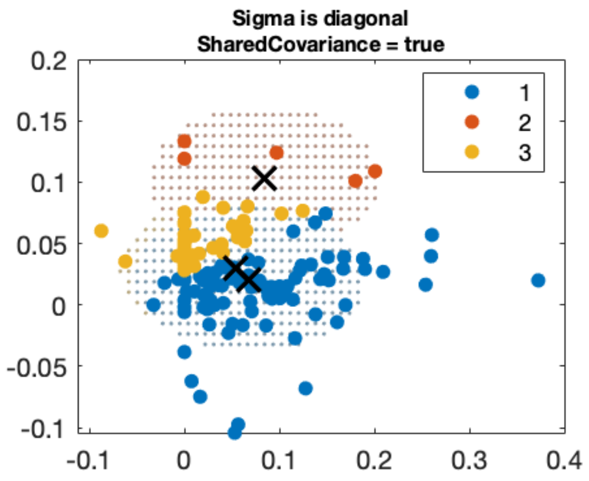


Figure 2: GMM Method Tumor Data k=3 Diag-True

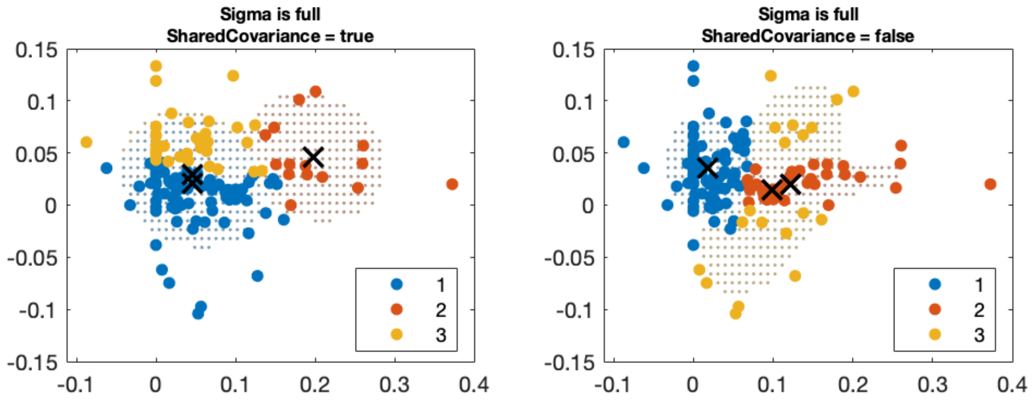


Figure 3: GMM Method Tumor Data k=3 Full-True, Full-False

- Tumor Data, $k = 4$

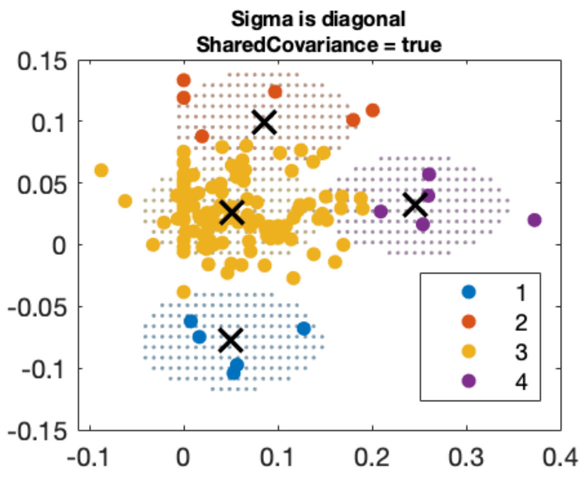


Figure 4: GMM Method Tumor Data k=4 Diag-True

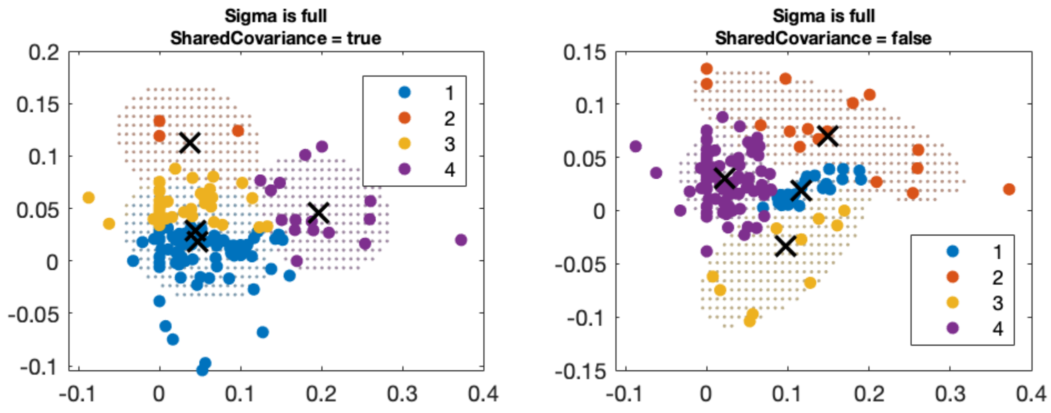


Figure 5: GMM Method Tumor Data k=4 Full-True, Full-False

- Tumor Data, $k = 5$

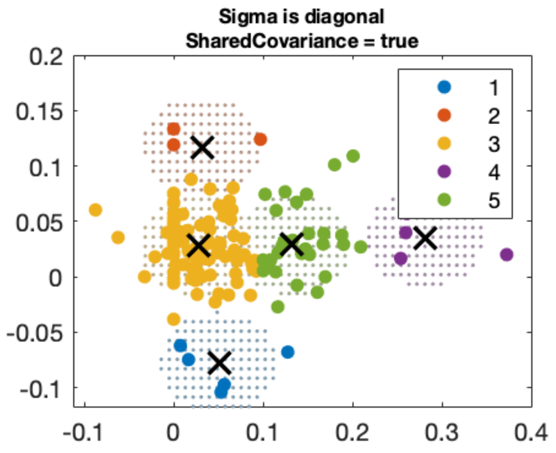


Figure 6: GMM Method Tumor Data k=5 Diag-True

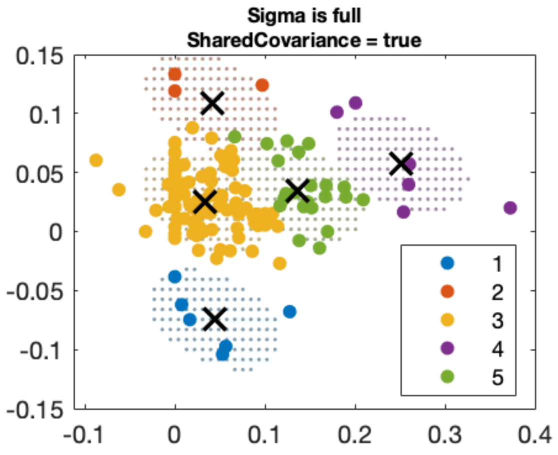


Figure 7: GMM Method Tumor Data k=5 Full-True

4.1.2 Gaussian Mixture Models Mouse Data

- We now look into Mice Data starting with $k = 2$.

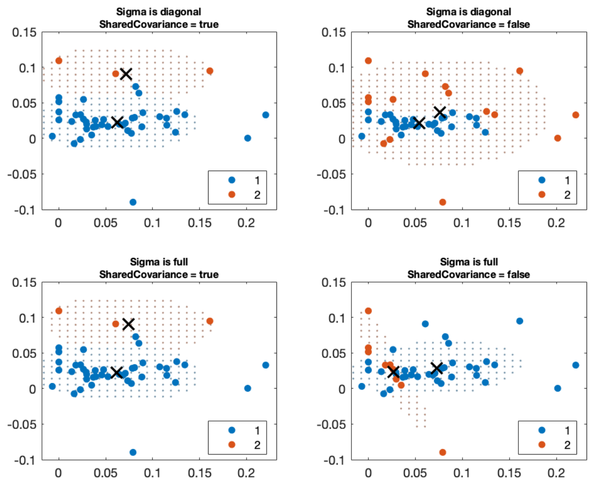


Figure 8: GMM Method Mouse Data k=2 All Combinations

- Mice Data, $k = 3$.

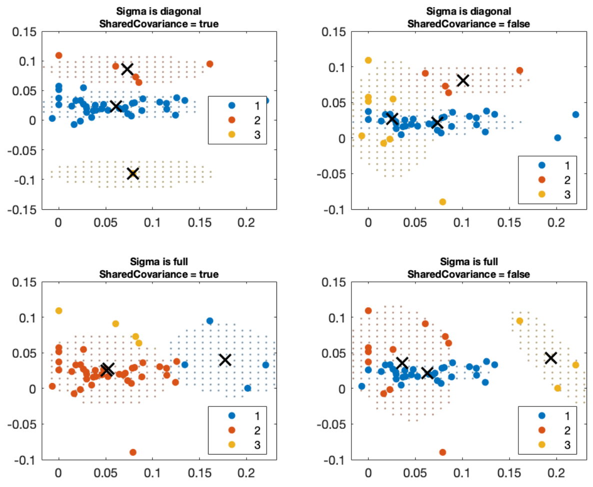
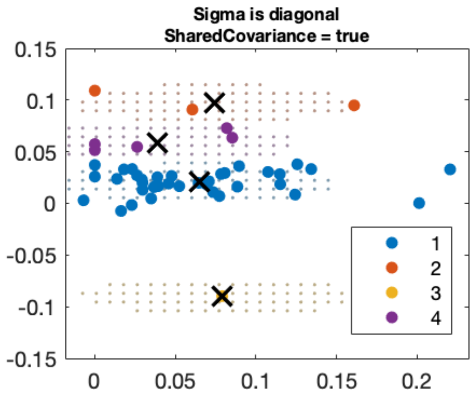


Figure 9: GMM Method Mouse Data k=3 All Combinations

- Mice Data, $k = 4$.



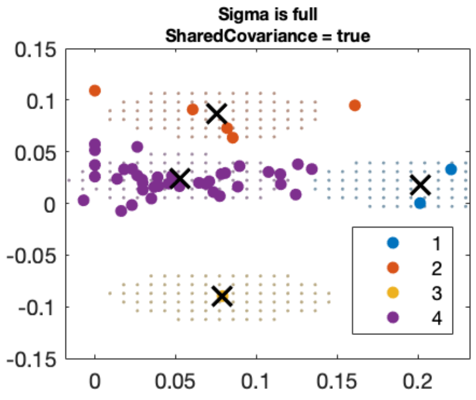


Figure 10: GMM Method Mouse Data k=4 Diagonal-True, Full-True

- Mice Data, $k = 5$.

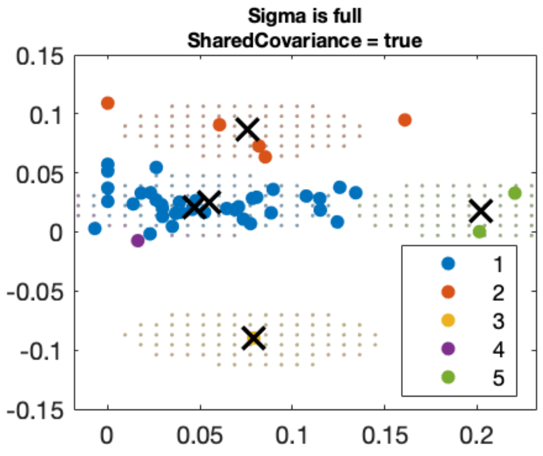


Figure 11: GMM Method Mouse Data k=5 Full-True

Note that all possible graphs from the code are present.

4.2 Hierarchical Agglomerative Methods Findings

In this section we investigate the performance of the Hierarchical Agglomerative cluster methods. There are four types of method used: *Single Linkage*, *Complete Linkage*, *Average Linkage*, and *Ward Linkage*. Each method has a dendrogram along with the cluster graphs. This dendrogram is read from bottom to top and shows the clusters that merged. (Note the dendrogram will look the same for $k = 2, 3, 4, 5$ as the only difference is that the clustering stops at the specified number of clusters).

The code for this can be found in appendix (A.3), these methods were written with the help from the following citations [4], [7], [9].

4.2.1 Hierarchical Agglomerative Tumor Data

- Graphs of Tumor Data Single Linkage method $k = 2, 3, 4, 5$ and its dendrogram.

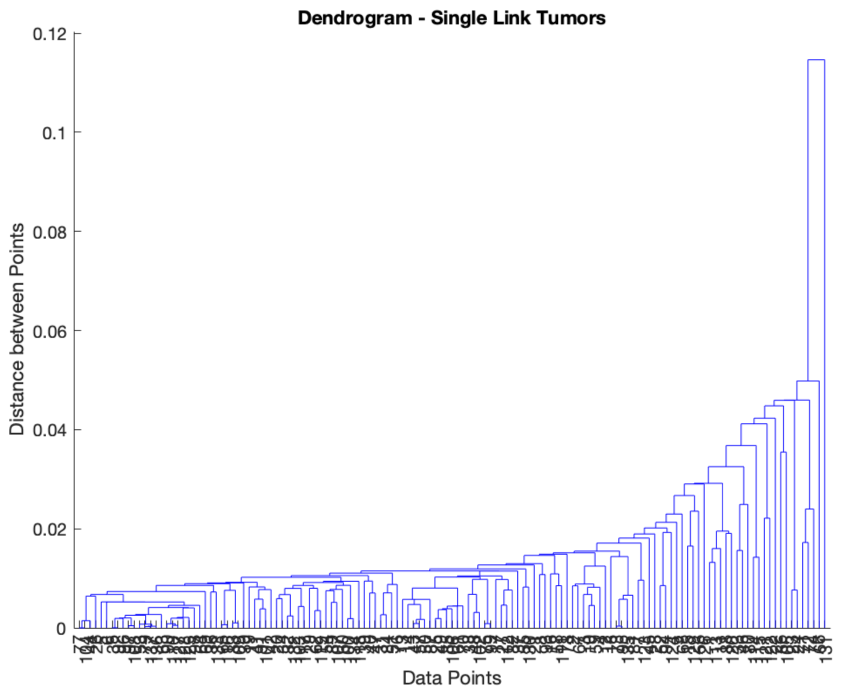


Figure 12: Hierarchical Agglo Method Tumor Data Single Linkage Dendrogram

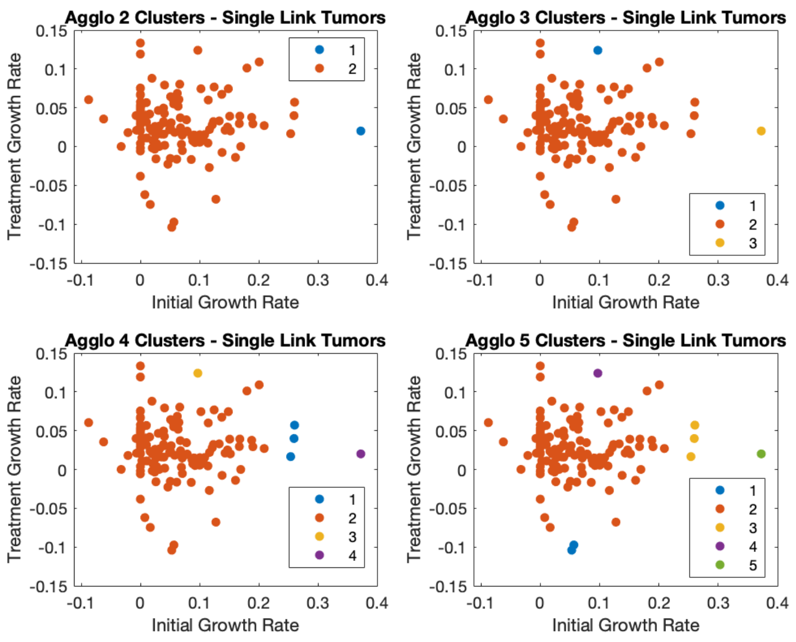


Figure 13: Hierarchical Agglo Method Tumor Data Single Linkage All k

- Tumor Data Complete Linkage method $k = 2, 3, 4, 5$ and its dendrogram.

