A COMPANY PROJECT IN

Statistical Techniques for Detection of Bacteria in Hyperspectral Images

Author:

MSc student, Nina Louise PEDERSEN

Supervisors:

Associate Professor, Hans Christian PETERSEN

Head of Research Newtec Engineering A/S, Bjarke JØRGENSEN

February 6, 2019

This project is made in cooperation with

NEWTEC

Resumé

I dette projekt blev der opstillet en metode til detektering af bakterier på overfladen af råt kød ved hjælp af hyperspektrale billeder, som var produceret og forarbejdet med udgangspunkt i den chemometriske metode uafhængig komponentanalyse. Anvendelsesmulighederne i projektet var at kunne inkorporere metoden i det internationale firma Newtecs allerede eksisterende mekaniske løsninger til sortering og pakning af fødevarer. Det vil potentielt kunne medføre langt færre tilbagekaldelser i fødevareindustrien ved detektering direkte på transportbåndet. Endvidere vil spare den enkelte fødevarevirksomhed for både direkte- og socioøkonomiske tab. I dette projekt er det taget udgangspunkt i hakket oksekød og koteletter.

Den chemometriske metode implementeret i dette projekt har på en tilfredsstillende måde været i stand til at isolere de hyperspektrale bølgelængder optaget af kødet alene frem for baggrunden. For endvidere at udføre uafhængig komponentanalyse anvendes algoritmen fastICA, som er en effektiv metode til estimering af de uafhængige komponenter, der kan anvendes til egenskab selektion. Uafhængig komponentanalyse adskiller de forskellige egenskaber i kødet, hvoraf bakterierne vil være at finde i en komponent.

På baggrund af den opstillede chemometriske metode kan det konkluderes, at det er muligt ud fra hyperspektrale billeder at detektere bakterier på overfladen af kød - selvom det dog i dette projekt kun var muligt at fuldt ud identificere bakterierne på koteletterne og ikke det hakkede oksekød. Det tyder på, at vi arbejder med Gaussisk fordelt data, hvilket ikke kan anvendes med uafhængig komponentanalyse.

Flere studier er nødvendige, for at udvikle en endegyldig metode, der kan fungere på alle kødtyper og oveflader, men dette projekt er et vigtigt første skridt på vejen mod større fødevaresikkerhed og færre økonomiske tab for fødevareproducenterne ved tilbagekaldelser.

Contents

1 Introduction

Bacterial growth in retail meat can lead to a recall of the product and food-borne illnesses when consumed, which can endanger the consumers health (Pozo & Schroeder, 2015; Zheng *et al.* , 2017). Therefore, it is important to do quality control in the food industry for contaminants before it is consumed. The consumption of meat have increased over the past 50 years (Ritchie & Roser, 2018) and with this increasing industry, it is important to keep up the hygiene and make sure that the products is not contaminated when leaving the factory. Today's methods for detecting bacterial growth cannot keep up with the increasing production pace of the industry, and today many consumable food products are shipped to retail stores before all test results are in. A typical detection method will be destructive and have an incubation period of more than 24 hours (Zheng *et al.* , 2017) and in a world where we want our meat to be as fresh as possible, we do not have the time to do proper controls. When a company recalls a product they have to recall everything produced on the given dates that they suspect contamination, which therefore can include food that is not contaminated. Once the public is aware of a recall, it will also have a negative impact on the company's market value (Pozo & Schroeder, 2015). As result there is a growing global marked for easy, fast, and reliable methods for doing quality control of retail meat as early in the packing process as possible.

The Research Department at the company Newtec have been working on methods for early detection of contaminated foods. They are a company that develops industrial weighing, sorting and packing machinery and they want to create a machine that based on Hyperspectal imaging (HSI) detects bacteria in retail meat.

This project focuses on detection of bacterial growth during the packing process in the food industry and investigates the possibility of creating a non-destructive method directly applicable on the conveyor belt. Thus, it must be able to do a real-time detection of bacteria. HSI is used to generate data that our chemometric method will use to classify the features in the images. The advantage to using HSI is that it contains spectral and spatial information of an object.

There exist only sparse literature on this topic, however; they are all focusing on

methods like Partial Least Square Regression (PLSR) or Principle Component Analysis (PCA)(Huang *et al.* , 2013; Feng *et al.* , 2013). In this project, we will describe another chemometric method that have not previously been used as much in this context and use it to improve the detection of bacteria using HSI data. We will employ Independent Component Analysis (ICA) for feature selection and reduction of components in our data.

The structure of this work reflects the process from idea to execution. We will be working with raw data provided by Newtec and we will need to do some preprocessing before doing the analysis. The preprocessing consists of removing any unwanted background noise in the images prior to doing the analysis. The analysis gives a superficial review of PCA and a more thorough review of the ICA given it is the one we will apply. In summery, this project will lay the ground for more thorough analyses conducted in a master thesis project.

1.1 Company Project

This project is made in cooperation with the Research Department at Newtec. An important lesson learned whilst doing this company project must be that everything is a process. From something simply not working and you are struggling to find a solution to getting a positive solution without understanding how. You are not provided with a data set that works perfectly with your methods and you will have to examine the data set in a way you will not come across in a regular course. This makes you able to take on a job after getting your degree without being surprised that the world is not perfectly aligned to work with what you have been taught.

The results created during this project have also been to benefit another project that Newtec are doing where they train an Artificial Neural Network (ANN). They require some training data and the independent components (ICs) that the method in this projects generates can be used to train this network.

All of these experiences would not occur if the project was done at the university. Sure, we could have been provided with the same data set but we would not be presented with them in the same way. You can really understand were the data is coming from

when getting it directly from the laboratory.

1.2 Hyperspectral imaging

HSI obtains the spectrum for each pixel of an object by projecting light onto the sample, which in turn reflects a smaller amount back after interacting with the object. It combines the features from imaging and spectroscopy to achieve spatial and spectral information of the object (Park & Lu, 2015). The hyperspectral image is formed as a stack of two-dimensional spatial images of the sample. The height of this stack depends on how many bands you want to look at. This type of structure is called a hypercube and it has two spatial dimensions and one spectral dimension. This enables us to look at it as an image for a single wavelength or as a spectrum for a particular pixel (Sendin *et al.* , 2018).

Since we have a hypercube and it is a three dimensional data structure and most chemometric methods are based on matrices, we will need to rearrange it prior to analysis (Park & Lu, 2015). The output is a matrix that have the format $(200 \times 500) \times 400$ as seen below. Here we can see the layers of the cube, which is also depicted on figure 1.

$$\begin{bmatrix} a_{1,1} & a_{1,2} & \dots & a_{1,399} & a_{1,400} \\ a_{2,1} & \ddots & & & \\ \vdots & & & & \\ a_{200,1} & & & & \\ b_{1,1} & b_{1,2} & \dots & b_{1,399} & b_{1,400} \\ b_{2,1} & \ddots & & & \\ \vdots & & & & \\ b_{200,1} & \dots & & & \\ c_{1,1} & \dots & & & \\ \vdots & \ddots & & & \\ & & & & \end{bmatrix} \tag{1}$$

We rearrange this matrix by unfolding the three dimensional hypercube into a two dimensional matrix where each spectra is stacked, as we can see in the top of figure 1.

After removing the background from the data, we can perform an analysis and the result can be refolded into a hypercube again. Each layer of the hypercube can be viewed as an image plane, as can be seen in the right corner on figure 1 and we can examine each image plane at each band.

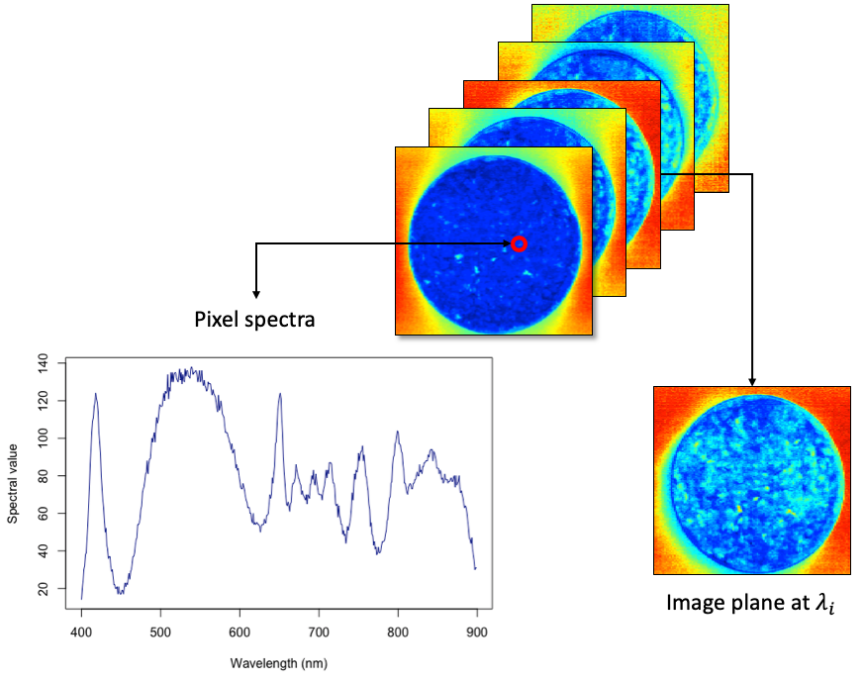


Figure 1: Schematic representation of a hyperspectral image of minced beef showing the relationship between spectral and spatial dimension.