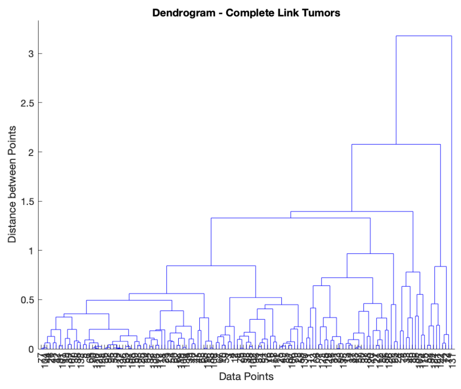


Figure 14: Hierarchical Agglo Method Tumor Data Complete Linkage Dendrogram

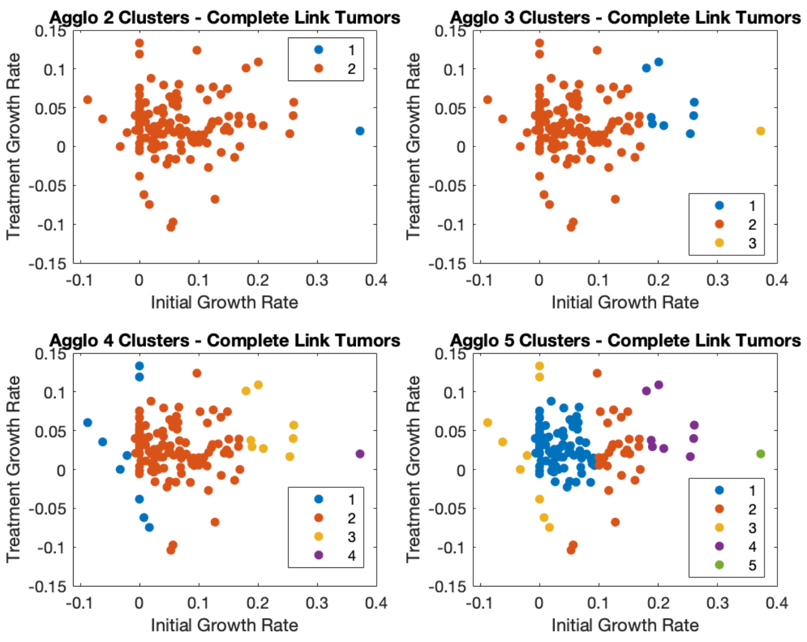


Figure 15: Hierarchical Agglo Method Tumor Data Complete Linkage All k

- Tumor Data Average Linkage method $k = 2, 3, 4, 5$ and its dendrogram.

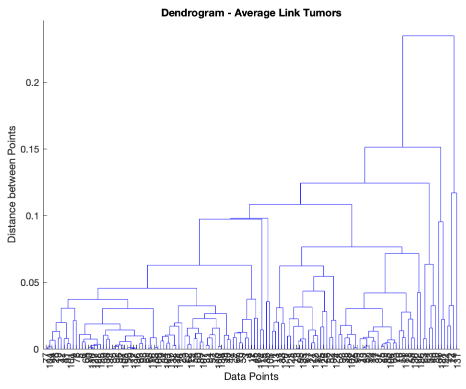


Figure 16: Hierarchical Agglo Method Tumor Data Average Linkage Dendrogram

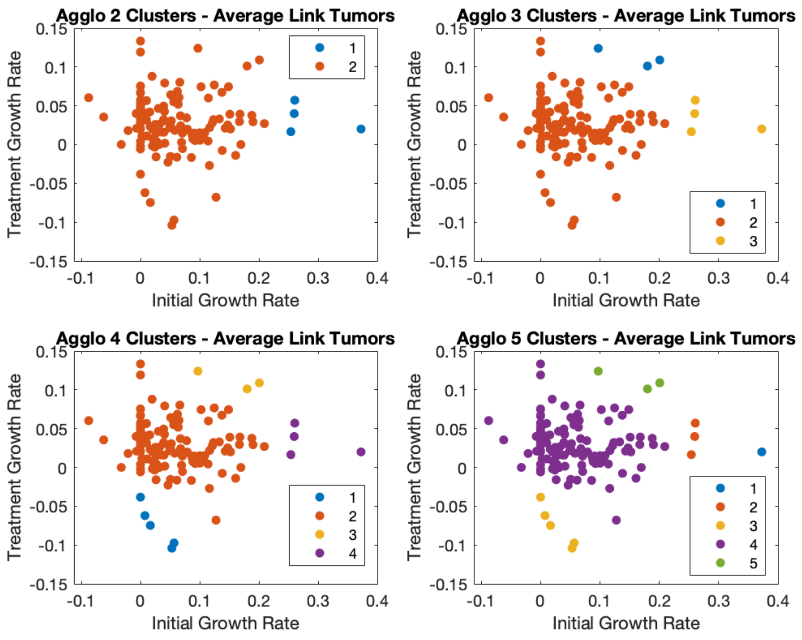


Figure 17: Hierarchical Agglo Method Tumor Data Average Linkage All k

- Tumor Data Ward Linkage method $k = 2, 3, 4, 5$ and its dendrogram.

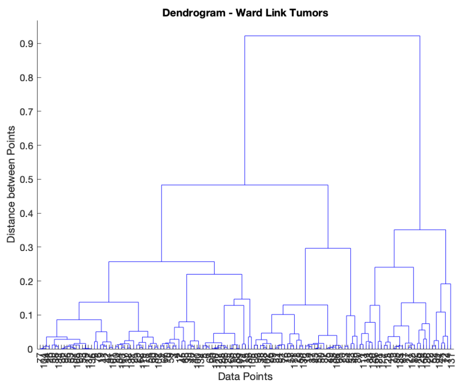


Figure 18: Hierarchical Agglo Method Tumor Data Ward Linkage Dendrogram

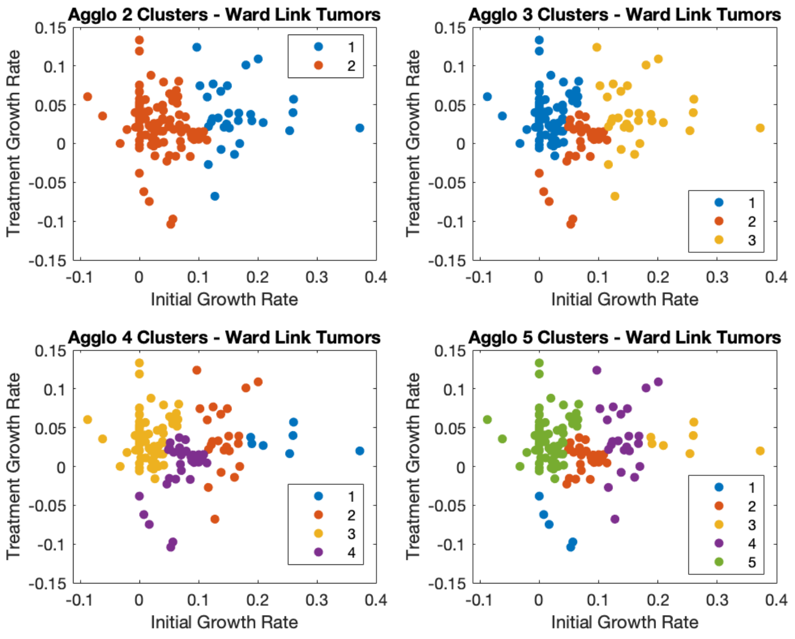


Figure 19: Hierarchical Agglo Method Tumor Data Ward Linkage All k

4.2.2 Hierarchical Agglomerative Mouse Data

- Graphs of Mice Data Single Linkage method $k = 2, 3, 4, 5$ and its dendrogram.

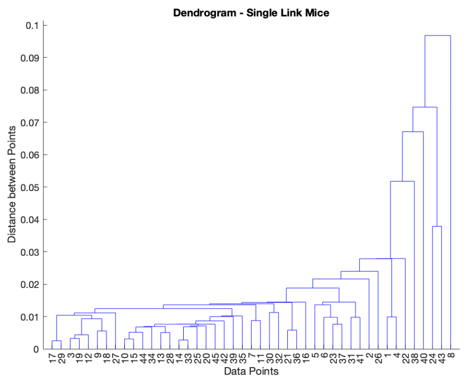


Figure 20: Hierarchical Agglo Method Mice Data Single Linkage Dendrogram

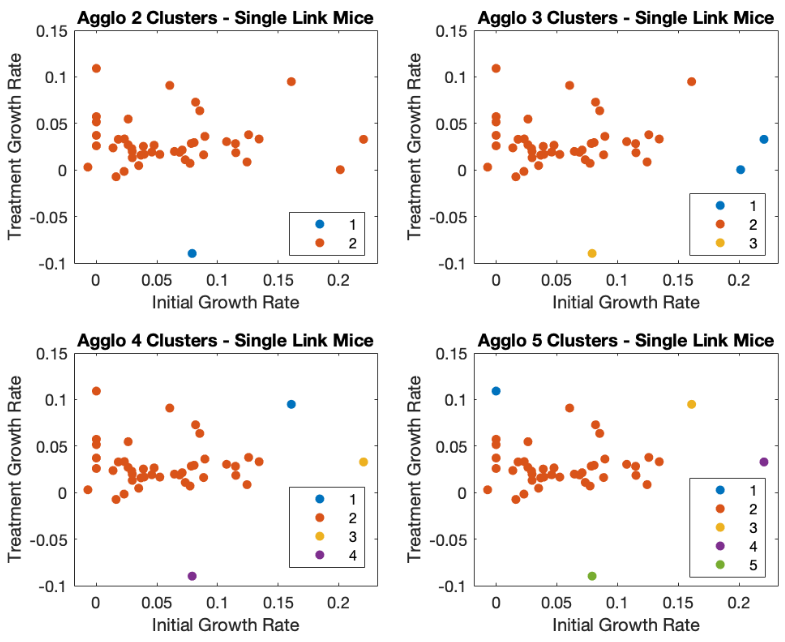


Figure 21: Hierarchical Agglo Method Mice Data Single Linkage All k

- Mice Data Complete Linkage method $k = 2, 3, 4, 5$ and its dendrogram.

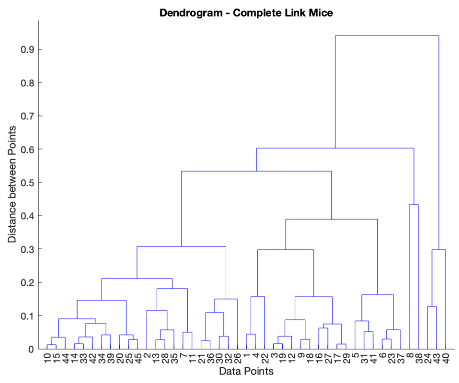


Figure 22: Hierarchical Agglo Method Mice Data Complete Linkage Dendrogram

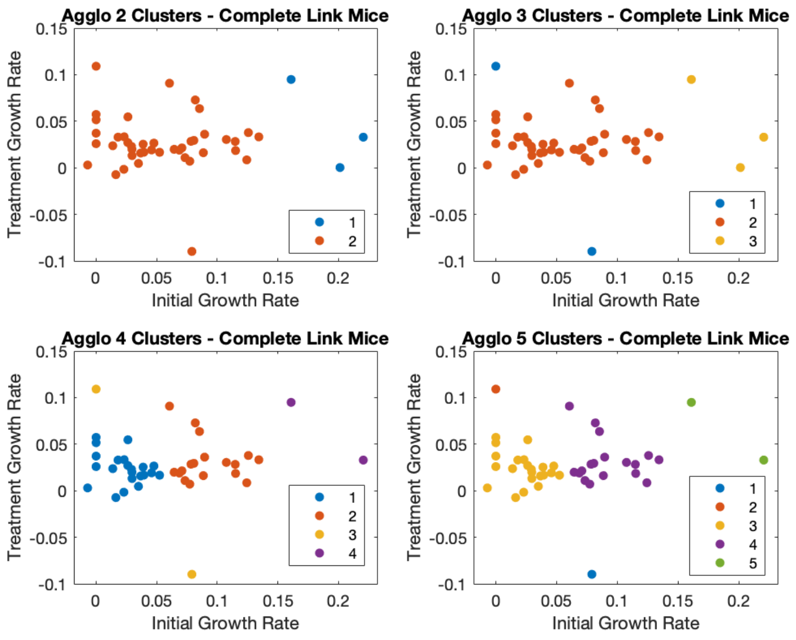


Figure 23: Hierarchical Agglo Method Mice Data Complete Linkage All k

- Mice Data Average Linkage method $k = 2, 3, 4, 5$ and its dendrogram.