One of the advantages to HSI is that we in each pixel of the data have a whole spectrum as illustrated in the left corner of figure 1. This give us a large quantity of data, and main issue will therefore be to extract useful and meaningful information from the images. We can already see that we have plenty of information to investigate and the logical proceedings from here, would therefore be to only look at the most important part and reduce the dimension of the data.

2 Data preprocessing

We will do some data preparation before applying our methods to the data. The idea behind doing preprocessing is to help the analysis to not focus on the background and give a better result for the part containing the sample. Therefore, we will be isolating the sample and reduce the data to a minor section. The data we will apply the preprocessing to is a data set with minced beef created by the Kallipolitis research group at the department of biochemistry and molecular biology at SDU. The data is based on HSI of minced beef in a Petri dish under different circumstances. We will in this project be looking at experiment 18 and 19, which contain 68 and 56 data sets, respectively. The bacteria plated is *Listeria Innocua* and the meat is minced beef.

In the interest of keeping the preprocessing methods general, we cannot simply look at the image in order to isolate a large chunk of Petri dish with beef. We will make a preprocessing that we can apply to new experimental data. Also, we want to reduce human involvement in the selection of the section containing the sample.

The preprocessing consist of several methods where the objective is to increase the contrast between the Petri dish and the background in order to do a better isolation of the section containing the Petri dish with minced beef. Surprisingly, we have been able to use some very simple methods in order to achieve this data separation. The implementation of the methods can found in Appendix A.1.

2.1 Sum of squares

We use the distance between the mean value and each point to find the sum of squares. It gives us the deviation from the mean

$$SS = \sum_{i=1}^{n} (x_i - \bar{x})^2 . \tag{2}$$

We calculate the sum of squares for each line and compare it to an average sum of squares calculated from a sample within the Petri dish.

2.2 Dot product

To compute the dot product between two vectors, we can either compute it by combining the components of the vectors, or by combining the magnitudes and angle. We can use this to look at the difference between a vector in the Petri dish and one that is outside the Petri dish. If two vectors appear similar, we should see a different result than we would for two very dissimilar vectors.

2.3 Standard deviation

The standard deviation is a measure that describes the dispersion of the data around the mean value. If we have a small standard deviation, it indicates that the data tend to be closer to the mean value, and a large standard deviation indicates data are more spread out. We can define a vector that is the average of a part of the vectors in the Petri dish and use it as a threshold to see, if there is a too large dispersion.

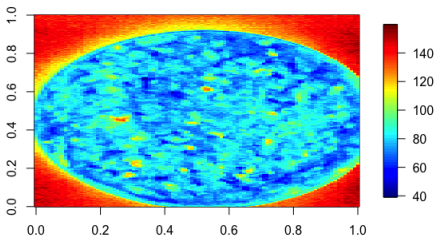
2.4 K-means

K-means is an unsupervised prototype clustering algorithm, where each cluster is defined by a prototype and it assigns objects to the cluster that are most similar to the prototype. The prototype is called centroid and they are the mean of the cluster. We have one parameter, k, that defines the number of clusters we are looking for. The k centroids are randomly selected, which are the starting point of the clusters, and then it performs iterations until the positions of the centroids are optimised.

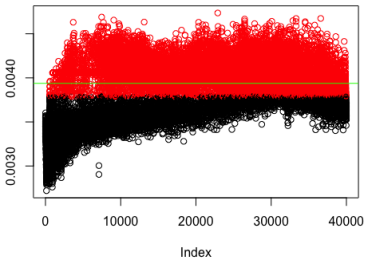
We will use the function `kmeans()`, which is standard in R. There exists different types of the algorithm, but we will use the default Hartigan-Wong algorithm (Hartigan & Wong, 1979). It begins by randomly selecting k centroids and it will for each data point assign them to their closest centroid. The points in each cluster will be averaged and the centroid is updated to their average. The data points will be reassigned to the closest centroid and the average of the data points in these new clusters will again update the centroids position. This will continue until the points does not change between clusters or the maximum number of iterations is reached.

2.4.1 K-means test

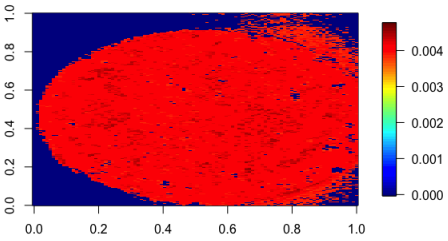
We need to check whether using k-means only works for the particular data set and we will test it with two extreme scenarios. We will test it by changing the amount of background surrounding the Petri dish and we will use an arbitrary data set from experiment 18. The test is based on the assumption that it would affect the clustering method if there were a smaller amount of background. Therefore, if we have more background, we would see a larger difference between the two clusters, where we will have one cluster with Petri dish data and one with background data. The first scenario can be seen in figure 2a and here we have removed the background on each side of the Petri dish.



(a) Image plot of the Petri dish at band 251.



(b) The clusters made with k-means. The green line are the mean of the data.

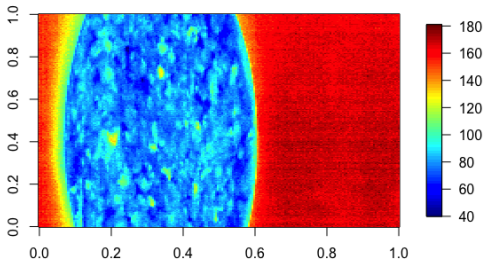


(c) Image plot of the Petri dish at band 251 based where the non-Petri dish part have been set to zero.

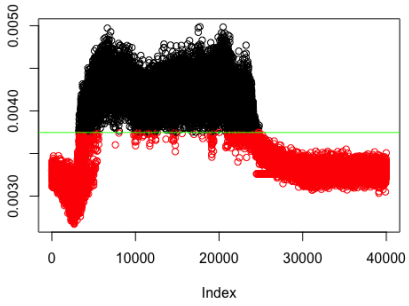
Figure 2: Result for k-means test when removing a lot of background.

Figure 2b shows the two clusters that are separated for the data and there is a green line to show the mean, which gives us an idea of how the separation would have been if we had used the mean instead of k-means. We can see that there is a significant difference and we would classify a large part of the data incorrectly if we had used mean. We got a satisfying separation of the data as we can see on figure 2c. Overall we can observe that it separates the Petri dish from the background, which could be sufficient enough for extracting a large square of the Petri dish without including the background.

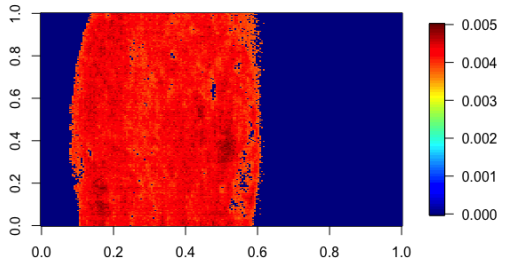
In the second scenario, we remove the top and bottom of the Petri dish, which gives us more background. We assume that this could affect the result of the clustering.



(a) Image plot of the Petri dish at band 251.



(b) The clusters made with k-means. The green line are the mean of the data.



(c) Image plot of the Petri dish at band 251 based where the non-Petri dish part have been set to zero.

Figure 3: Result for k-means test when allowing for a larger part to be background.

Figure 3b shows the result of the clustering and indicates how the clustering separates the Petri dish from the background very well. We can also note that we do not have a large part of the data exceeding the edge of the Petri dish as we did for the other scenario. This suggest that it is important to have a background that is significantly different in order to create the best separation of the data using k-means.

2.5 Masking

In the previous section we saw that we used k-means to separate the data into background and Petri dish. In this method we used masking to make this separation more clear and it can be seen in figure 2c and 3c. In the k-means test we use a binary mask and set the background to zero. In the final implementation we will use a mask of -1 in order to create an even greater difference between the Petri dish and the background. We set the Petri dish to 0.1 such that false positives do not pull the data in a wrong direction.

2.6 Maximum submatrix

We have until now been focusing on enhancing the difference between the Petri dish and the background. The reason for this is to find a section of the data of which we are sure to be within the borders of the Petri dish. To find this section after using the previous methods, we use the R function `maxsub2d()` to find the maximum submatrix (Borchers, 2018).

3 Analysis

The analysis techniques that we will use in this project to analyse the HSI data will be chemometric methods. Chemometrics is a collection of many analysis techniques and methods that can be applied to Food Science problem and they each have their own applications that they work best with. In this project, we will be focusing on HSI of meat and classification methods. When it comes to HSI, the most widely used method is PCA, which we will give a short introduction to in this project and we will focus on the less familiar method ICA(Marini, 2013). Even though, PCA is the cornerstone of chemometrics and it is a method you cannot avoid when examining multivariate data, PCA have some shortcomings that ICA have no problem with. PCA uses the correlation between the data elements and this have the advantage that the analysis can be based on second-order statistics solely(Hyvärinen *et al.* , 2001). Before explaining these drawback, we will first of all describe PCA briefly to get an idea of the method.

3.1 Principle Component Analysis

PCA is an unsupervised technique that decompose the data and its purpose is to reduce the dimension. The idea is that although all components are needed to reproduce a systems total variability, often we will only need a small number of them to account for the majority of the variability. The reason for using it for HSI is that HSI is based on the fact that neighbouring bands are highly correlated (Rodarmel & Shan, 2002)

The way PCA works is by decomposing the data and projecting it onto a space with a lower dimension, which will give us the principal components (PCs) consisting of loading and score vectors(Park & Lu, 2015). It selects the directions that have the largest variability. In other words, it keeps those with the highest amount of information.

The details behind how the method identifies these components are beyond the scope of this project, and it will only be subject of comparison to the ICA.

3.2 Independent Component Analysis

ICA is an unmixing method that can be used to isolate noise from images and audio or it can isolate individual signals from a mixture of signals. The best way to describe it is to use the Cocktail-party problem as an example. Let's say we are at a party and n people are talking together at the same time. If we had a microphone in the room, we would hear an overlapping of the speakers voices. Instead, we could place n microphones around the room and they will collect their own combination of the speakers in the room. With the ICA we would be able to separate the signals for each speaker. This problem is one that the ICA excels at compared to other component analyses such as PCA (Hyvärinen *et al.* , 2001).

In statistics and signal preprocessing, we often want to find a representation of the data and we will often look for a linear transformation of the data. For the ICA we want, as the name implies, to find a representation where the transformed components are as statistically independent from each other as possible (Hyvärinen, 1999). The reason why we want the components to be statistically independent and not just independent is that merely independence would imply that they are uncorrelated, however; if two values are uncorrelated, this does not imply independence. Therefore, using statistically independent variables gives us a stronger independence, which is essential to ICA (Hyvärinen *et al.* , 2001).

For a more precise mathematical definition, we observe n random variables $x_1, ..., x_n$, which are modelled as linear combinations of n random variables $s_1, ..., s_n$

$$x_i = a_{i1}s_1 + a_{i2}s_2 + ... + a_{in}s_n, \quad i = 1, ..., n \tag{3}$$

where $a_i j$, $i, j = 1, ..., n$ are some real coefficients that are the mixing weights. All the s_i are mutually statistically independent and they are the ICs. This gives us the basic ICA model. We only have the observed variables x_i and we will use them to determine the mixing coefficients and the ICs.

We change the previous notation and use vector-matrix notation. We let $\mathbf{x} = (x_1, ..., x_n)^T$, a random vector whose elements are the mixtures, and $\mathbf{s} = (s_1, ..., s_n)^T$, a random vector. We denote $\mathbf{A}$ as the matrix with elements a_{ij}. Using this notation, our mixing model is

written

$$\mathbf{x} = \mathbf{As} \tag{4}$$

x is the vector of the observed data (signals), **A** is the matrix with the mixing coefficients and **s** is the vector of the source signals. ICA finds a separating matrix **W** such that

$$\mathbf{y} = \mathbf{Wx} = \mathbf{WAs} \tag{5}$$

where **y** is the vector of independent components (Villa *et al.* , 2009).

This is the most basic ICA model and to ensure that it can be estimated, we will need some restrictions. The first is that the ICs are assumed statistically independent. The ICA rests on this principle, which is what makes it possible as we described earlier. The second is that the ICs must have a non-Gaussian distribution. This is a more relaxed restriction since we can have a few components that are Gaussian without disturbing the analysis. However, we will see that since the the original sources often are non-Gaussian, the restriction will not be a problem. It is one of the ways the ICA distinguishes from PCA and other multivariate statistical methods. It actually connects with the statistical independence restriction since we want it maximised for the estimated components, which can be obtained by the maximisation of non-Gaussianity (Marini, 2013).

An algorithm that uses this, which is also the one we employ for our analysis, is the fastICA algorithm. It is based on the Central Limit Theorem, and in general the measured signals will be more Gaussian than the source components. In order to extract the sources, id est the ICs, we want to find an unmixing matrix that maximises the sources non-Gaussianity (Marchini *et al.* , 2017). The non-Gaussianity is measured with approximations to neg-entropy. The algorithm is defined in R as a package and it makes it quite simple to perform the analysis. The difficulty comes when we have to specify the number of components. In the PCA there are simple methods for choosing the number of relevant components like a scree plot.

The fastICA starts by centering the data and whitening it by projecting it onto the directions of its PCs. This means that we can actually find the components for PCA with the result from this algorithm. After centering and whitening, the algorithm estimates a matrix **W** that is chosen to maximise the neg-entropy. It will then attempt to find the

interesting directions in the data set and these are the components that show the least Gaussian distribution. From the algorithm we get the signals separated, however; not it any specific order. The fact that the ordering of the components are not determined is a drawback for the ICA compared to PCA since we cannot be sure what each component represents. The code can be found in appendix A.4.

4 Results

We have applied the component analysis on two types of data: minced beef and pork chops. The main focus has been on the minced beef, which is also why the preprocessing method is only used on the minced beef data. The pork chop data have an initial data set at 0 hours, where there the bacteria have been applied onto the meat together with a control without bacteria. After 24 hours the same pork chops are photographed again. The minced beef data contains 50-70 data sets with varying rotation, bacteria count and incubation time. The bacteria is also seeded onto the meat in three ways. Either by streaking bacteria on the surface, by placing droplets with an increasing amount of bacteria or by placing a concentrated pellet of bacteria at a defined spot. The second method is referred as position box. The bacteria plated in each type of experiment also differs. The bacteria for the minced beef are diluted and have more fluid. For the pork chops the bacteria have been pelleted by centrifuged, and therefore comes in a larger concentration.

4.1 Minced beef

The minced beef is flattened in the Petri dish as we can see on figure 4 and we do not have any gap between the beef and the edge of the Petri dish. Therefore, the characteristic form of the minced beef will not appear in the images.

Figure 4: Minced beef before (left) and after it is flatted (right) in the Petri dish.

In the minced beef experiments we also have a significantly larger amount of data given that we have between 60-70 data sets. It is therefore time consuming to be looking

at each data set individually and checking if we will need to remove some of the background if it is interfering with the analysis. That is why we have a data preprocessing step and a step where we isolate a section of the data sets. In the experiments that involve rotation of the Petri dish, it will be harder to remove the background as it does not have a simple uniform background. On figure 5, we can see grey tone images of the experiments. Here we can observe that for experiment 19, there is attached a device that is used to rotate the Petri dish and it gives us an non-uniform background to sort out compared to the background in experiment 18.