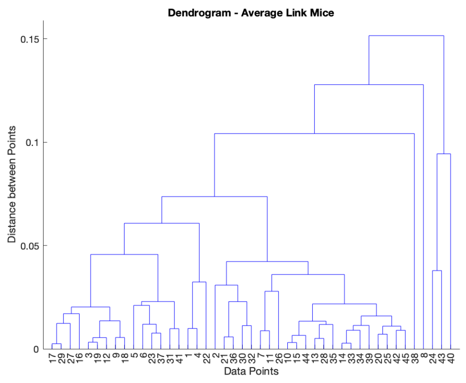


Figure 24: Hierarchical Agglo Method Mice Data Average Linkage Dendrogram

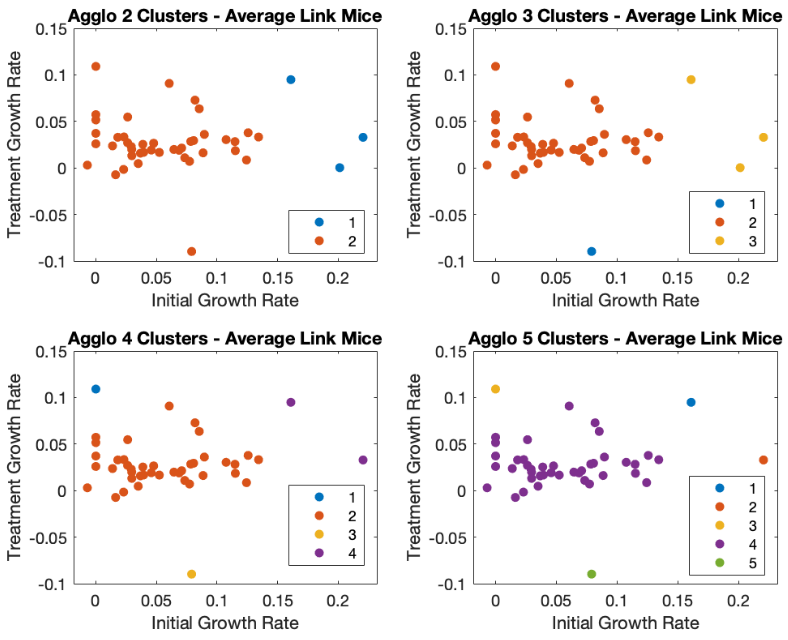


Figure 25: Hierarchical Agglo Method Mice Data Average Linkage All k

- Mice Data Ward Linkage method $k = 2, 3, 4, 5$ and its dendrogram.

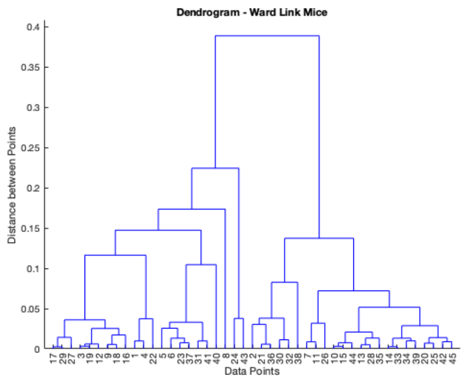


Figure 26: Hierarchical Agglo Method Mice Data Ward Linkage Dendrogram

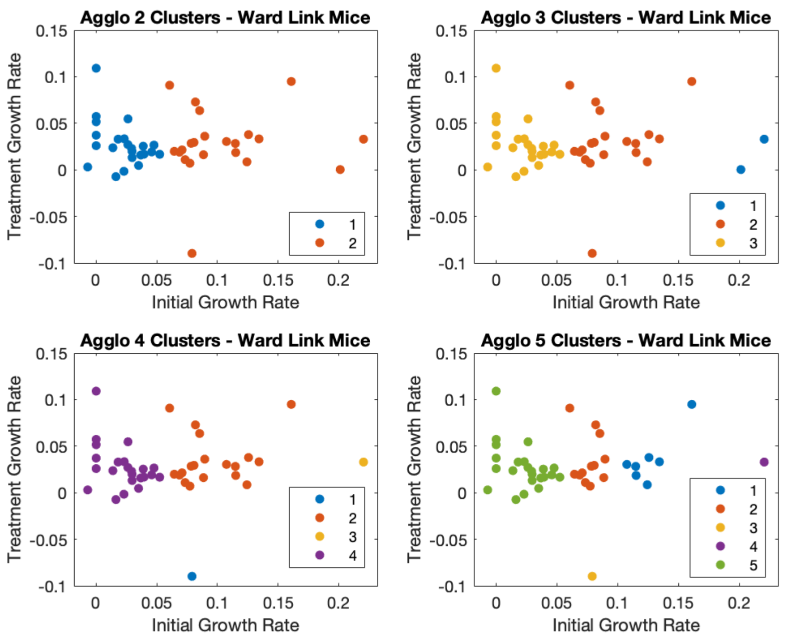


Figure 27: Hierarchical Agglo Method Mice Data Ward Linkage All k

4.3 K means Method Findings

Her we will investigate the performance of the K means method. This method is quite self-contained and has only one main method. That means that we will have four graphs for the Tumor Data for $k = 2, 3, 4, 5$ and four graphs for the Mice Data for $k = 2, 3, 4, 5$.

The code for this can be found in appendix (A.4), these methods were written with help from the following citation [8].

4.3.1 K means Tumor Data

- Graphs of Tumor Data K means Method $k = 2, 3, 4, 5$.

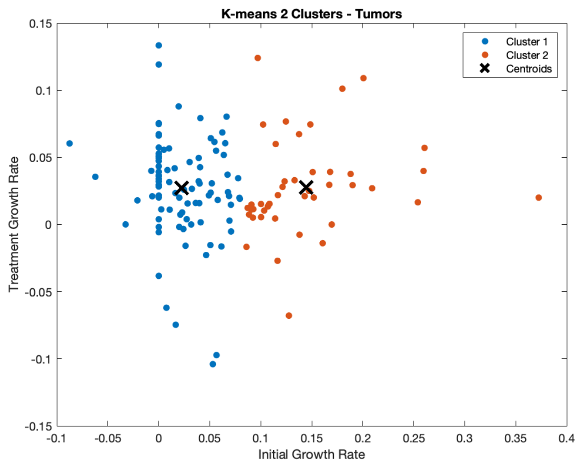


Figure 28: K means Method
Tumor Data $k = 2$

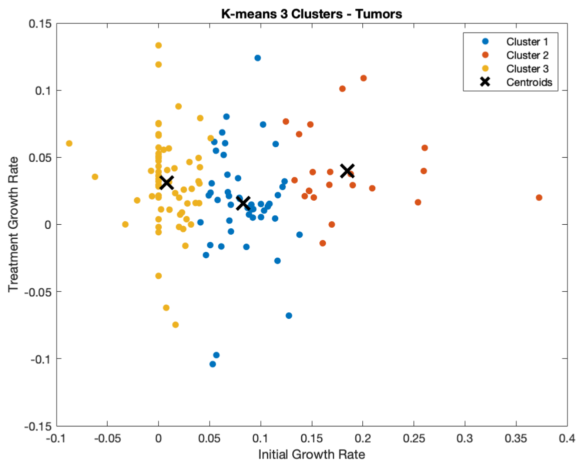


Figure 29: K means Method
Tumor Data $k = 3$

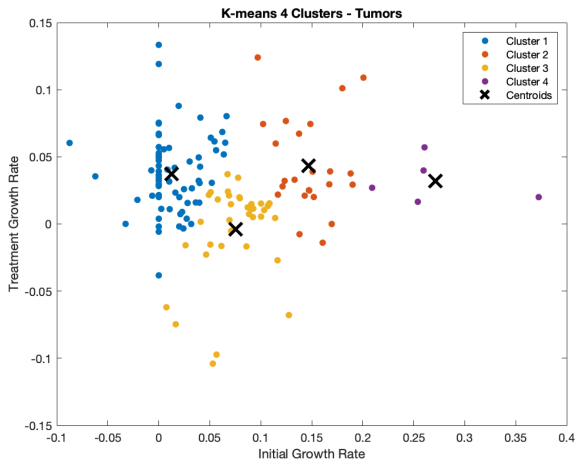


Figure 30: K means Method
Tumor Data $k = 4$

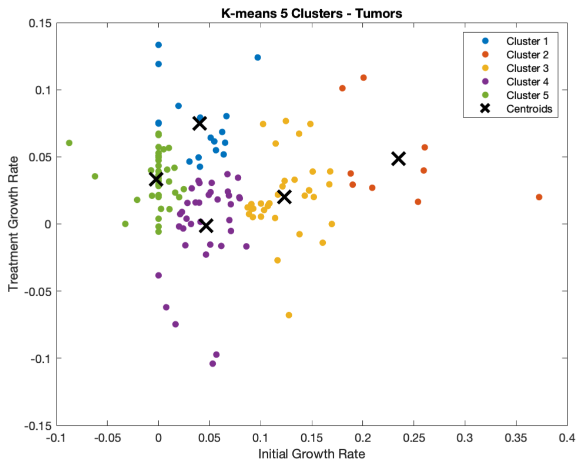


Figure 31: K means Method
Tumor Data $k = 5$

4.3.2 K means Mouse Data

- Graphs of Mice Data K means Method $k = 2, 3, 4, 5$.

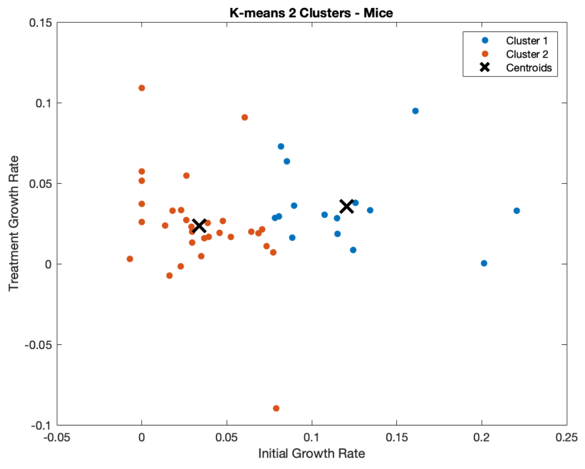


Figure 32: K means Method
Mice Data $k = 2$

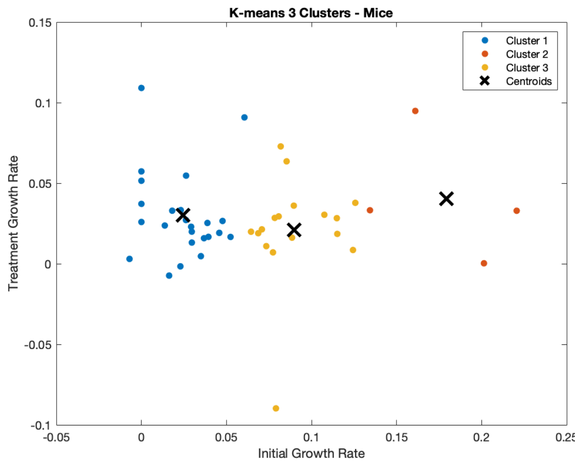


Figure 33: K means Method
Mice Data $k = 3$

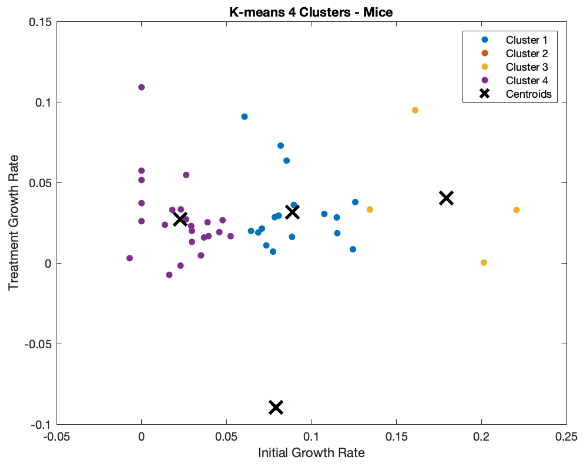


Figure 34: K means Method
Mice Data $k = 4$

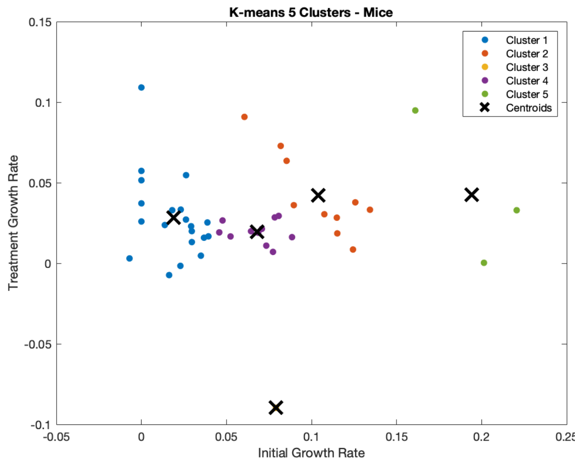


Figure 35: K means Method
Mice Data $k = 5$

5 Conclusion

We have successfully implemented three kinds of clustering methods onto two sets of data. For our first method, Gaussian Mixture Models with EM algorithm, we had four subcategories/ways to compute the clusters. We found that as the numbers of clusters increased some combinations were not able to be computed due to an ill-conditioned covariance matrix (one case of index error). For the second method, Hierarchical Agglomerative clustering methods, there were four main methods on how to merge clusters. Each method has their own use cases; for accurate clusters of non-singular size the average linkage and ward linkage performed the best from this group. For our final method, K means method, we found it to be self-containing and that it performed relatively well.

We will be performing a qualitative evaluation on both data sets and their graphs for each cluster size ranging from $k = 2$ to 5. Some important information for the rubric is that we are looking for graphs that cluster realistically and not asymptotically in one direction. This means that we want to stay away from vertical and horizontal lined clusters because this means that there is too much variance in one growth rate while the other, in comparison, had very little variation.

In the following two sections we pick the best method and graph for each cluster number for the data set.