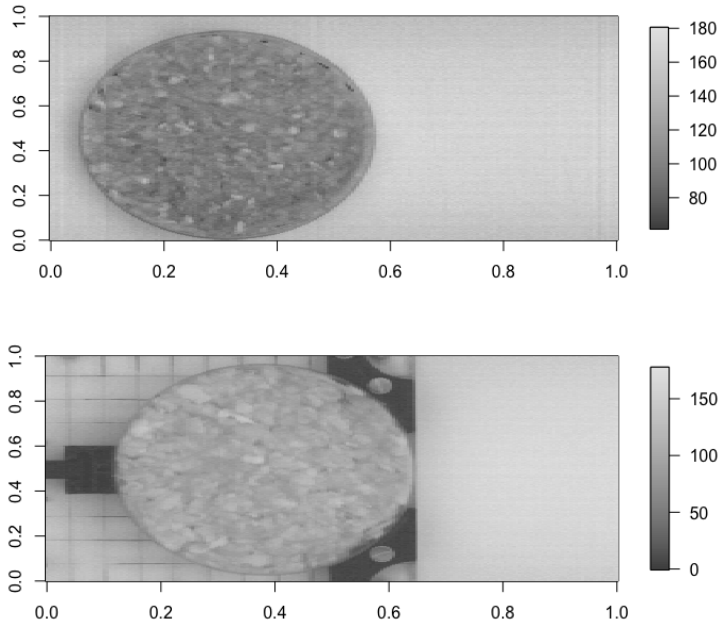


Figure 5: Grey tone image of experiment 18 (top) and experiment 19 (bottom).

4.1.1 Data preprocessing results

We already saw partially a result for the preprocessing in section 2.4.1 where we looked at the test for k-means. Here we could see how well the background were sorted even though we manipulated with the amount of background there were around the Petri dish. On figure 6, we can see the preprocessing performed on data from experiment 18. After setting the background to -1, we use `maxsub2d()` and average the finding

of all the 60-70 data sets to find the section that we will isolate and use for analysis. Since we have a large amount of data sets, we only show the preprocessing result for two randomly picked data sets. For experiment 18, we can see that we are able to remove all of the background and this makes it easier to find a maximum matrix that are within the Petri dish. Experiment 19, shows that we do not have a distinction between the Petri dish and the background to the same degree as in experiment 18. If we compare it to the grey tone image in figure 5, we can see that the part not chosen to be background is the device attached to the Petri dish.

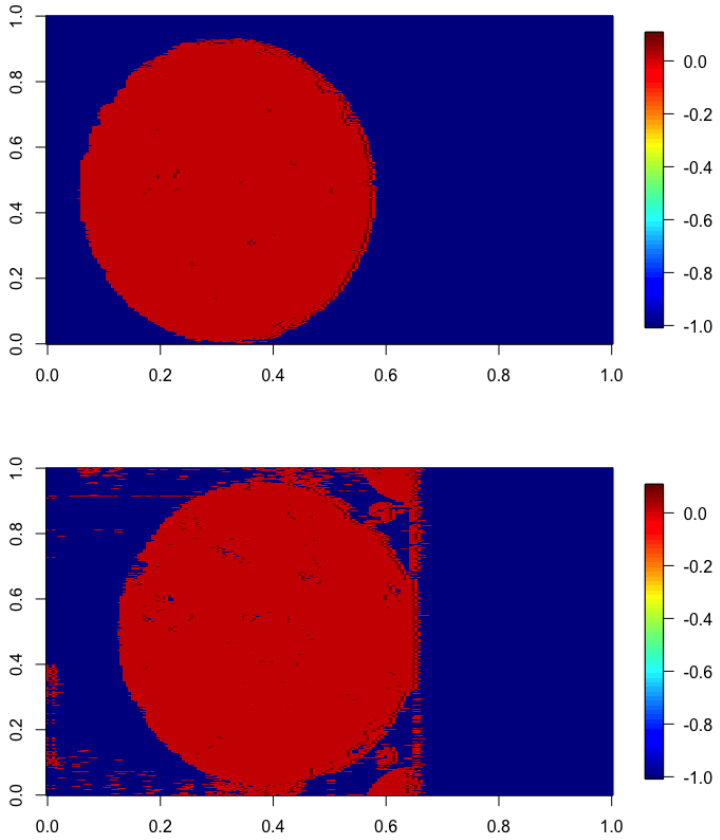


Figure 6: Preprocessing result for experiment 18 (top) and experiment 19 (bottom).

4.1.2 Analysis results

After finding the sections of Petri dish without the background, we are able to perform the ICA. We will be looking at four sections chosen at random where two images are taken after 24 hours and two after 48 hours. We start by looking at the images on figure 7, where we can observe at lot of patterns that we cannot clearly identify as bacteria or beef patterns, so; we cannot see if a specific component contain the bacteria. However, we can observe a connection between component 1 and the grey tone image. We can observe that the grey tone have lighter spots, which could be fat, and on the same spots in component 1 we observe low values. Similarly, we can observe that the darker spots have a higher value on component 1. There exists similar connections for the rest of the experiments as seen in appendix B.

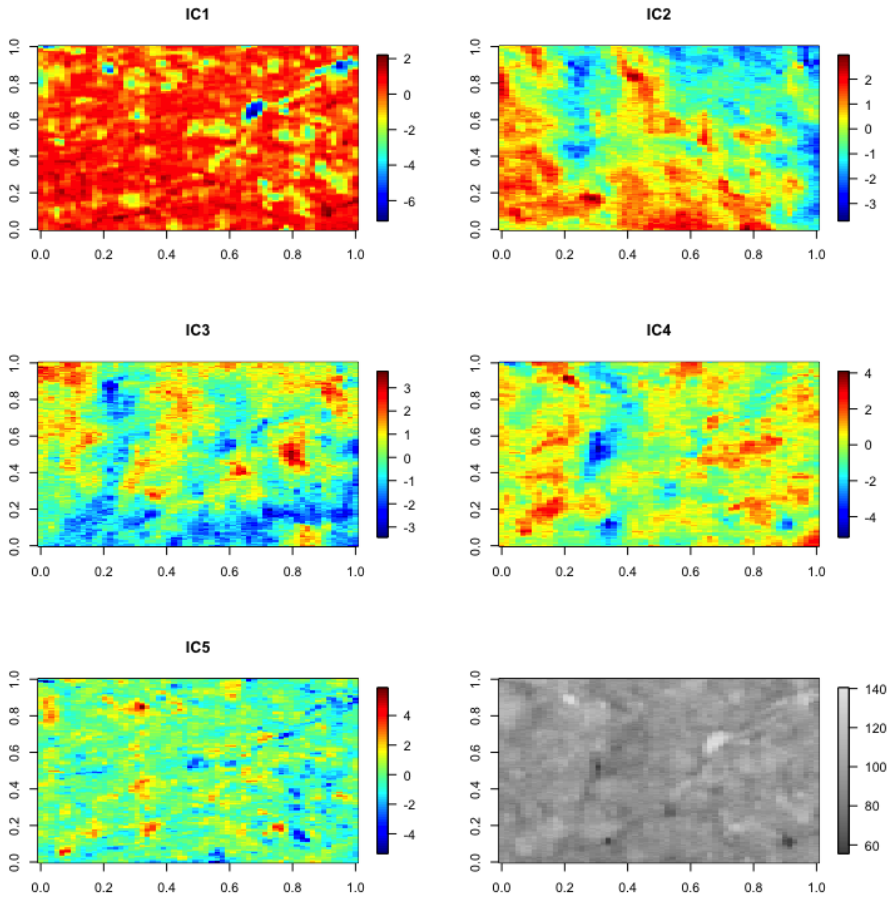


Figure 7: Caption

4.2 Pork chops

The pork chop data have three data sets for each time mark (0 h or 24 h) where the amount of bacteria placed on the pork chop varies for each data set. The background is not removed for the pork chop, except for the scale, which we can see on figure 8. We can clearly see that the scales affects the image plot and it would be probable that it also affects the analysis.

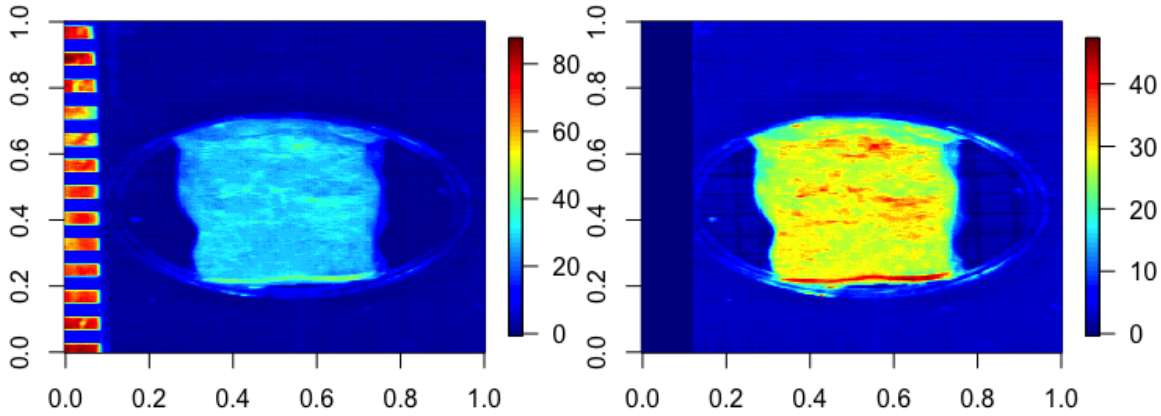


Figure 8: Comparison between an image plot with the scale (left) and an image plot (right) with the scale removed.

On the grey tone image, we can see what the pork chop looks like without any analysis performed on the data. We have a grey tone image of each pork chop to use for comparison between the ICs and they can be seen in appendix C.

4.3 Analysis results

We apply the ICA directly on the data sets after removing the measure and will not isolate a section only with pork. The results of the analysis can be found in appendix C for all of the pork chop data. In figure 9, we cannot observe a component that contains bacteria. We can, however, still observe certain patterns in the components. Component 1 appears to be the fat side of the pork chop and reflections of light in the Petri dish. For the same pork chop 24 hours later, seen on figure 10, we can observe a difference in the pork chop in component 2, which could indicate bacterial growth. For the other pork chop experiments we can observe the same relation. At 0 hours we do not detect anything significant and after 24 hours we detect a difference that appear to be bacteria.

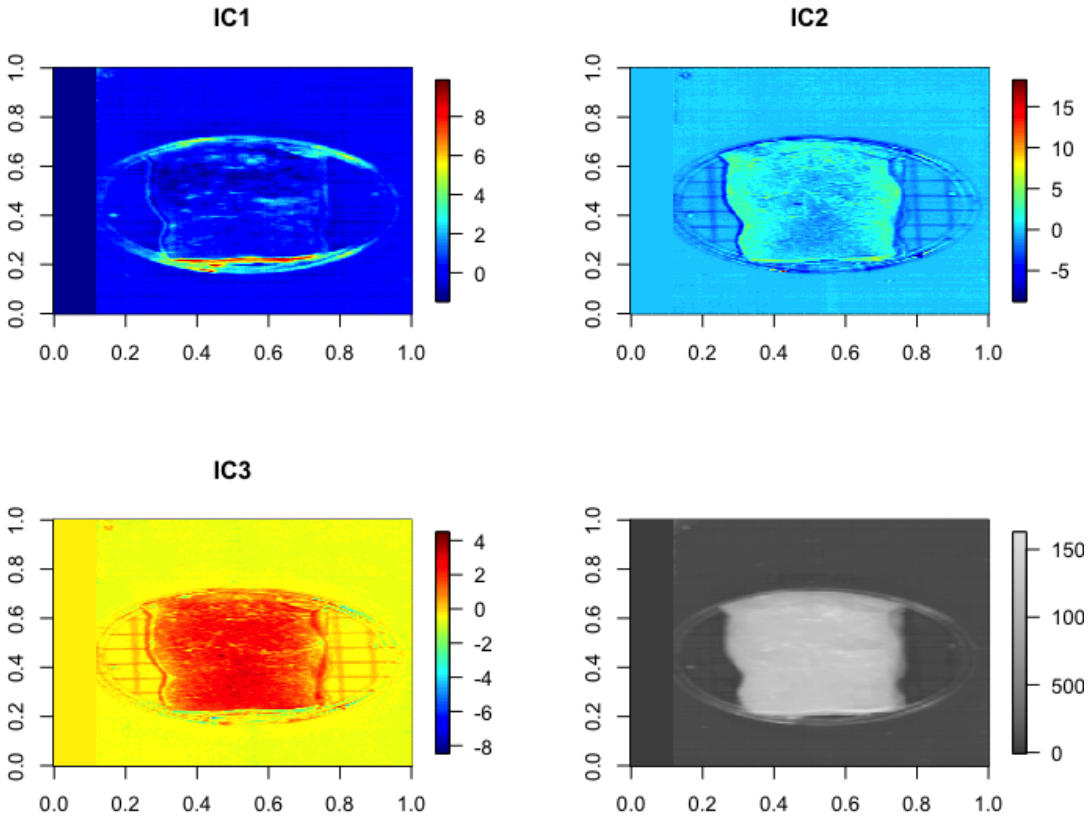


Figure 9: ICA of pork chop after bacteria have just been plated.

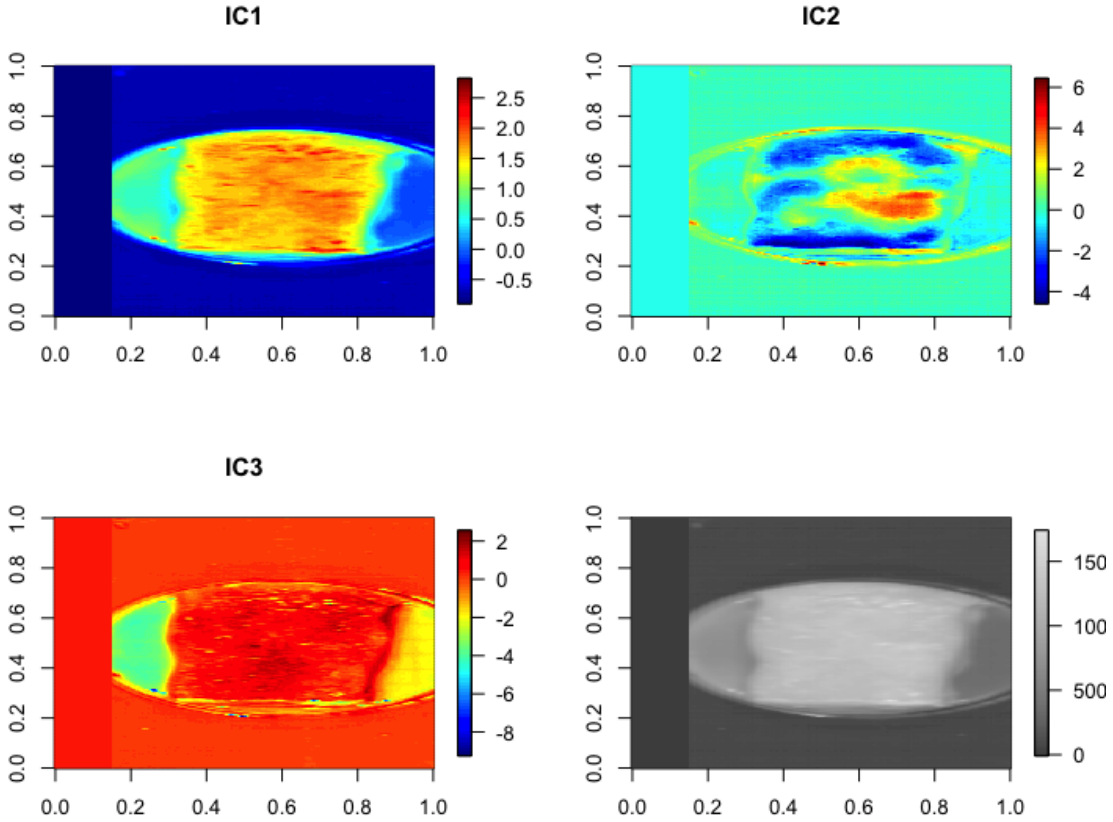


Figure 10: ICA of pork chop after 24 hour incubation.

5 Discussion

On figure 6, we observed that the preprocessing method handles the data for the experiment 18 better than for experiment 19. On the figure, we can see what appears to be the device used to rotate the Petri dish and it gives the data set a non-uniform background. This background does not affect the ability to find sections within the border of the Petri dish as we can see on the figures in appendix B.2 and B.3 given that we do not observe anything outside the Petri dish. The reason for this can be attributed to the use of sum of squares before normalising the data given, it did have a great impact in removing a lot of the differences between the Petri dish and the more similar parts of the background.

The most striking result to emerge from our analysis is that we could detect bacteria in the pork chops data set where the incubation time was 24 hours. This might not be satisfying for real-time prediction, though we do not expect the bacteria to need incubation time before detection. This result is further strengthened by all of the pork chop data sets displaying the same satisfying result. The pork chop results seems to confirm that the ICA can divide the information into several components each containing their own feature. Thus, we have reduced the dimension from the number of bands. which is 216 for pork chop data and 500 for minced beef data, to a single dimension where all of the information is contained.