5.1 Comparing Tumor Findings

For 2 cluster sets ($k = 2$) on Tumor Data the best cluster method was the K means method:

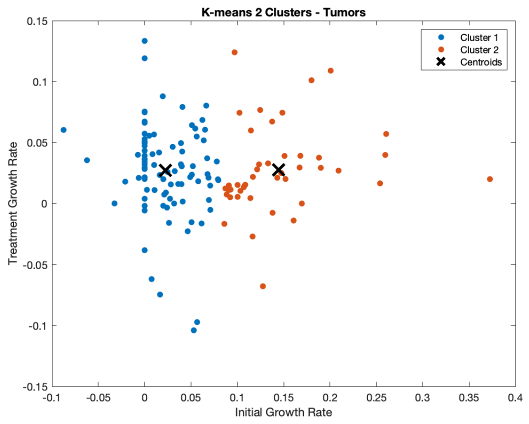


Figure 36: Best $k = 2$ method for Tumor Data: K means Method

For 3 cluster sets ($k = 3$) on Tumor Data the best cluster method was the Gaussian Mixture Models Method with a full and shared covariance matrix:

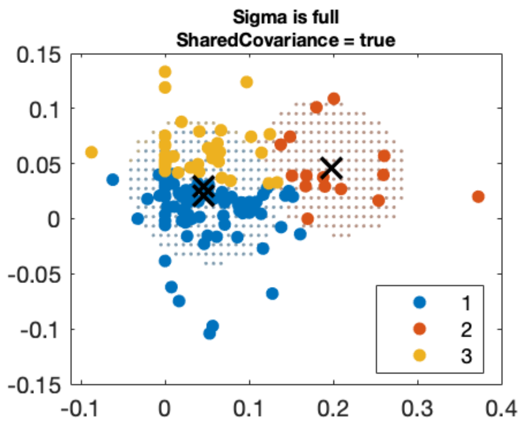


Figure 37: Best $k = 3$ method for Tumor Data: GMM Method Full-True

For 4 cluster sets ($k = 4$) on Tumor Data the best cluster method was the Gaussian Mixture Models Method with a full and shared covariance matrix:

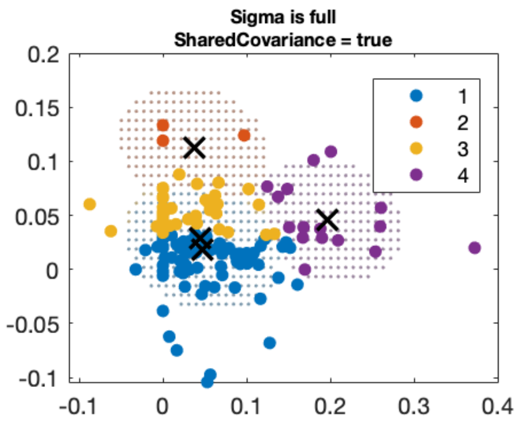


Figure 38: Best $k = 4$ method for Tumor Data: GMM Method Full-True

For 5 cluster sets ($k = 5$) on Tumor Data the best cluster method was the Gaussian Mixture Models Method with a full and shared covariance matrix:

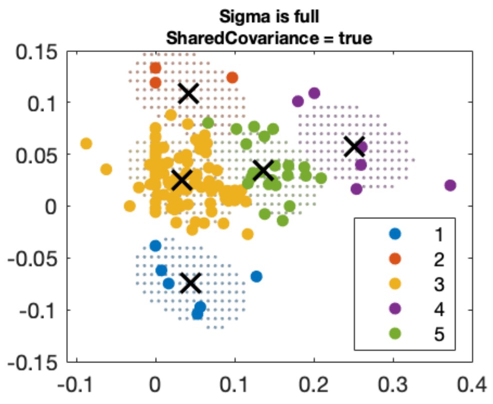


Figure 39: Best $k = 5$ method for Tumor Data: GMM Method Full-True

5.2 Comparing Mice Findings

For 2 cluster sets ($k = 2$) on Mice Data the best cluster method was the K means method:

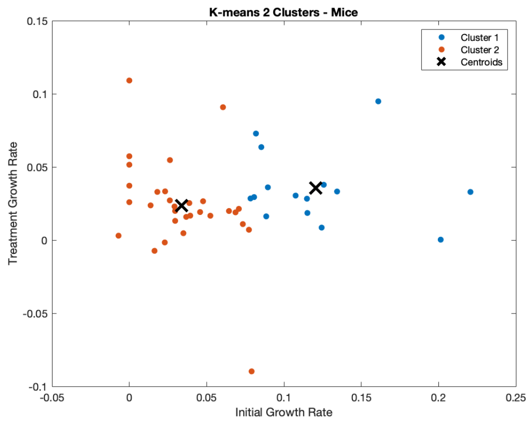


Figure 40: Best $k = 2$ method for Mice Data: K means Method

For 3 cluster sets ($k = 3$) on Mice Data the best cluster method was the Ward Linkage method:

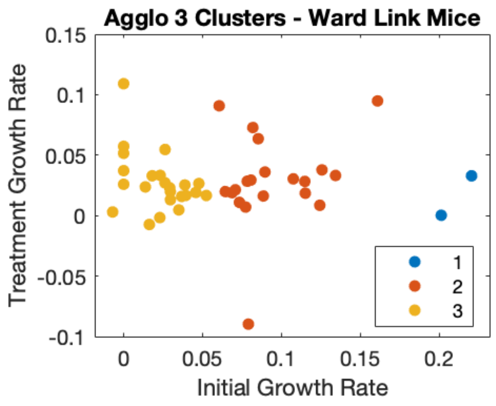


Figure 41: Best $k = 3$ method for Mice Data: Hierarchical Agglo Method Ward Linkage

For 4 cluster sets ($k = 4$) on Mice Data the best cluster method was the K means method:

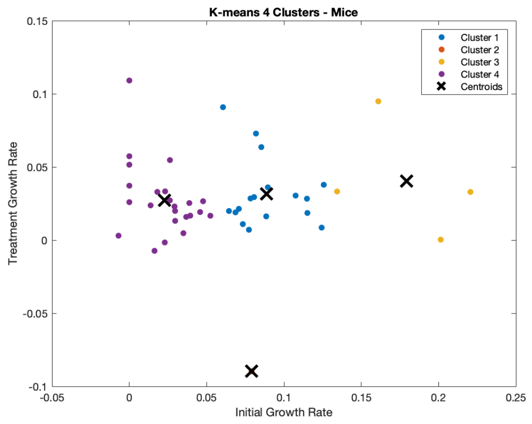


Figure 42: Best $k = 4$ method for Mice Data: K means Method

For 5 cluster sets ($k = 5$) on Mice Data the best cluster method was the Complete Linkage method:

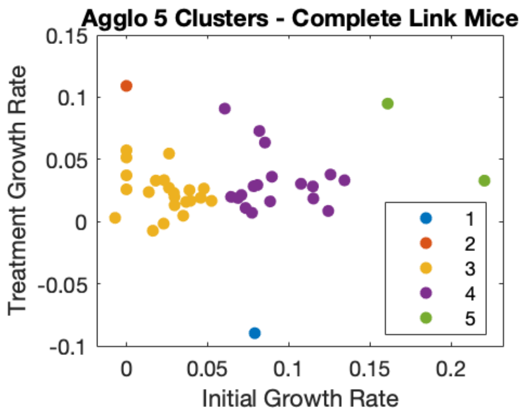


Figure 43: Best $k = 5$ method for Mice Data: Hierarchical Agglo Method Complete Linkage

5.3 Final Thoughts and Future Research Possibilities

From the previous two sections we see the best groupings of the tumor data and the best groupings of the mice data. Through this experiment it is viable to conclude that more data creates clearer and stronger clusters. I believe it was important to show both viewings of the data, tumor and mice, as it gave an example of how more information on your data means better clustering patterns start to form. The data for the mice may be smaller than one would like, but the great attribute of having three or more tumors on a mouse allows for a reasonable expansion of the data and helps the Unsupervised Learning methods run smoother. I find the comparison of these two data sets to be well motivated and helpful in understanding the methods of clustering.

This project focused mainly on the creation of the Unsupervised Learning cluster methods. Understanding these processes and implementing them into code was a great goal and I believed help me better understand the process of machine learning. Through this experience I have found that there isn't one main method that works for all scenarios; instead it is important to assess your problem and understand what you are trying to do with the material. Once you have a good understanding of these things, then the next step is to find a method that excels at what you're interested in and that works for your specific data. There is no free lunch!

There are many avenues one can go down for future research on this problem. The main and immediate extension of this project is to compare these clusters to that of the five different mice groups and see how they correlate. After finding out whether or not the above correlation is true, one should look into the *other* features of the mice to look for other correlations that could explain the specific cluster's growth rate relationship. Another research possibility that a person could do that's more related to the machine learning side is to test the processing speed of each of these methods. If one does this they can then determine which method would be best to use on a very large data set so that you may be able to do research efficiently in the future. Another possible machine learning research avenue is to investigate how all of these methods perform by running training sets and test sets of the data to validate how well each clustering method performs. Lastly, one could come up with a metric to analytically and quantitatively compare the clustering algorithms by inducing an error through distances to the average cluster point or another point to compute distances.

All in all this project has been very interesting, thought provoking, and enjoyable. I'm happy that I took this course and I hope to be able to do research in the future with either of you, thank you Ben and Ziyu!

A Appendix

Note that all code below was done specifically for this project. The code here uses many more codes, but they were not made for/ directly related to this project so they are not listed here.
Note that these codes use saved .mat files; if you wish to replicate or run these files the .mat files are necessary. Please inquire about them if needed/wanted.

A.1 Growth Rate Code

```
    clear vars; close all; format long;
% Math 590 Project: Unsupervised Learning on Cancer Data

% Steps: - Import data of mice
%        - Determine growth rate of tumors
%           - Before Treatment
%           - After Treatment
%        - Unsupervised Learning Cluster Technique 1: GMM EM Algorithm
%           - Investigate other features inside each cluster
%        - Unsupervised Learning Cluster Technique 2: Agglomerative Clusters
%           - Investigate other features inside each cluster
%        - Unsupervised Learning Cluster Technique 3: K-means, K-means++
%           - Investigate other features inside each cluster


%% Calculating Growth rate of tumors
% Will Create variable for list of volumes
% Create a variable for days of injection for each mouse

%% Mice from CM3
% Volumes:
load('CM37Vols.mat')
load('CM38Vols.mat')
load('CM39Vols.mat')
load('CM40Vols.mat')
load('CM41Vols.mat')
load('CM42Vols.mat')
load('CM43Vols.mat')
load('CM46Vols.mat')
load('CM47Vols.mat')
load('CM48Vols.mat')
load('CM49Vols.mat')
% Dates:
load('CM37Time.mat')
load('CM38Time.mat')
load('CM39Time.mat')
load('CM40Time.mat')
load('CM41Time.mat')
load('CM42Time.mat')
load('CM43Time.mat')
load('CM46Time.mat')
load('CM47Time.mat')
load('CM48Time.mat')
load('CM49Time.mat')

% Making Big CM3 Volume Matrix
% Note CM47 and CM49 have 4 instead of 3
[CM3_Vols, tf] = ...
padcat(CM37_vol_tumors(1,:), CM37_vol_tumors(2,:), CM37_vol_tumors(3,:), ...
CM38_vol_tumors(1,:), CM38_vol_tumors(2,:), CM38_vol_tumors(3,:), ...
CM39_vol_tumors(1,:), CM39_vol_tumors(2,:), CM39_vol_tumors(3,:), ...
CM40_vol_tumors(1,:), CM40_vol_tumors(2,:), CM40_vol_tumors(3,:), ...
CM41_vol_tumors(1,:), CM41_vol_tumors(2,:), CM41_vol_tumors(3,:), ...
CM42_vol_tumors(1,:), CM42_vol_tumors(2,:), CM42_vol_tumors(3,:), ...
CM43_vol_tumors(1,:), CM43_vol_tumors(2,:), CM43_vol_tumors(3,:), ...
CM46_vol_tumors(1,:), CM46_vol_tumors(2,:), CM46_vol_tumors(3,:), ...
CM47_vol_tumors(1,:), CM47_vol_tumors(2,:), CM47_vol_tumors(3,:), ...
CM47_vol_tumors(4,:),...
CM48_vol_tumors(1,:), CM48_vol_tumors(2,:), CM48_vol_tumors(3,:), ...
CM49_vol_tumors(1,:), CM49_vol_tumors(2,:), CM49_vol_tumors(3,:), ...
CM49_vol_tumors(4,:));

CM3_Vols(~tf) = 0;

% Making Big CM3 Time Matrix
% Note CM47 and CM49 have 4 instead of 3
[CM3_Times, tg] = padcat(CM37_time, CM37_time, CM37_time, ...
                         CM38_time, CM38_time, CM38_time, ...
                         CM39_time, CM39_time, CM39_time, ...
                         CM40_time, CM40_time, CM40_time, ...
                         CM41_time, CM41_time, CM41_time, ...
                         CM42_time, CM42_time, CM42_time, ...
                         CM43_time, CM43_time, CM43_time, ...
                         CM46_time, CM46_time, CM46_time, ...
                         CM47_time, CM47_time, CM47_time, CM47_time, ...
                         CM48_time, CM48_time, CM48_time, ...
                         CM49_time, CM49_time, CM49_time, CM49_time);
CM3_Times(~tg) = 0;

% This will be what indices the injection day is closest to,
% note that each mouse has the same injection day for each tumor
% Note CM47 and CM49 have 4 instead of 3
CM3_injection_dates = [5, 5, 5, ...            % CM37
                       7, 7, 7, ...            % CM38
                       7, 7, 7, ...            % CM39
                       5, 5, 5, ...            % CM40
                       8, 8, 8, ...            % CM41
                       9, 9, 9, ...            % CM42
                       31, 31, 31, ...         % CM43
                       3, 3, 3, ...            % CM46
                       7, 7, 7, 7, ...         % CM47
                       7, 7, 7, ...            % CM48
                       9, 9, 9, 9];            % CM49

CM3_Initial_growth_rates = init_grow(CM3_Vols, CM3_Times, CM3_injection_dates);
CM3_Treatment_growth_rates = treat_grow(CM3_Vols, CM3_Times, CM3_injection_dates);

% Want to find average initial and treatment growth rate for each mouse

% Initial
CM3_i_rate = CM3_Initial_growth_rates; % for cleanliness
CM37_avg_init_rate = (CM3_i_rate(1) + CM3_i_rate(2) + CM3_i_rate(3))/3;
CM38_avg_init_rate = (CM3_i_rate(4) + CM3_i_rate(5) + CM3_i_rate(6))/3;
CM39_avg_init_rate = (CM3_i_rate(7) + CM3_i_rate(8) + CM3_i_rate(9))/3;
CM40_avg_init_rate = (CM3_i_rate(10) + CM3_i_rate(11) + CM3_i_rate(12))/3;
CM41_avg_init_rate = (CM3_i_rate(13) + CM3_i_rate(14) + CM3_i_rate(15))/3;
CM42_avg_init_rate = (CM3_i_rate(16) + CM3_i_rate(17) + CM3_i_rate(18))/3;
CM43_avg_init_rate = (CM3_i_rate(19) + CM3_i_rate(20) + CM3_i_rate(21))/3;
CM46_avg_init_rate = (CM3_i_rate(22) + CM3_i_rate(23) + CM3_i_rate(24))/3;
CM47_avg_init_rate = (CM3_i_rate(25) + CM3_i_rate(26) + CM3_i_rate(27) ...
    + CM3_i_rate(28))/4;
CM48_avg_init_rate = (CM3_i_rate(29) + CM3_i_rate(30) + CM3_i_rate(31))/3;
CM49_avg_init_rate = (CM3_i_rate(32) + CM3_i_rate(33) + CM3_i_rate(34) ...
    + CM3_i_rate(35))/4;
```

```
CM3_Average_initial_growth_rates = ...
    [CM37_avg_init_rate, CM38_avg_init_rate, CM39_avg_init_rate, ...
    CM40_avg_init_rate, CM41_avg_init_rate, CM42_avg_init_rate, ...
    CM43_avg_init_rate, CM46_avg_init_rate, CM47_avg_init_rate, ...
    CM48_avg_init_rate, CM49_avg_init_rate];

% Treatment
CM3_t_rate = CM3_Treatment_growth_rates;% for cleanliness

CM37_avg_treat_rate = (CM3_t_rate(1) + CM3_t_rate(2) + CM3_t_rate(3))/3;
CM38_avg_treat_rate = (CM3_t_rate(4) + CM3_t_rate(5) + CM3_t_rate(6))/3;
CM39_avg_treat_rate = (CM3_t_rate(7) + CM3_t_rate(8) + CM3_t_rate(9))/3;
CM40_avg_treat_rate = (CM3_t_rate(10) + CM3_t_rate(11) + CM3_t_rate(12))/3;
CM41_avg_treat_rate = (CM3_t_rate(13) + CM3_t_rate(14) + CM3_t_rate(15))/3;
CM42_avg_treat_rate = (CM3_t_rate(16) + CM3_t_rate(17) + CM3_t_rate(18))/3;
CM43_avg_treat_rate = (CM3_t_rate(19) + CM3_t_rate(20) + CM3_t_rate(21))/3;
CM46_avg_treat_rate = (CM3_t_rate(22) + CM3_t_rate(23) + CM3_t_rate(24))/3;
CM47_avg_treat_rate = (CM3_t_rate(25) + CM3_t_rate(26) + CM3_t_rate(27) ...
    + CM3_t_rate(28))/4;
CM48_avg_treat_rate = (CM3_t_rate(29) + CM3_t_rate(30) + CM3_t_rate(31))/3;
CM49_avg_treat_rate = (CM3_t_rate(32) + CM3_t_rate(33) + CM3_t_rate(34) ...
    + CM3_t_rate(35))/4;

CM3_Average_treatment_growth_rates = ...
    [CM37_avg_treat_rate, CM38_avg_treat_rate, CM39_avg_treat_rate, ...
    CM40_avg_treat_rate, CM41_avg_treat_rate, CM42_avg_treat_rate, ...
    CM43_avg_treat_rate, CM46_avg_treat_rate, CM47_avg_treat_rate, ...
    CM48_avg_treat_rate, CM49_avg_treat_rate];

% It works! :)



%% Mice from CM4
% Volumes:
load('CM51Vols.mat')
load('CM52Vols.mat')
load('CM53Vols.mat')
load('CM54Vols.mat')
load('CM55Vols.mat')
load('CM56Vols.mat')
load('CM57Vols.mat')
load('CM58Vols.mat')
load('CM59Vols.mat')
% Dates:
load('CM51Time.mat')
load('CM52Time.mat')
load('CM53Time.mat')
load('CM54Time.mat')
load('CM55Time.mat')
load('CM56Time.mat')
load('CM57Time.mat')
load('CM58Time.mat')
load('CM59Time.mat')

% Making Big CM4 Volume Matrix

[CM4_Vols, tf] = ...
padcat(CM51_vol_tumors(1,:), CM51_vol_tumors(2,:), CM51_vol_tumors(3,:), ...
    CM52_vol_tumors(1,:), CM52_vol_tumors(2,:), CM52_vol_tumors(3,:), ...
    CM53_vol_tumors(1,:), CM53_vol_tumors(2,:), CM53_vol_tumors(3,:), ...
    CM54_vol_tumors(1,:), CM54_vol_tumors(2,:), CM54_vol_tumors(3,:), ...
    CM55_vol_tumors(1,:), CM55_vol_tumors(2,:), CM55_vol_tumors(3,:), ...
    CM56_vol_tumors(1,:), CM56_vol_tumors(2,:), CM56_vol_tumors(3,:), ...
    CM57_vol_tumors(1,:), CM57_vol_tumors(2,:), CM57_vol_tumors(3,:), ...
    CM58_vol_tumors(1,:), CM58_vol_tumors(2,:), CM58_vol_tumors(3,:), ...
    CM59_vol_tumors(1,:), CM59_vol_tumors(2,:), CM59_vol_tumors(3,:));
CM4_Vols(~tf) = 0;

% Making Big CM4 Time Matrix

[CM4_Times, tg] = padcat(CM51_time, CM51_time, CM51_time, ...
                         CM52_time, CM52_time, CM52_time, ...
                         CM53_time, CM53_time, CM53_time, ...
                         CM54_time, CM54_time, CM54_time, ...
                         CM55_time, CM55_time, CM55_time, ...
                         CM56_time, CM56_time, CM56_time, ...
                         CM57_time, CM57_time, CM57_time, ...
                         CM58_time, CM58_time, CM58_time, ...
                         CM59_time, CM59_time, CM59_time);
CM4_Times(~tg) = 0;

% This will be what indices the injection day is closest to,
% note that each mouse has the same injection day for each tumor
```

```
CM4_injection_dates = [9, 9, 9, ...            % CM51
                       11, 11, 11, ...         % CM52
                       5, 5, 5, ...            % CM53
                       12, 12, 12, ...         % CM54
                       6, 6, 6, ...            % CM55
                       6, 6, 6, ...            % CM56
                       6, 6, 6, ...            % CM57
                       6, 6, 6, ...            % CM58
                       7, 7, 7];               % CM59

CM4_Initial_growth_rates = init_grow(CM4_Vols, CM4_Times, CM4_injection_dates);
CM4_Treatment_growth_rates = treat_grow(CM4_Vols, CM4_Times, CM4_injection_dates);

% Want to find average initial and treatment growth rate for each mouse

% Initial
CM4_i_rate = CM4_Initial_growth_rates; % for cleanliness
CM51_avg_init_rate = (CM4_i_rate(1) + CM4_i_rate(2) + CM4_i_rate(3))/3;
CM52_avg_init_rate = (CM4_i_rate(4) + CM4_i_rate(5) + CM4_i_rate(6))/3;
CM53_avg_init_rate = (CM4_i_rate(7) + CM4_i_rate(8) + CM4_i_rate(9))/3;
CM54_avg_init_rate = (CM4_i_rate(10) + CM4_i_rate(11) + CM4_i_rate(12))/3;
CM55_avg_init_rate = (CM4_i_rate(13) + CM4_i_rate(14) + CM4_i_rate(15))/3;
CM56_avg_init_rate = (CM4_i_rate(16) + CM4_i_rate(17) + CM4_i_rate(18))/3;
CM57_avg_init_rate = (CM4_i_rate(19) + CM4_i_rate(20) + CM4_i_rate(21))/3;
CM58_avg_init_rate = (CM4_i_rate(22) + CM4_i_rate(23) + CM4_i_rate(24))/3;
CM59_avg_init_rate = (CM4_i_rate(25) + CM4_i_rate(26) + CM4_i_rate(27))/3;


CM4_Average_initial_growth_rates = ...
    [CM51_avg_init_rate, CM52_avg_init_rate, CM53_avg_init_rate, ...
    CM54_avg_init_rate, CM55_avg_init_rate, CM56_avg_init_rate, ...
    CM57_avg_init_rate, CM58_avg_init_rate, CM59_avg_init_rate];

% Treatment
CM4_t_rate = CM4_Treatment_growth_rates;% for cleanliness

CM51_avg_treat_rate = (CM4_t_rate(1) + CM4_t_rate(2) + CM4_t_rate(3))/3;
CM52_avg_treat_rate = (CM4_t_rate(4) + CM4_t_rate(5) + CM4_t_rate(6))/3;
CM53_avg_treat_rate = (CM4_t_rate(7) + CM4_t_rate(8) + CM4_t_rate(9))/3;
CM54_avg_treat_rate = (CM4_t_rate(10) + CM4_t_rate(11) + CM4_t_rate(12))/3;
CM55_avg_treat_rate = (CM4_t_rate(13) + CM4_t_rate(14) + CM4_t_rate(15))/3;
CM56_avg_treat_rate = (CM4_t_rate(16) + CM4_t_rate(17) + CM4_t_rate(18))/3;
```

```
CM57_avg_treat_rate = (CM4_t_rate(19) + CM4_t_rate(20) + CM4_t_rate(21))/3;
CM58_avg_treat_rate = (CM4_t_rate(22) + CM4_t_rate(23) + CM4_t_rate(24))/3;
CM59_avg_treat_rate = (CM4_t_rate(25) + CM4_t_rate(26) + CM4_t_rate(27))/3;

CM4_Average_treatment_growth_rates = ...
    [CM51_avg_treat_rate, CM52_avg_treat_rate, CM53_avg_treat_rate, ...
    CM54_avg_treat_rate, CM55_avg_treat_rate, CM56_avg_treat_rate, ...
    CM57_avg_treat_rate, CM58_avg_treat_rate, CM59_avg_treat_rate];

% Worked! :)



%% Mice from CM5
% Volumes:
load('CM60Vols.mat')
load('CM62Vols.mat')
load('CM63Vols.mat')
load('CM66Vols.mat')
load('CM67Vols.mat')
load('CM68Vols.mat')
load('CM69Vols.mat')
load('CM70Vols.mat')
load('CM71Vols.mat')
% Dates:
load('CM60Time.mat')
load('CM62Time.mat')
load('CM63Time.mat')
load('CM66Time.mat')
load('CM67Time.mat')
load('CM68Time.mat')
load('CM69Time.mat')
load('CM70Time.mat')
load('CM71Time.mat')

% Making Big CM5 Volume Matrix

[CM5_Vols, tf] = ...
padcat(CM60_vol_tumors(1,:), CM60_vol_tumors(2,:), CM60_vol_tumors(3,:), ...
    CM62_vol_tumors(1,:), CM62_vol_tumors(2,:), CM62_vol_tumors(3,:), ...
    CM63_vol_tumors(1,:), CM63_vol_tumors(2,:), CM63_vol_tumors(3,:), ...
    CM66_vol_tumors(1,:), CM66_vol_tumors(2,:), CM66_vol_tumors(3,:), ...
    CM67_vol_tumors(1,:), CM67_vol_tumors(2,:), CM67_vol_tumors(3,:), ...
    CM68_vol_tumors(1,:), CM68_vol_tumors(2,:), CM68_vol_tumors(3,:), ...
    CM69_vol_tumors(1,:), CM69_vol_tumors(2,:), CM69_vol_tumors(3,:), ...
    CM70_vol_tumors(1,:), CM70_vol_tumors(2,:), CM70_vol_tumors(3,:), ...
    CM71_vol_tumors(1,:), CM71_vol_tumors(2,:), CM71_vol_tumors(3,:));
CM5_Vols(~tf) = 0;

% Making Big CM5 Time Matrix

[CM5_Times, tg] = padcat(CM60_time, CM60_time, CM60_time, ...
                         CM62_time, CM62_time, CM62_time, ...
                         CM63_time, CM63_time, CM63_time, ...
                         CM66_time, CM66_time, CM66_time, ...
                         CM67_time, CM67_time, CM67_time, ...
                         CM68_time, CM68_time, CM68_time, ...
                         CM69_time, CM69_time, CM69_time, ...
                         CM70_time, CM70_time, CM70_time, ...
                         CM71_time, CM71_time, CM71_time);
CM5_Times(~tg) = 0;

% This will be what indices the injection day is closest to,
% note that each mouse has the same injection day for each tumor
CM5_injection_dates = [1, 1, 1, ...            % CM60
                       5, 5, 5, ...            % CM62
                       7, 7, 7, ...            % CM63
                       3, 3, 3, ...            % CM66
                       6, 6, 6, ...            % CM67
                       2, 2, 2, ...            % CM68
                       4, 4, 4, ...            % CM69
                       8, 8, 8, ...            % CM70
                       5, 5, 5];               % CM71