Contrary to these results and our expectations, we did not find a similar result for the minced beef, nonetheless; we were still able to observe some patterns indicating that it is not a random result. We discovered this because of another example, where HSI is used in recovering text from old books, where text was otherwise undetectable with the naked eye (Holck, 2018). This result can be found in appendix D and in the figure we can see that there is a hint of lines of text that we cannot observe on the grey tone image. We can see these lines because it is easier for us to understand text rather than bacteria. This shows us that we need to look at patterns to understand the result. The patterns found for the minced beef is to hard to interpret unlike the pork chops data. Perhaps, we could look at the samples in the same way as for the pork chops and not remove the background. We could also look into the method used to plate the bacteria onto the samples. It is also plausible that by evenly streaking the bacteria on the surface we might

make the data more Gaussian, which is a restriction for the ICA, and it could be a reason for the method not being successful.

We did not see any usable results from the minced beef, so; it could be interesting to compare the two experimental setups and look at the differences. An obvious difference is the preprocessing. The pork chop experiments does not have any preprocessing before analysis, which might be the problem for the minced beef results, and we could be isolating a too small section of the data set. We are perhaps looking very closely at a square full of bacteria and cannot detect it with ICA, because it is distributed evenly in the section.

The method for plating out the bacteria matches with this observation, since for the minced beef we either streak the bacteria on the surface or place droplets on the entire surface. Contrary to this, the bacteria in the pork experiments were centrifuged into pellets, which is not plated out on the entire surface. When putting the two methods up against each other in this manner, we can also speculate if the bacteria for the minced beef exhibit a stronger tendency for random walks, since bacteria is in a fluid and it is more distributed on the entire surface whereas the pork chops bacteria would not do random walks and thus stay in the place, where it was plated. The random walks would contribute to the data being more Gaussian which is not something we want when doing an ICA.

Therefore, we could look at a new experimental setup where the bacteria for the minced beef is centrifuged and place in a single location rather than plated out on the entire surface. This would make it more about locating the bacteria in a components and we would be able to avoid Gaussian data. If we understand how the bacteria moves then we will also be able to develop our method further.

6 Conclusion

The scope of this thesis was to do investigate some ground work for further study and be able to detect bacteria in meat using ICA and HSI in order to create a method for faster non-destructive detection. Our work has led to the conclusion that this is indeed possible, but it will need further investigation of the subject.

We obtained satisfactory results for the experiments with pork chops and considerable progress have been made with respect to detection when using the experimental procedure from the pork chops experiments. However, the same results were not found for the minced beef and action must be taken in order to find a procedure such that we can obtain similar results for all types of retail meat. Even though our work show limitations on this aspects, it still entails promising possibilities to be explored through further studies. We can see that any further research should look into the differences between the procedures and use it to improve the minced beef data in further studies.

7 Future perspectives

The results in this project can be used for other analyses, where we can apply the ICs containing bacteria as training data for ANN. The results can also be used as a starting point for further investigations in another project. Here both the positive and negative results can lead to a better understanding of how the bacteria can be detected in meat. The positive results can be used as a measure for bacterial growth and the procedure for the experiment can be used to improve the negative results.

The next steps to take would be to use other methods on the minced beef data and try to detect bacteria in another way. This way we might gain a better understanding of the results for the minced beef. We will also need to investigate the issue raised by Gaussian data. We could expect that if we increase the incubation for the pork chop data to 48 hours we would also see that the bacteria would spread more.

Looking into the theory surrounding motility of the bacterial strains used could also be important, in order to understand how the bacteria behaves, and therefore get a better understanding of how the chemometrics should be applied. This could help us understand how the spectral images could be employed to look under the surface of the meat. This also gives us an opportunity to look into the experimental procedures and create an experiment where the bacteria are plated under the surface of the meat.

We can expand the ICA to include PCA as a whitening tool that we apply prior to ICA. Perhaps we would be able to look at the minced beef data without changing the experimental procedure. It would also be relevant to look at other methods for our data rather than focusing on a single method.

References

Borchers, Hans Werner. 2018. *adagio: Discrete and Global Optimization Routines*. R package version 0.7.1, `https://CRAN.R-project.org/package=adagio`.

Efenberger-Szmechtyk, Magdalena, Nowak, Agnieszka, & Kregiel, Dorota. 2018. Implementation of chemometrics in quality evaluation of food and beverages. *Critical reviews in food science and nutrition*, **58**(10), 1747–1766.

Feng, Yao-Ze, & Sun, Da-Wen. 2014. "Seeing the bacteria": hyperspectral imaging for bacterial prediction and visualisation on chicken meat. *NIR news*, **25**(7), 4–6.

Feng, Yao-Ze, ElMasry, Gamal, Sun, Da-Wen, Scannell, Amalia GM, Walsh, Des, & Morcy, Noha. 2013. Near-infrared hyperspectral imaging and partial least squares regression for rapid and reagentless determination of Enterobacteriaceae on chicken fillets. *Food Chemistry*, **138**(2-3), 1829–1836.

Hartigan, John A, & Wong, Manchek A. 1979. Algorithm AS 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, **28**(1), 100–108.

Holck, Jakob Povl. 2018. Nye teknologier og Herlufsholm-samlingen. Det skjulte bibliotek kommer for en dag. *Bibliotekshistorie*, **13**, 38–66.

Huang, Lin, Zhao, Jiewen, Chen, Quansheng, & Zhang, Yanhua. 2013. Rapid detection of total viable count (TVC) in pork meat by hyperspectral imaging. *Food Research International*, **54**(1), 821–828.

Hyvärinen, Aapo. 1999. Fast and robust fixed-point algorithms for independent component analysis. *IEEE transactions on Neural Networks*, **10**(3), 626–634.

Hyvärinen, Aapo, Karhunen, Juha, & Oja, Erkki. 2001. *Independent Component Analysis*. John Wiley & Sons.

Marchini, J L, Heaton, C, & Ripley, B D. 2017. *fastICA: FastICA Algorithms to Perform ICA and Projection Pursuit*. R package version 1.2-1, `https://CRAN.R-project.org/package=fastICA`.

Marini, Federico. 2013. *Chemometrics in food chemistry*. Vol. 28. Newnes.

Park, Bosoon, & Lu, Renfu. 2015. *Hyperspectral imaging technology in food and agriculture*. Springer.

Pozo, Veronica F, & Schroeder, Ted C. 2015. Costs of Meat and Poultry Recalls to Food Firms. *Finance and Economics*.

Ritchie, Hannah, & Roser, Max. 2018. *Meat and Seafood Production & Consumption*. Published online at OurWorldInData.org. Retrieved from: `https://ourworldindata.org/meat-and-seafood-production-consumption` [Online Resource].

Rodarmel, Craig, & Shan, Jie. 2002. Principal component analysis for hyperspectral image classification. *Surveying and Land Information Science*, **62**(2), 115–122.

Sendin, Kate, Williams, Paul J, & Manley, Marena. 2018. Near infrared hyperspectral imaging in quality and safety evaluation of cereals. *Critical reviews in food science and nutrition*, **58**(4), 575–590.

Villa, Alberto, Chanussot, Jocelyn, Jutten, Christian, Benediktsson, Jon Atli, & Moussaoui, Saïd. 2009. On the use of ICA for hyperspectral image analysis. *Pages IV–97 of: Geoscience and Remote Sensing Symposium, 2009 IEEE International, IGARSS 2009*, vol. 4. IEEE.

Wang, Wei, ZHANG, Xiao-li, *et al.* . 2010. Study on modeling method of total viable count of fresh pork meat based on hyperspectral imaging system. *Spectroscopy and Spectral Analysis*, **30**(2), 411–415.

Zheng, Xiaochun, Peng, Yankun, & Wang, Wenxiu. 2017. A nondestructive real-time detection method of total viable count in pork by hyperspectral imaging technique. *Applied Sciences*, **7**(3), 213.