CM5_Initial_growth_rates = init_grow(CM5_Vols, CM5_Times, CM5_injection_dates);
CM5_Treatment_growth_rates = treat_grow(CM5_Vols, CM5_Times, CM5_injection_dates);

% Want to find average initial and treatment growth rate for each mouse

% Initial
CM5_i_rate = CM5_Initial_growth_rates; % for cleanliness
CM60_avg_init_rate = (CM5_i_rate(1) + CM5_i_rate(2) + CM5_i_rate(3))/3;
CM62_avg_init_rate = (CM5_i_rate(4) + CM5_i_rate(5) + CM5_i_rate(6))/3;
CM63_avg_init_rate = (CM5_i_rate(7) + CM5_i_rate(8) + CM5_i_rate(9))/3;
```

```
CM66_avg_init_rate = (CM5_i_rate(10) + CM5_i_rate(11) + CM5_i_rate(12))/3;
CM67_avg_init_rate = (CM5_i_rate(13) + CM5_i_rate(14) + CM5_i_rate(15))/3;
CM68_avg_init_rate = (CM5_i_rate(16) + CM5_i_rate(17) + CM5_i_rate(18))/3;
CM69_avg_init_rate = (CM5_i_rate(19) + CM5_i_rate(20) + CM5_i_rate(21))/3;
CM70_avg_init_rate = (CM5_i_rate(22) + CM5_i_rate(23) + CM5_i_rate(24))/3;
CM71_avg_init_rate = (CM5_i_rate(25) + CM5_i_rate(26) + CM5_i_rate(27))/3;


CM5_Average_initial_growth_rates = ...
    [CM60_avg_init_rate, CM62_avg_init_rate, CM63_avg_init_rate, ...
    CM66_avg_init_rate, CM67_avg_init_rate, CM68_avg_init_rate, ...
    CM69_avg_init_rate, CM70_avg_init_rate, CM71_avg_init_rate];

% Treatment
CM5_t_rate = CM5_Treatment_growth_rates;% for cleanliness

CM60_avg_treat_rate = (CM5_t_rate(1) + CM5_t_rate(2) + CM5_t_rate(3))/3;
CM62_avg_treat_rate = (CM5_t_rate(4) + CM5_t_rate(5) + CM5_t_rate(6))/3;
CM63_avg_treat_rate = (CM5_t_rate(7) + CM5_t_rate(8) + CM5_t_rate(9))/3;
CM66_avg_treat_rate = (CM5_t_rate(10) + CM5_t_rate(11) + CM5_t_rate(12))/3;
CM67_avg_treat_rate = (CM5_t_rate(13) + CM5_t_rate(14) + CM5_t_rate(15))/3;
CM68_avg_treat_rate = (CM5_t_rate(16) + CM5_t_rate(17) + CM5_t_rate(18))/3;
CM69_avg_treat_rate = (CM5_t_rate(19) + CM5_t_rate(20) + CM5_t_rate(21))/3;
CM70_avg_treat_rate = (CM5_t_rate(22) + CM5_t_rate(23) + CM5_t_rate(24))/3;
CM71_avg_treat_rate = (CM5_t_rate(25) + CM5_t_rate(26) + CM5_t_rate(27))/3;

CM5_Average_treatment_growth_rates = ...
    [CM60_avg_treat_rate, CM62_avg_treat_rate, CM63_avg_treat_rate, ...
    CM66_avg_treat_rate, CM67_avg_treat_rate, CM68_avg_treat_rate, ...
    CM69_avg_treat_rate, CM70_avg_treat_rate, CM71_avg_treat_rate];

% Yuppp :)



%% Mice from CM6
% Volumes:
load('CM72Vols.mat')
load('CM73Vols.mat')
load('CM74Vols.mat')
load('CM75Vols.mat')
load('CM76Vols.mat')
load('CM77Vols.mat')
load('CM78Vols.mat')
load('CM79Vols.mat')
% Dates:
load('CM72Time.mat')
load('CM73Time.mat')
load('CM74Time.mat')
load('CM75Time.mat')
load('CM76Time.mat')
load('CM77Time.mat')
load('CM78Time.mat')
load('CM79Time.mat')

% Making Big CM6 Volume Matrix

[CM6_Vols, tf] = ...
padcat(CM72_vol_tumors(1,:), CM72_vol_tumors(2,:), CM72_vol_tumors(3,:), ...
    CM73_vol_tumors(1,:), CM73_vol_tumors(2,:), CM73_vol_tumors(3,:), ...
    CM74_vol_tumors(1,:), CM74_vol_tumors(2,:), CM74_vol_tumors(3,:), ...
    CM75_vol_tumors(1,:), CM75_vol_tumors(2,:), CM75_vol_tumors(3,:), ...
    CM76_vol_tumors(1,:), CM76_vol_tumors(2,:), CM76_vol_tumors(3,:), ...
    CM77_vol_tumors(1,:), CM77_vol_tumors(2,:), CM77_vol_tumors(3,:), ...
    CM78_vol_tumors(1,:), CM78_vol_tumors(2,:), CM78_vol_tumors(3,:), ...
    CM79_vol_tumors(1,:), CM79_vol_tumors(2,:), CM79_vol_tumors(3,:));
CM6_Vols(~tf) = 0;

% Making Big CM6 Time Matrix

[CM6_Times, tg] = padcat(CM72_time, CM72_time, CM72_time, ...
                         CM73_time, CM73_time, CM73_time, ...
                         CM74_time, CM74_time, CM74_time, ...
                         CM75_time, CM75_time, CM75_time, ...
                         CM76_time, CM76_time, CM76_time, ...
                         CM77_time, CM77_time, CM77_time, ...
                         CM78_time, CM78_time, CM78_time, ...
                         CM79_time, CM79_time, CM79_time);
CM6_Times(~tg) = 0;

% This will be what indices the injection day is closest to,
% note that each mouse has the same injection day for each tumor
CM6_injection_dates = [1, 1, 1, ...             % CM72
                       5, 5, 5, ...             % CM73
```

```
                        1, 1, 1, ...             % CM74
                        5, 5, 5, ...             % CM75
                        5, 5, 5, ...             % CM76
                        4, 4, 4, ...             % CM77
                        1, 1, 1, ...             % CM78
                        7, 7, 7];                % CM79

CM6_Initial_growth_rates = init_grow(CM6_Vols, CM6_Times, CM6_injection_dates);
CM6_Treatment_growth_rates = treat_grow(CM6_Vols, CM6_Times, CM6_injection_dates);

% Want to find average initial and treatment growth rate for each mouse

% Initial
CM6_i_rate = CM6_Initial_growth_rates; % for cleanliness
CM72_avg_init_rate = (CM6_i_rate(1) + CM6_i_rate(2) + CM6_i_rate(3))/3;
CM73_avg_init_rate = (CM6_i_rate(4) + CM6_i_rate(5) + CM6_i_rate(6))/3;
CM74_avg_init_rate = (CM6_i_rate(7) + CM6_i_rate(8) + CM6_i_rate(9))/3;
CM75_avg_init_rate = (CM6_i_rate(10) + CM6_i_rate(11) + CM6_i_rate(12))/3;
CM76_avg_init_rate = (CM6_i_rate(13) + CM6_i_rate(14) + CM6_i_rate(15))/3;
CM77_avg_init_rate = (CM6_i_rate(16) + CM6_i_rate(17) + CM6_i_rate(18))/3;
CM78_avg_init_rate = (CM6_i_rate(19) + CM6_i_rate(20) + CM6_i_rate(21))/3;
CM79_avg_init_rate = (CM6_i_rate(22) + CM6_i_rate(23) + CM6_i_rate(24))/3;


CM6_Average_initial_growth_rates = ...
    [CM72_avg_init_rate, CM73_avg_init_rate, CM74_avg_init_rate, ...
    CM75_avg_init_rate, CM76_avg_init_rate, CM77_avg_init_rate, ...
    CM78_avg_init_rate, CM79_avg_init_rate];

% Treatment
CM6_t_rate = CM6_Treatment_growth_rates;% for cleanliness

CM72_avg_treat_rate = (CM6_t_rate(1) + CM6_t_rate(2) + CM6_t_rate(3))/3;
CM73_avg_treat_rate = (CM6_t_rate(4) + CM6_t_rate(5) + CM6_t_rate(6))/3;
CM74_avg_treat_rate = (CM6_t_rate(7) + CM6_t_rate(8) + CM6_t_rate(9))/3;
CM75_avg_treat_rate = (CM6_t_rate(10) + CM6_t_rate(11) + CM6_t_rate(12))/3;
CM76_avg_treat_rate = (CM6_t_rate(13) + CM6_t_rate(14) + CM6_t_rate(15))/3;
CM77_avg_treat_rate = (CM6_t_rate(16) + CM6_t_rate(17) + CM6_t_rate(18))/3;
CM78_avg_treat_rate = (CM6_t_rate(19) + CM6_t_rate(20) + CM6_t_rate(21))/3;
CM79_avg_treat_rate = (CM6_t_rate(22) + CM6_t_rate(23) + CM6_t_rate(24))/3;

CM6_Average_treatment_growth_rates = ...
```

```
    [CM72_avg_treat_rate, CM73_avg_treat_rate, CM74_avg_treat_rate, ...
    CM75_avg_treat_rate, CM76_avg_treat_rate, CM77_avg_treat_rate, ...
    CM78_avg_treat_rate, CM79_avg_treat_rate];

% Almost there! :)



%% Mice from CM7
% Volumes:
load('CM83Vols.mat')
load('CM84Vols.mat')
load('CM85Vols.mat')
load('CM86Vols.mat')
load('CM90Vols.mat')
load('CM91Vols.mat')
load('CM92Vols.mat')
load('CM93Vols.mat')
% Dates:
load('CM83Time.mat')
load('CM84Time.mat')
load('CM85Time.mat')
load('CM86Time.mat')
load('CM90Time.mat')
load('CM91Time.mat')
load('CM92Time.mat')
load('CM93Time.mat')

% Making Big CM6 Volume Matrix

[CM7_Vols, tf] = ...
padcat(CM83_vol_tumors(1,:), CM83_vol_tumors(2,:), CM83_vol_tumors(3,:), ...
    CM84_vol_tumors(1,:), CM84_vol_tumors(2,:), CM84_vol_tumors(3,:), ...
    CM85_vol_tumors(1,:), CM85_vol_tumors(2,:), CM85_vol_tumors(3,:), ...
    CM86_vol_tumors(1,:), CM86_vol_tumors(2,:), CM86_vol_tumors(3,:), ...
    CM90_vol_tumors(1,:), CM90_vol_tumors(2,:), CM90_vol_tumors(3,:), ...
    CM91_vol_tumors(1,:), CM91_vol_tumors(2,:), CM91_vol_tumors(3,:), ...
    CM92_vol_tumors(1,:), CM92_vol_tumors(2,:), CM92_vol_tumors(3,:), ...
    CM93_vol_tumors(1,:), CM93_vol_tumors(2,:), CM93_vol_tumors(3,:));
CM7_Vols(~tf) = 0;

% Making Big CM6 Time Matrix
```

```
[CM7_Times, tg] = padcat(CM83_time, CM83_time, CM83_time, ...
                        CM84_time, CM84_time, CM84_time, ...
                        CM85_time, CM85_time, CM85_time, ...
                        CM86_time, CM86_time, CM86_time, ...
                        CM90_time, CM90_time, CM90_time, ...
                        CM91_time, CM91_time, CM91_time, ...
                        CM92_time, CM92_time, CM92_time, ...
                        CM93_time, CM93_time, CM93_time);
CM7_Times(~tg) = 0;

% This will be what indices the injection day is closest to,
% note that each mouse has the same injection day for each tumor
CM7_injection_dates = [1, 1, 1, ...             % CM83
                      11, 11, 11, ...          % CM84
                      4, 4, 4, ...             % CM85
                      6, 6, 6, ...             % CM86
                      3, 3, 3, ...             % CM90
                      5, 5, 5, ...             % CM91
                      7, 7, 7, ...             % CM92
                      3, 3, 3];                % CM93

CM7_Initial_growth_rates = init_grow(CM7_Vols, CM7_Times, CM7_injection_dates);
CM7_Treatment_growth_rates = treat_grow(CM7_Vols, CM7_Times, CM7_injection_dates);

% Want to find average initial and treatment growth rate for each mouse

% Initial
CM7_i_rate = CM7_Initial_growth_rates; % for cleanliness
CM83_avg_init_rate = (CM7_i_rate(1) + CM7_i_rate(2) + CM7_i_rate(3))/3;
CM84_avg_init_rate = (CM7_i_rate(4) + CM7_i_rate(5) + CM7_i_rate(6))/3;
CM85_avg_init_rate = (CM7_i_rate(7) + CM7_i_rate(8) + CM7_i_rate(9))/3;
CM86_avg_init_rate = (CM7_i_rate(10) + CM7_i_rate(11) + CM7_i_rate(12))/3;
CM90_avg_init_rate = (CM7_i_rate(13) + CM7_i_rate(14) + CM7_i_rate(15))/3;
CM91_avg_init_rate = (CM7_i_rate(16) + CM7_i_rate(17) + CM7_i_rate(18))/3;
CM92_avg_init_rate = (CM7_i_rate(19) + CM7_i_rate(20) + CM7_i_rate(21))/3;
CM93_avg_init_rate = (CM7_i_rate(22) + CM7_i_rate(23) + CM7_i_rate(24))/3;


CM7_Average_initial_growth_rates = ...
    [CM83_avg_init_rate, CM84_avg_init_rate, CM85_avg_init_rate, ...
    CM86_avg_init_rate, CM90_avg_init_rate, CM91_avg_init_rate, ...
    CM92_avg_init_rate, CM93_avg_init_rate];

% Treatment
CM7_t_rate = CM7_Treatment_growth_rates;% for cleanliness

CM83_avg_treat_rate = (CM7_t_rate(1) + CM7_t_rate(2) + CM7_t_rate(3))/3;
CM84_avg_treat_rate = (CM7_t_rate(4) + CM7_t_rate(5) + CM7_t_rate(6))/3;
CM85_avg_treat_rate = (CM7_t_rate(7) + CM7_t_rate(8) + CM7_t_rate(9))/3;
CM86_avg_treat_rate = (CM7_t_rate(10) + CM7_t_rate(11) + CM7_t_rate(12))/3;
CM90_avg_treat_rate = (CM7_t_rate(13) + CM7_t_rate(14) + CM7_t_rate(15))/3;
CM91_avg_treat_rate = (CM7_t_rate(16) + CM7_t_rate(17) + CM7_t_rate(18))/3;
CM92_avg_treat_rate = (CM7_t_rate(19) + CM7_t_rate(20) + CM7_t_rate(21))/3;
CM93_avg_treat_rate = (CM7_t_rate(22) + CM7_t_rate(23) + CM7_t_rate(24))/3;

CM7_Average_treatment_growth_rates = ...
    [CM83_avg_treat_rate, CM84_avg_treat_rate, CM85_avg_treat_rate, ...
    CM86_avg_treat_rate, CM90_avg_treat_rate, CM91_avg_treat_rate, ...
    CM92_avg_treat_rate, CM93_avg_treat_rate];

% IT HAS BEEN DONE! :)))


%% Making a Complete List of all Growth Rates

Complete_Initial_Rates = [CM3_Initial_growth_rates, ...
                          CM4_Initial_growth_rates, ...
                          CM5_Initial_growth_rates, ...
                          CM6_Initial_growth_rates, ...
                          CM7_Initial_growth_rates];

Complete_Treatment_Rates = [CM3_Treatment_growth_rates, ...
                            CM4_Treatment_growth_rates, ...
                            CM5_Treatment_growth_rates, ...
                            CM6_Treatment_growth_rates, ...
                            CM7_Treatment_growth_rates];

Complete_Avg_Init_Rates = [CM3_Average_initial_growth_rates, ...
                           CM4_Average_initial_growth_rates, ...
                           CM5_Average_initial_growth_rates, ...
                           CM6_Average_initial_growth_rates, ...
                           CM7_Average_initial_growth_rates];
```

```
Complete_Avg_Treat_Rates = [CM3_Average_treatment_growth_rates, ...
                            CM4_Average_treatment_growth_rates, ...
                            CM5_Average_treatment_growth_rates, ...
                            CM6_Average_treatment_growth_rates, ...
                            CM7_Average_treatment_growth_rates];



%% Functions

% Function to find the initial growth rate

function initial_growth = init_grow(Vol_mat, Time_mat, Injection_times)

initial_growth = [];
num_tumors = length(Vol_mat(:,1));

for i = 1:num_tumors
    tumor_start_location = find(Vol_mat(i,:), 1, "first");
    tumor_start = Time_mat(i, tumor_start_location);
    inject_date_location = Injection_times(i);
    inject_date = Time_mat(i, inject_date_location);
    total_time = (inject_date - tumor_start);
    if total_time <= 0
        initial_growth(i) = 0;
    else
        vol_0 = Vol_mat(i, tumor_start_location);
        vol_final = Vol_mat(i, inject_date_location);
        growth_rate = (vol_final / vol_0)^(1/total_time) - 1;
        initial_growth(i) = growth_rate;
    end
end
end

% Function to find the treatment growth rate

function treatment_growth = treat_grow(Vol_mat, Time_mat, Injection_times)

treatment_growth = [];
num_tumors = length(Vol_mat(:,1));

for i =1:num_tumors
```

```
    inject_date_location = Injection_times(i);
    inject_date = Time_mat(i, inject_date_location);
    tumor_end_location = find(Vol_mat(i,:), 1, "last");
    tumor_end = Time_mat(i, tumor_end_location);
    total_time = (tumor_end - inject_date);
    if total_time <= 0
        treatment_growth(i) = 0;
    else
        vol_0 = Vol_mat(i, inject_date_location);
        if vol_0 == 0
            vol_0 = find(Vol_mat(i,:), 1, "first");
        else
            vol_0 = vol_0;
        end
        vol_final = Vol_mat(i, tumor_end_location);
        growth_rate = (vol_final / vol_0)^(1/total_time) - 1;
        treatment_growth(i) = growth_rate;
    end
end
end
```