A Code

A.1 Data preprocessing - Find Section

```
###PACKAGES###
library(fields)
library(ggplot2)
library(adagio)
####

# Sort data sets
files <- list.files(pattern="*.csv")
subMinds <- c()

time0 <- grep("Ti0", files)
time1440 <- grep("Ti1440", files)
time2880 <- grep("Ti2880", files)
files0 <- files[time0]
files0_ordered <- files0[order(as.numeric(gsub("[^0-9]+", "",
   files0)))]
files1440 <- files[time1440]
files1440_ordered <- files1440[order(as.numeric(gsub("[^0-9]+", ""
   , files1440)))]
files2880 <- files[time2880]
files2880_ordered <- files2880[order(as.numeric(gsub("[^0-9]+", ""
   , files2880)))]

files_sorted <- c()
files_sorted <- append(files_sorted, files0_ordered)
files_sorted <- append(files_sorted, files1440_ordered)
files_sorted <- append(files_sorted, files2880_ordered)

####INPUT DATA
```

```
27 for(d in 1:length(files_sorted)){
28   dataex18 <- read.csv(file=files_sorted[d], header=F, sep=",")
29   data <- as.matrix(dataex18)
30
31   ## Change matrix (200x500)x400 to array 200x400x500
32   data.arr <- array(0, dim = c(200,400,500))
33   r1 <- 1
34   r2 <- 200
35   for(h in 1:500){
36     data.arr[,,h] <- data[r1:r2,]
37     r1 <- r1 + 200
38     r2 <- r2 + 200
39   }
40
41   # Make dataM.arr into a 80.000 x 500 matrix where each row is h
       in dataM.arr
42   data.mat <- matrix(0,80000,500)
43   i = 1
44   while(i < 80000){
45     for(r in 1:200){
46       for(c in 1:400){
47         data.mat[i,] <- data.arr[r,c,]
48         i <- i + 1
49       }
50     }
51   }
52
53   # Remove lines with sum of squares before normalizing
54   ss.mean <- mean(data.mat)
55   ss.data.mat <- c()
56   for(i in 1:80000){
57     ss.data.mat[i] <- sum((data.mat[i,]-ss.mean)^2)
58   }
```

```
59  avr.ss <- mean(ss.data.mat[50000:50100]) # area with meat
60  indices3 <- c() # A vector with the indices that are meat
61  for(i in 1:80000){
62    if(ss.data.mat[i] < avr.ss){
63      indices3 <- rbind(indices3,i)
64    }
65  }
66
67  ###Normalizing input data
68  dataT <- data.mat
69  dataT.rowM <- norm(dataT)
70  dataT.n <- matrix(0,80000,500) #normalized dataT
71  for(i in 1:80000){
72    dataT.n[i,] <- dataT[i,]/norm(dataT[i,])
73  }
74
75  dataT <- dataT.n # set dataT to the normalized dataT
76
77  ### Remove lines
78
79  # 1. Removing lines with dot product
80  avr.vec <- colMeans(dataT[50000:50100,]) # area with meat
81  th.dot <- avr.vec%*%dataT[50000,]
82  indices1 <- c() # A vector with the indices that meat
83  for(i in 1:80000){
84    if(dataT[i,]%*%avr.vec < th.dot){
85      indices1 <- rbind(indices1,i)
86    }
87  }
88
89  # 2. Removing lines with standard deviation
90  sd.dataT <- c()
91  for(i in 1:80000){
```

```
92    sd.dataT[i] <- sd(dataT[i,])
93  }
94  avr.sd <- mean(sd.dataT[50000:50100]) # area with meat
95  indices2 <- c() # A vector with the indices that are meat
96  for(i in 1:80000){
97    if(sd.dataT[i] < avr.sd){
98      indices2 <- rbind(indices2,i)
99    }
100 }
101
102 # Make new dataset that have the rows that are meat
103 U.indices <- union(indices1,indices2)
104 U.indices <- union(U.indices,indices3)
105
106 # Matrix with the mean spektrum for the last 10.000 rows
107 dataM <- matrix(colMeans(dataT[70000:80000,]),nrow=80000,ncol
      =500,byrow=TRUE)
108 dataM[U.indices,] <- dataT[U.indices,]
109
110 # Make dataM into an r x c x h array where each h is a row in
      dataM
111 dataM.arr <- array(0, dim = c(200,400,500))
112 i <- 1
113 r <- 0
114 while(i < 80000){
115   r <- r + 1
116   for(c in 1:400){
117     dataM.arr[r,c,] <- dataM[i,]
118     i <- i + 1
119   }
120 }
121
```

```
122 # Make dataM.arr into a 80.000 x 500 matrix where each row is h
      in dataM.arr
123 dataM.mat <- matrix(0,80000,500)
124 i = 1
125 while(i < 80000){
126   for(r in 1:200){
127     for(c in 1:400){
128       dataM.mat[i,] <- dataM.arr[r,c,]
129       i <- i + 1
130     }
131   }
132 }
133
134 # Determine which wavelength to look at
135 dataM.arr.mean <- matrix(0,500,1)
136 for(i in 1:500){
137   dataM.arr.mean[i] <- mean(dataM.arr[,,i])
138 }
139 a <- which.max(dataM.arr.mean)
140
141 ## Clustering method
142 clusters <- kmeans(dataM.mat[,a], 2)
143 dataM.clust <- matrix(0,80000,500)
144 clust.th <- clusters$cluster[80000]
145 for(i in 1:80000){
146   if(clusters$cluster[i] != clust.th){
147     dataM.clust[i,] <- dataM[i,]
148   }
149 }
150
151
152 # Make dataM into an r x c x h array where each h is a row in
      dataM.clust
```

```
153 clust.arr <- array(0, dim = c(200,400,500))
154 i <- 1
155 j <- 0
156 while(i < 80000){
157   j <- j + 1
158   for(c in 1:400){
159     clust.arr[j,c,] <- dataM.clust[i,]
160     i <- i + 1
161   }
162 }
163
164 # Isolate the petri dish, dataM.clust, using the largest
      submatrix
165 clust.arr.temp <- clust.arr[,,a]
166 for(r in 1:200){
167   for(c in 1:400){
168     if(r < 3){
169       if(clust.arr[r,c,a] == 0){
170         clust.arr.temp[r,c] <- -1
171       }
172     }else{
173       if(clust.arr[r,c,a] == 0 && clust.arr[r-1,c,a] == 0 && clust
            .arr[r-2,c,a] == 0){
174         clust.arr.temp[r,c] <- -1
175       }
176       if(clust.arr[r,c,a] == 0 && clust.arr[r-1,c,a] != 0 && clust
            .arr[r-2,c,a] != 0){
177         clust.arr.temp[r,c] <- 0.1
178       }
179     }
180   }
181 }
182 subM <- maxsub2d(clust.arr.temp)
```

```
183   subMinds <- rbind(subMinds, subM$inds)
184 }
185
186 ind1 <- floor(mean(subMinds[,1]))
187 ind2 <- floor(mean(subMinds[,3]))
```

A.2 Data preprocessing - Create Sections

```
###PACKAGES###
library(fields)
library(ggplot2)
library(adagio)
####

# Sort data sets
files <- list.files(pattern="*.csv")
subMinds <- c()

time0 <- grep("Ti0", files)
time1440 <- grep("Ti1440", files)
time2880 <- grep("Ti2880", files)
files0 <- files[time0]
files0_ordered <- files0[order(as.numeric(gsub("[^0-9]+", "",
   files0)))]
files1440 <- files[time1440]
files1440_ordered <- files1440[order(as.numeric(gsub("[^0-9]+", ""
   , files1440)))]
files2880 <- files[time2880]
files2880_ordered <- files2880[order(as.numeric(gsub("[^0-9]+", ""
   , files2880)))]

files_sorted <- c()
files_sorted <- append(files_sorted, files0_ordered)
files_sorted <- append(files_sorted, files1440_ordered)
files_sorted <- append(files_sorted, files2880_ordered)

####INPUT DATA
for(d in 1:length(files_sorted)){
  dataex18 <- read.csv(file=files_sorted[d], header=F, sep=",")
```

```
29  data <- as.matrix(dataex18)
30
31  ## Change matrix (200x500)x400 to array
32  data.arr <- array(0, dim = c(200,400,500))
33  r1 <- 1
34  r2 <- 200
35  for(h in 1:500){
36    data.arr[,,h] <- data[r1:r2,]
37    r1 <- r1 + 200
38    r2 <- r2 + 200
39  }
40
41  # Make dataM.arr into a 80.000 x 500 matrix where each row is h
      in dataM.arr
42  data.mat <- matrix(0,80000,500)
43  i = 1
44  while(i < 80000){
45    for(r in 1:200){
46      for(c in 1:400){
47        data.mat[i,] <- data.arr[r,c,]
48        i <- i + 1
49      }
50    }
51  }
52  data.mat
53  clust.arr <- array(0, dim = c(200,400,500))
54  i <- 1
55  j <- 0
56  while(i < 80000){
57    j <- j + 1
58    for(c in 1:400){
59      clust.arr[j,c,] <- data.mat[i,]
60      i <- i + 1
```

```
61      }
62    }
63
64    #Isolate a section of the petri dish based on the average subM
          indices found:
65    clust.arr.sec <- clust.arr[(ind1+5):(ind1+64),(ind2+5):(ind2
          +125),]
66    assign(paste0("data.sec", d), clust.arr.sec)
67  }
```