A.2 Gaussian Mixture Models with EM Algorithm Code

Note that the code is initialized for k=2 and all options aka diagonal, full; shared true, shared false. To get certain graphs you may have to specify which conditions you want. Also note that not all graphs are possible due to ill-conditioned covariance matrices and an index error. Note that all possible graphs are found in the project (Section 4).

```
    clear vars; close all; format long;
% Math 590 Project: Unsupervised Learning on Cancer Data

% Steps: - Import data of mice
%        - Determine growth rate of tumors
%            - Before Treatment
%            - After Treatment
%        - Unsupervised Learning Cluster Technique 1: GMM EM Algorithm
%            - Investigate other features inside each cluster
%        - Unsupervised Learning Cluster Technique 2: Agglomerative Clusters
%            - Investigate other features inside each cluster
%        - Unsupervised Learning Cluster Technique 3: K-means Clustering
%            - Investigate other features inside each cluster

%% Unsupervised Learning Technique 1: GMM EM Algorithm
```

```
% First we load the data and plot the two sets

% Data Set 1
load('Complete_Init_Rates.mat')
load('Complete_Treat_Rates.mat')
Comp_Init = Complete_Initial_Rates';
Comp_Treat = Complete_Treatment_Rates';
Data_Set_1 = [Comp_Init, Comp_Treat];

% Data Set 2
load('Complete_Avg_Init_Rates.mat')
load('Complete_Avg_Treat_Rates.mat')
Avg_Init = Complete_Avg_Init_Rates';
Avg_Treat = Complete_Avg_Treat_Rates';
Data_Set_2 = [Avg_Init, Avg_Treat];

%% Plotting Data

% Plotting Data Set 1
figure
plot(Complete_Initial_Rates, Complete_Treatment_Rates, '.', 'MarkerSize', 15)
xlabel('Initial Growth Rate')
ylabel('Treatment Growth Rate')
title('All Tumors')
hold on;

% Plotting Data Set 2
figure
plot(Complete_Avg_Init_Rates, Complete_Avg_Treat_Rates, '.', 'MarkerSize', 15)
xlabel('Initial Growth Rate')
ylabel('Treatment Growth Rate')
title('All Mice')
hold on;


%% GMM EM algorithm Data Set 1

[n1,p1] = size(Data_Set_1);                            % # of points, # of features

figure
plot(Data_Set_1(:,1), Data_Set_1(:,2), '.', 'MarkerSize', 15)
```

```
title('Data Set 1 - All Tumors');
xlabel('Initial Growth Rate');
ylabel('Treatment Growth Rate');





rng(3);                                                 % Create random seed
k = 2;                                                  % Number of clusters,
%                                    remember to change legend if changing k
options = statset('MaxIter', 1000);                     % Max Iteration
Sigma = {'diagonal', 'full'};                           % Covariance Type
% Sigma = {'full'};                                     % Covariance Type
% Sigma = {'diagonal'};                                 % Covariance Type
nSigma = numel(Sigma);                                  % # of types of Cov
SharedCovariance = {true, false};                       % Yes/No for shared
SCtext = {'true', 'false'};                             % Text format
% SharedCovariance = {true};                            % Yes/No for shared
% SCtext = {'true'};                                    % Text format
% SharedCovariance = {false};                           % Yes/No for shared
% SCtext = {'false'};                                   % Text format
nSC = numel(SharedCovariance);                          % # of options

% Make 2D grid

d = 500;                                                % # of grid points
% Creating points on axes
x1 = linspace(min(Data_Set_1(:,1)) - 2, max(Data_Set_1(:,1)) + 2, d);
x2 = linspace(min(Data_Set_1(:,2)) - 2, max(Data_Set_1(:,2)) + 2, d);
[x1grid, x2grid] = meshgrid(x1,x2);                     % resizing x1 and x2
X0 = [x1grid(:) x2grid(:)];                             % Getting all possible
%                                          combos for x1 and x2 on themselves

% EM algorithm

threshold = sqrt(chi2inv(0.99, 2));
count = 1;

for i = 1:nSigma
    for j = 1:nSC
        % Fitted GMM
        gmfit = fitgmdist(Data_Set_1,k,'CovarianceType',Sigma{i}, ...
                          'SharedCovariance',SharedCovariance{j}, ...
                          'Options',options);
        % Getting clusters
        clusterData1 = cluster(gmfit, Data_Set_1);
        % Compute Mahal distance for each grid point to each GMM piece
        mahalDist = mahal(gmfit, X0);

        % Make ellipsoids for each GMM piece and plot clusters
        subplot(2,2, count);
        h1 = gscatter(Data_Set_1(:,1), Data_Set_1(:,2), clusterData1);
        hold on
            for m = 1:k
                idx = mahalDist(:,m)<=threshold;
                Color = h1(m).Color*0.75 - 0.5*(h1(m).Color - 1);
                h2 = plot(X0(idx,1),X0(idx,2),'.','Color',Color,'MarkerSize',1);
                uistack(h2,'bottom');
            end
        plot(gmfit.mu(:,1),gmfit.mu(:,2),'kx','LineWidth',2,'MarkerSize',10)
        title(sprintf('Sigma is %s\nSharedCovariance = %s',...
            Sigma{i},SCtext{j}),'FontSize',8)
        legend(h1,{'1','2'})
        hold off
        count = count + 1;
    end
end



%% GMM EM Algorithm Data Set 2

[n2,p2] = size(Data_Set_2);                            % # of points, # of features

plot(Data_Set_2(:,1), Data_Set_2(:,2), '.', 'MarkerSize', 15)
title('Data Set 2 - All Mice');
xlabel('Initial Growth Rate');
ylabel('Treatment Growth Rate');
```

```
rng(3);                                                    % Create random seed
k = 2;                                                     % Number of clusters,
%                                    remember to change legend if changing k
options = statset('MaxIter', 1000);                        % Max Iteration
Sigma = {'diagonal', 'full'};                              % Covariance Type
% Sigma = {'full'};                                        % Covariance Type
% Sigma = {'diagonal'};                                    % Covariance Type
nSigma = numel(Sigma);                                     % # of types of Cov
SharedCovariance = {true, false};                          % Yes/No for shared
SCtext = {'true', 'false'};                                % Text format
% SharedCovariance = {true};                               % Yes/No for shared
% SCtext = {'true'};                                       % Text format
% SharedCovariance = {false};                              % Yes/No for shared
% SCtext = {'false'};                                      % Text format
nSC = numel(SharedCovariance);                             % # of options

% Make 2D grid

d = 500;                                                   % # of grid points
% Creating points on axes
x1 = linspace(min(Data_Set_2(:,1)) - 2, max(Data_Set_2(:,1)) + 2, d);
x2 = linspace(min(Data_Set_2(:,2)) - 2, max(Data_Set_2(:,2)) + 2, d);
[x1grid, x2grid] = meshgrid(x1,x2);                        % resizing x1 and x2
X0 = [x1grid(:) x2grid(:)];                                % Getting all possible
%                                         combos for x1 and x2 on themselves

% EM algorithm

threshold = sqrt(chi2inv(0.99, 2));
count = 1;

for i = 1:nSigma
    for j = 1:nSC
        % Fitted GMM
        gmfit = fitgmdist(Data_Set_2,k,'CovarianceType',Sigma{i}, ...
                          'SharedCovariance',SharedCovariance{j}, ...
                          'Options',options);
        % Getting clusters
        clusterData1 = cluster(gmfit, Data_Set_2);
        % Compute Mahal distance for each grid point to each GMM piece
        mahalDist = mahal(gmfit, X0);
        % Make ellipsoids for each GMM piece and plot clusters
        subplot(2,2, count);
        h1 = gscatter(Data_Set_2(:,1), Data_Set_2(:,2), clusterData1);
        hold on
            for m = 1:k
                idx = mahalDist(:,m)<=threshold;
                Color = h1(m).Color*0.75 - 0.5*(h1(m).Color - 1);
                h2 = plot(X0(idx,1),X0(idx,2),'.','Color',Color,'MarkerSize',1);
                uistack(h2,'bottom');
            end
        plot(gmfit.mu(:,1),gmfit.mu(:,2),'kx','LineWidth',2,'MarkerSize',10)
        title(sprintf('Sigma is %s\nSharedCovariance = %s',...
            Sigma{i},SCtext{j}),'FontSize',8)
        legend(h1,{'1','2'})
        hold off
        count = count + 1;
    end
end
```

A.3 Hierarchical Agglomerative Clustering Code

This code should be ran by sections pertaining to which method you wish to use. Doing this will give all of the graphs for the method and cluster number = 2. Note that you must change the k to get other number of clusters. Note that all graphs are found in the project (Section 4).

```
    clear vars; close all; format long;
% Math 590 Project: Unsupervised Learning on Cancer Data

% Steps: - Import data of mice
%        - Determine growth rate of tumors
%           - Before Treatment
%           - After Treatment
%        - Unsupervised Learning Cluster Technique 1: GMM EM Algorithm
%           - Investigate other features inside each cluster
%        - Unsupervised Learning Cluster Technique 2: Agglomerative Clusters
%           - Investigate other features inside each cluster
%        - Unsupervised Learning Cluster Technique 3: K-means Clustering
%           - Investigate other features inside each cluster

%% Unsupervised Learning Technique 2: Agglomerative Clustering
% - Looking at the following types: Single, Complete, Average, Ward
% First we load the data sets

% Data Set 1
load('Complete_Init_Rates.mat')
load('Complete_Treat_Rates.mat')
Comp_Init = Complete_Initial_Rates';
Comp_Treat = Complete_Treatment_Rates';
Data_Set_1 = [Comp_Init, Comp_Treat];

% Data Set 2
load('Complete_Avg_Init_Rates.mat')
load('Complete_Avg_Treat_Rates.mat')
Avg_Init = Complete_Avg_Init_Rates';
Avg_Treat = Complete_Avg_Treat_Rates';
Data_Set_2 = [Avg_Init, Avg_Treat];


%% Single Linkage

% Data Set 1

Data1_dist = pdist(Data_Set_1);

Mat_data1_dist = squareform(Data1_dist);

Data1_SingleLink = linkage(Data1_dist);

figure
dendrogram(Data1_SingleLink,0)
xlabel('Data Points')
ylabel('Distance between Points')
title('Dendrogram - Single Link Tumors')

figure
for i = 1:4
    subplot(2,2,i)
    Data1_single_clusters = cluster(Data1_SingleLink,'maxclust',i+1);
    gscatter(Data_Set_1(:,1), Data_Set_1(:,2), Data1_single_clusters)
    xlabel('Initial Growth Rate')
    ylabel('Treatment Growth Rate')
    title_name = ['Agglo ' num2str(i+1) ' Clusters - Single Link Tumors'];
    title(title_name)
end

% Data Set 2

Data2_dist = pdist(Data_Set_2);

Mat_data2_dist = squareform(Data2_dist);

Data2_SingleLink = linkage(Data2_dist);

figure
dendrogram(Data2_SingleLink,0)
xlabel('Data Points')
ylabel('Distance between Points')
title('Dendrogram - Single Link Mice')

figure
for i = 1:4
    subplot(2,2,i)
    Data2_single_clusters = cluster(Data2_SingleLink,'maxclust',i+1);
    gscatter(Data_Set_2(:,1), Data_Set_2(:,2), Data2_single_clusters)
    xlabel('Initial Growth Rate')
    ylabel('Treatment Growth Rate')
    title_name = ['Agglo ' num2str(i+1) ' Clusters - Single Link Mice'];
    title(title_name)
end

%% Complete Linkage

% Data Set 1

% Complete needs to take a square form
Data1_CompleteLink = linkage(Mat_data1_dist, 'complete');

figure
dendrogram(Data1_CompleteLink,0)
xlabel('Data Points')
ylabel('Distance between Points')
title('Dendrogram - Complete Link Tumors')

figure
```

```
for i = 1:4
    subplot(2,2,i)
    Data1_complete_clusters = cluster(Data1_CompleteLink, 'maxclust',i+1);
    gscatter(Data_Set_1(:,1), Data_Set_1(:,2), Data1_complete_clusters)
    xlabel('Initial Growth Rate')
    ylabel('Treatment Growth Rate')
    title_name = ['Agglo ' num2str(i+1) ' Clusters - Complete Link Tumors'];
    title(title_name)
end


% Data Set 2

% Complete needs to take a square form
Data2_CompleteLink = linkage(Mat_data2_dist, 'complete');

figure
dendrogram(Data2_CompleteLink,0)
xlabel('Data Points')
ylabel('Distance between Points')
title('Dendrogram - Complete Link Mice')

figure
for i = 1:4
    subplot(2,2,i)
    Data2_complete_clusters = cluster(Data2_CompleteLink, 'maxclust',i+1);
    gscatter(Data_Set_2(:,1), Data_Set_2(:,2), Data2_complete_clusters)
    xlabel('Initial Growth Rate')
    ylabel('Treatment Growth Rate')
    title_name = ['Agglo ' num2str(i+1) ' Clusters - Complete Link Mice'];
    title(title_name)
end


%% Average Linkage

% Data Set 1

Data1_AverageLink = linkage(Data1_dist, 'average');

figure
dendrogram(Data1_AverageLink,0)
```

```
xlabel('Data Points')
ylabel('Distance between Points')
title('Dendrogram - Average Link Tumors')

figure
for i = 1:4
    subplot(2,2,i)
    Data1_average_clusters = cluster(Data1_AverageLink, 'maxclust',i+1);
    gscatter(Data_Set_1(:,1), Data_Set_1(:,2), Data1_average_clusters)
    xlabel('Initial Growth Rate')
    ylabel('Treatment Growth Rate')
    title_name = ['Agglo ' num2str(i+1) ' Clusters - Average Link Tumors'];
    title(title_name)
end


% Data Set 2

Data2_AverageLink = linkage(Data2_dist, 'average');

figure
dendrogram(Data2_AverageLink,0)
xlabel('Data Points')
ylabel('Distance between Points')
title('Dendrogram - Average Link Mice')

figure
for i = 1:4
    subplot(2,2,i)
    Data2_average_clusters = cluster(Data2_AverageLink, 'maxclust',i+1);
    gscatter(Data_Set_2(:,1), Data_Set_2(:,2), Data2_average_clusters)
    xlabel('Initial Growth Rate')
    ylabel('Treatment Growth Rate')
    title_name = ['Agglo ' num2str(i+1) ' Clusters - Average Link Mice'];
    title(title_name)
end


%% Ward Linkage

% Data Set 1
```

```
Data1_WardLink = linkage(Data1_dist, 'ward');

figure
dendrogram(Data1_WardLink,0)
xlabel('Data Points')
ylabel('Distance between Points')
title('Dendrogram - Ward Link Tumors')

figure
for i = 1:4
    subplot(2,2,i)
    Data1_ward_clusters = cluster(Data1_WardLink, 'maxclust',i+1);
    gscatter(Data_Set_1(:,1), Data_Set_1(:,2), Data1_ward_clusters)
    xlabel('Initial Growth Rate')
    ylabel('Treatment Growth Rate')
    title_name = ['Agglo ' num2str(i+1) ' Clusters - Ward Link Tumors'];
    title(title_name)
end


% Data Set 2

Data2_WardLink = linkage(Data2_dist, 'ward');

figure
dendrogram(Data2_WardLink,0)
xlabel('Data Points')
ylabel('Distance between Points')
title('Dendrogram - Ward Link Mice')

figure
for i = 1:4
    subplot(2,2,i)
    Data2_ward_clusters = cluster(Data2_WardLink, 'maxclust',i+1);
    gscatter(Data_Set_2(:,1), Data_Set_2(:,2), Data2_ward_clusters)
    xlabel('Initial Growth Rate')
    ylabel('Treatment Growth Rate')
    title_name = ['Agglo ' num2str(i+1) ' Clusters - Ward Link Mice'];
    title(title_name)
end
```

A.4 K means Code

This code runs perfectly and gives you both the tumor and mice graphs for k=2,3,4,5. :)

```
    clear vars; close all; format long;
% Math 590 Project: Unsupervised Learning on Cancer Data

% Steps: - Import data of mice
%        - Determine growth rate of tumors
%           - Before Treatment
%           - After Treatment
%        - Unsupervised Learning Cluster Technique 1: GMM EM Algorithm
%           - Investigate other features inside each cluster
%        - Unsupervised Learning Cluster Technique 2: Agglomerative Clusters
%           - Investigate other features inside each cluster
%        - Unsupervised Learning Cluster Technique 3: K-means Clustering
%           - Investigate other features inside each cluster

%% Unsupervised Learning Technique 3: K-means


% First we load the data sets
% Data Set 1
load('Complete_Init_Rates.mat')
load('Complete_Treat_Rates.mat')
Comp_Init = Complete_Initial_Rates';
Comp_Treat = Complete_Treatment_Rates';
Data_Set_1 = [Comp_Init, Comp_Treat];

% Data Set 2
load('Complete_Avg_Init_Rates.mat')
load('Complete_Avg_Treat_Rates.mat')
Avg_Init = Complete_Avg_Init_Rates';
Avg_Treat = Complete_Avg_Treat_Rates';
Data_Set_2 = [Avg_Init, Avg_Treat];

%% K-means Data Set 1

rng default

options = statset('Display', 'final');

% K = 2
```

```
[idx,C] = kmeans(Data_Set_1,2,'Replicates',5,'Options',options);

figure;
plot(Data_Set_1(idx==1,1),Data_Set_1(idx==1,2),'.','MarkerSize',15)
hold on
plot(Data_Set_1(idx==2,1),Data_Set_1(idx==2,2),'.','MarkerSize',15)
plot(C(:,1),C(:,2),'kx',...
     'MarkerSize',15,'LineWidth',3)
xlabel('Initial Growth Rate')
ylabel('Treatment Growth Rate')
legend('Cluster 1','Cluster 2','Centroids',...
       'Location','NE')
title 'K-means 2 Clusters - Tumors'
hold off

% k = 3
[idx,C] = kmeans(Data_Set_1,3,'Replicates',5,'Options',options);

figure;

plot(Data_Set_1(idx==1,1),Data_Set_1(idx==1,2),'.','MarkerSize',15)
hold on
plot(Data_Set_1(idx==2,1),Data_Set_1(idx==2,2),'.','MarkerSize',15)
hold on
plot(Data_Set_1(idx==3,1),Data_Set_1(idx==3,2),'.','MarkerSize',15)

plot(C(:,1),C(:,2),'kx',...
     'MarkerSize',15,'LineWidth',3)
xlabel('Initial Growth Rate')
ylabel('Treatment Growth Rate')
legend('Cluster 1','Cluster 2','Cluster 3', 'Centroids',...
       'Location','NE')
title 'K-means 3 Clusters - Tumors'
hold off

% K = 4
[idx,C] = kmeans(Data_Set_1,4,'Replicates',5,'Options',options);

figure;

plot(Data_Set_1(idx==1,1),Data_Set_1(idx==1,2),'.','MarkerSize',15)
hold on
```

```
plot(Data_Set_1(idx==2,1),Data_Set_1(idx==2,2),'.','MarkerSize',15)
hold on
plot(Data_Set_1(idx==3,1),Data_Set_1(idx==3,2),'.','MarkerSize',15)
hold on
plot(Data_Set_1(idx==4,1),Data_Set_1(idx==4,2),'.','MarkerSize',15)

plot(C(:,1),C(:,2),'kx',...
     'MarkerSize',15,'LineWidth',3)
xlabel('Initial Growth Rate')
ylabel('Treatment Growth Rate')
legend('Cluster 1','Cluster 2','Cluster 3','Cluster 4 ','Centroids',...
       'Location','NE')
title 'K-means 4 Clusters - Tumors'
hold off

% K = 5
[idx,C] = kmeans(Data_Set_1,5,'Replicates',5,'Options',options);

figure;

plot(Data_Set_1(idx==1,1),Data_Set_1(idx==1,2),'.','MarkerSize',15)
hold on
plot(Data_Set_1(idx==2,1),Data_Set_1(idx==2,2),'.','MarkerSize',15)
hold on
plot(Data_Set_1(idx==3,1),Data_Set_1(idx==3,2),'.','MarkerSize',15)
hold on
plot(Data_Set_1(idx==4,1),Data_Set_1(idx==4,2),'.','MarkerSize',15)
hold on
plot(Data_Set_1(idx==5,1),Data_Set_1(idx==5,2),'.','MarkerSize',15)

plot(C(:,1),C(:,2),'kx',...
     'MarkerSize',15,'LineWidth',3)
xlabel('Initial Growth Rate')
ylabel('Treatment Growth Rate')
legend('Cluster 1','Cluster 2','Cluster 3','Cluster 4 ','Cluster 5',...
    'Centroids','Location','NE')
title 'K-means 5 Clusters - Tumors'



%% K-means Data Set 2
```

```
rng default

options = statset('Display', 'final');

% K = 2
[idx,C] = kmeans(Data_Set_2,2,'Replicates',5,'Options',options);

figure;
plot(Data_Set_2(idx==1,1),Data_Set_2(idx==1,2),'.','MarkerSize',15)
hold on
plot(Data_Set_2(idx==2,1),Data_Set_2(idx==2,2),'.','MarkerSize',15)
plot(C(:,1),C(:,2),'kx',...
     'MarkerSize',15,'LineWidth',3)
xlabel('Initial Growth Rate')
ylabel('Treatment Growth Rate')
legend('Cluster 1','Cluster 2','Centroids',...
       'Location','NE')
title 'K-means 2 Clusters - Mice'
hold off

% k = 3
[idx,C] = kmeans(Data_Set_2,3,'Replicates',5,'Options',options);

figure;

plot(Data_Set_2(idx==1,1),Data_Set_2(idx==1,2),'.','MarkerSize',15)
hold on
plot(Data_Set_2(idx==2,1),Data_Set_2(idx==2,2),'.','MarkerSize',15)
hold on
plot(Data_Set_2(idx==3,1),Data_Set_2(idx==3,2),'.','MarkerSize',15)

plot(C(:,1),C(:,2),'kx',...
     'MarkerSize',15,'LineWidth',3)
xlabel('Initial Growth Rate')
ylabel('Treatment Growth Rate')
legend('Cluster 1','Cluster 2','Cluster 3', 'Centroids',...
       'Location','NE')
title 'K-means 3 Clusters - Mice'
hold off

% K = 4
[idx,C] = kmeans(Data_Set_2,4,'Replicates',5,'Options',options);
```

```
figure;

plot(Data_Set_2(idx==1,1),Data_Set_2(idx==1,2),'.','MarkerSize',15)
hold on
plot(Data_Set_2(idx==2,1),Data_Set_2(idx==2,2),'.','MarkerSize',15)
hold on
plot(Data_Set_2(idx==3,1),Data_Set_2(idx==3,2),'.','MarkerSize',15)
hold on
plot(Data_Set_2(idx==4,1),Data_Set_2(idx==4,2),'.','MarkerSize',15)

plot(C(:,1),C(:,2),'kx',...
     'MarkerSize',15,'LineWidth',3)
xlabel('Initial Growth Rate')
ylabel('Treatment Growth Rate')
legend('Cluster 1','Cluster 2','Cluster 3','Cluster 4 ','Centroids',...
       'Location','NE')
title 'K-means 4 Clusters - Mice'
hold off

% K = 5
[idx,C] = kmeans(Data_Set_2,5,'Replicates',5,'Options',options);

figure;

plot(Data_Set_2(idx==1,1),Data_Set_2(idx==1,2),'.','MarkerSize',15)
hold on
plot(Data_Set_2(idx==2,1),Data_Set_2(idx==2,2),'.','MarkerSize',15)
hold on
plot(Data_Set_2(idx==3,1),Data_Set_2(idx==3,2),'.','MarkerSize',15)
hold on
plot(Data_Set_2(idx==4,1),Data_Set_2(idx==4,2),'.','MarkerSize',15)
hold on
plot(Data_Set_2(idx==5,1),Data_Set_2(idx==5,2),'.','MarkerSize',15)

plot(C(:,1),C(:,2),'kx',...
     'MarkerSize',15,'LineWidth',3)
xlabel('Initial Growth Rate')
ylabel('Treatment Growth Rate')
legend('Cluster 1','Cluster 2','Cluster 3','Cluster 4 ','Cluster 5',...
    'Centroids','Location','NE')
title 'K-means 5 Clusters - Mice'
```

References

[1] 5-Fluorouracil, 5 grams. RPI. (n.d.-a). https://www.rpicorp.com/products/biochemicals/biochemical-reagents/5-fluorouracil-5-g.html

[2] AGGLOMERTIVE hierarchical clustering using ward linkage. (n.d.). https://jbhender.github.io/Stats506/F18/GP/Group10.html

[3] "Cancer Facts and Figures 2023." American Cancer Society, www.cancer.org/research/cancer-facts-statistics/all-cancer-facts-figures/2023-cancer-facts-figures.html.

[4] Cluster. Construct agglomerative clusters from linkages - MATLAB. (n.d.). https://www.mathworks.com/help/stats/cluster.html

[5] El-Masry, Omar S, et al. "Oral Intragastric DMBA Administration Induces Acute Lymphocytic Leukemia and Other Tumors in Male Wistar Rats." Journal of Experimental Pharmacology, U.S. National Library of Medicine, 25 Feb. 2022, www.ncbi.nlm.nih.gov/pmc/articles/PMC8887968/#:~:text=have%20been%20made.-,7%2C12%2Ddimethylbenz%5Ba%5D%2Danthracene%20(DMBA),that%20initiates %20and%20promotes%20tumorigenesis.

[6] Fitgmdist. MathWorks. (n.d.). https://www.mathworks.com/help/stats/clustering-using-gaussian-mixture-models.html

[7] Hierarchical Clustering. Hierarchical clustering - MATLAB & simulink. (n.d.). https://www.mathworks.com/help/stats/hierarchical-clustering.html

[8] Kmeans. k. (n.d.). https://www.mathworks.com/help/stats/kmeans.html#namevalue pairarguments

[9] Linkage. Agglomerative hierarchical cluster tree - MATLAB. (n.d.). https://www.mathworks.com/help/stats/linkage.html#mw_59e9693d-3784-4a0d-89dd-5dd020a605b2

[10] Loizides, C., Iacovides, D., Hadjiandreou, M. M., Rizki, G., Achilleos, A., Strati, K., and Mitsis, G. D. (n.d.). Model-based tumor growth dynamics and therapy response in a mouse model of de novo carcinogenesis. PLOS ONE. https://journals.plos.org/plosone/article?id=10.1371%2Fjournal.pone.0143840#pone.0143840.ref007

[11] MA;, Kolb TM;Davis. "The Tumor Promoter 12-O-Tetradecanoylphorbol 13-Acetate (TPA) Provokes a Prolonged Morphologic Response and ERK Activation in TSC2-Null Renal Tumor Cells." Toxicological Sciences: An Official Journal of the Society of Toxicology, U.S. National Library of Medicine, pubmed.ncbi.nlm.nih.gov/15178807/.

[12] Murphy, K. P. (2022). Probabilistic machine learning: An introduction. The MIT Press.

[13] Padcat. File Exchange - MATLAB Central. (n.d.). https://www.mathworks.com/matlabcentral/fileexchange/22909-padcat

[14] U.S. National Library of Medicine. (n.d.). Imiquimod topical: Medlineplus Drug Information. MedlinePlus.
https://medlineplus.gov/druginfo/meds/a698010.html#: :text=Imiquimod%20is%20in %20a%20class,or%20superficial%20basal%20cell%20carcinoma.

[15] V79-4 - CCL-93 — ATCC. (n.d.-a). https://www.atcc.org/products/ccl-93

[16] YouTube. (2015, March 6). Clustering (4): Gaussian mixture models and em. YouTube. https://www.youtube.com/watch?v=qMTuMa86NzU&ab_channel=AlexanderIhler

[17] YouTube. (2019, March 27). Applied machine learning 2019 - lecture 15 - clustering and mixture models. YouTube. https://www.youtube.com/watch?v=q0QLdPphV00&ab_channel=AndreasMueller