A.3 K-means test

```
###PACKAGES###
library(fields)
library(ggplot2)
library(adagio)
####

####INPUT DATA
dataex18 <- read.csv("Ti2880Te20An0Am20W400H200B500Hour1006.csv",
   header=F, sep=",")
data <- as.matrix(dataex18)

#####TEST 1

## Change matrix (200x500)x400 to matrix 80.000x500
data.arr <- array(0, dim = c(200,400,500))
r1 <- 1
r2 <- 200
for(h in 1:500){
  data.arr[,,h] <- data[r1:r2,]
  r1 <- r1 + 200
  r2 <- r2 + 200
}

data.arr <- data.arr[16:115,,]
image.plot(data.arr[,,251])

# Make dataM.arr into a 80.000 x 500 matrix where each row is h in
    dataM.arr
data.mat <- matrix(0,dim(data.arr)[1]*dim(data.arr)[2],500)
i = 1
while(i < dim(data.arr)[1]*dim(data.arr)[2]){
```

```
30   for(r in 1:dim(data.arr)[1]){
31     for(c in 1:dim(data.arr)[2]){
32       data.mat[i,] <- data.arr[r,c,]
33       i <- i + 1
34     }
35   }
36 }
37
38 ###Normalizing input data
39 dataT <- data.mat
40 dataT.rowM <- norm(dataT)
41 dataT.n <- matrix(0,dim(data.mat)[1],500) #normalized dataT
42 for(i in 1:dim(data.mat)[1]){
43   dataT.n[i,] <- dataT[i,]/norm(dataT[i,])
44 }
45
46 dataT <- dataT.n # set dataT to the normalized dataT
47
48 ### Remove lines
49
50 # 1. Removing lines with dot product
51 avr.vec <- colMeans(dataT[30000:30100,]) # area with meat
52 th.dot <- avr.vec%*%dataT[30000,]
53 indices1 <- c() # A vector with the indices that meat
54 for(i in 1:dim(data.mat)[1]){
55   if(dataT[i,]%*%avr.vec < th.dot){
56     indices1 <- rbind(indices1,i)
57   }
58 }
59
60 # 2. Removing lines with standard deviation
61 sd.dataT <- c()
62 for(i in 1:dim(data.mat)[1]){
```

```
63    sd.dataT[i] <- sd(dataT[i,])
64  }
65  avr.sd <- mean(sd.dataT[30000:30100]) # area with meat
66  indices2 <- c() # A vector with the indices that are meat
67  for(i in 1:dim(data.mat)[1]){
68    if(sd.dataT[i] < avr.sd){
69      indices2 <- rbind(indices2,i)
70    }
71  }
72
73  # Make new dataset that have the rows that are meat
74  U.indices <- union(indices1,indices2)
75
76  # Matrix with the mean spektrum for the last 10.000 rows
77  dataM <- matrix(colMeans(dataT[30000:40000,]),nrow=40000,ncol=500,
      byrow=TRUE)
78  dataM[U.indices,] <- dataT[U.indices,]
79
80  # Make dataM into an r x c x h array where each h is a row in
      dataM
81  dataM.arr <- array(0, dim = c(dim(data.arr)[1],dim(data.arr)
      [2],500))
82  i <- 1
83  r <- 0
84  while(i < dim(data.mat)[1]){
85    r <- r + 1
86    for(c in 1:dim(data.arr)[2]){
87      dataM.arr[r,c,] <- dataM[i,]
88      i <- i + 1
89    }
90  }
91
```

```
92  # Make dataM.arr into a 80.000 x 500 matrix where each row is h in
        dataM.arr
93  dataM.mat <- matrix(0,dim(data.mat)[1],500)
94  i = 1
95  while(i < dim(data.mat)[1]){
96    for(r in 1:dim(data.arr)[1]){
97      for(c in 1:dim(data.arr)[2]){
98        dataM.mat[i,] <- dataM.arr[r,c,]
99        i <- i + 1
100     }
101   }
102 }
103
104 # Determine which wavelength to look at
105 dataM.arr.mean <- matrix(0,500,1)
106 for(i in 1:500){
107   dataM.arr.mean[i] <- mean(dataM.arr[,,i])
108 }
109 a <- which.max(dataM.arr.mean)
110
111 ## Clustering method
112 clusters <- kmeans(dataM.mat[,a], 2)
113 dataM.clust <- matrix(0,dim(data.mat)[1],500)
114 clust.th <- clusters$cluster[1]
115 for(i in 1:dim(data.mat)[1]){
116   if(clusters$cluster[i] != clust.th){
117     dataM.clust[i,] <- dataM[i,]
118   }
119 }
120
121 plot(dataM.mat[,a], col = clusters$cluster)
122 abline(mean(dataM.mat[,a]),0,col = "green")
123
```

```
124 # Make dataM into an r x c x h array where each h is a row in
       dataM.clust
125 clust.arr <- array(0, dim = c(dim(data.arr)[1],dim(data.arr)
       [2],500))
126 i <- 1
127 j <- 0
128 while(i < dim(data.mat)[1]){
129   j <- j + 1
130   for(c in 1:dim(data.arr)[2]){
131     clust.arr[j,c,] <- dataM.clust[i,]
132     i <- i + 1
133   }
134 }
135 image.plot(clust.arr[,,251])
136 clust.arr.temp <- clust.arr[,,a]
137 for(r in 1:200){
138   for(c in 1:400){
139     if(r < 3){
140       if(clust.arr[r,c,a] == 0){
141         clust.arr.temp[r,c] <- -1
142       }
143     }else{
144       if(clust.arr[r,c,a] == 0 && clust.arr[r-1,c,a] == 0 && clust.
           arr[r-2,c,a] == 0){
145         clust.arr.temp[r,c] <- -1
146       }
147       if(clust.arr[r,c,a] == 0 && clust.arr[r-1,c,a] != 0 && clust.
           arr[r-2,c,a] != 0){
148         clust.arr.temp[r,c] <- 0.1
149       }
150     }
151   }
152 }
```

```
153
154 rm(list=ls())
155
156 ####TEST 2
157 dataex18 <- read.csv("Ti2880Te20An0Am20W400H200B500Hour1006.csv",
       header=F, sep=",")
158 data <- as.matrix(dataex18)
159
160 ## Change matrix (200x500)x400 to matrix 80.000x500
161 data.arr <- array(0, dim = c(200,400,500))
162 r1 <- 1
163 r2 <- 200
164 for(h in 1:500){
165   data.arr[,,h] <- data[r1:r2,]
166   r1 <- r1 + 200
167   r2 <- r2 + 200
168 }
169
170 data.arr <- data.arr[,101:300,]
171 image.plot(data.arr[,,251])
172
173 # Make dataM.arr into a 80.000 x 500 matrix where each row is h in
       dataM.arr
174 data.mat <- matrix(0,dim(data.arr)[1]*dim(data.arr)[2],500)
175 i = 1
176 while(i < dim(data.arr)[1]*dim(data.arr)[2]){
177   for(r in 1:dim(data.arr)[1]){
178     for(c in 1:dim(data.arr)[2]){
179       data.mat[i,] <- data.arr[r,c,]
180       i <- i + 1
181     }
182   }
183 }
```

```
184
185 ###Normalizing input data
186 dataT <- data.mat
187 dataT.rowM <- norm(dataT)
188 dataT.n <- matrix(0,dim(data.mat)[1],500) #normalized dataT
189 for(i in 1:dim(data.mat)[1]){
190   dataT.n[i,] <- dataT[i,]/norm(dataT[i,])
191 }
192
193 dataT <- dataT.n # set dataT to the normalized dataT
194
195 ### Remove lines
196
197 # 1. Removing lines with dot product
198 avr.vec <- colMeans(dataT[30000:30100,]) # area with meat
199 th.dot <- avr.vec%*%dataT[30000,]
200 indices1 <- c() # A vector with the indices that meat
201 for(i in 1:dim(data.mat)[1]){
202   if(dataT[i,]%*%avr.vec < th.dot){
203     indices1 <- rbind(indices1,i)
204   }
205 }
206
207 # 2. Removing lines with standard deviation
208 sd.dataT <- c()
209 for(i in 1:dim(data.mat)[1]){
210   sd.dataT[i] <- sd(dataT[i,])
211 }
212 avr.sd <- mean(sd.dataT[30000:30100]) # area with meat
213 indices2 <- c() # A vector with the indices that are meat
214 for(i in 1:dim(data.mat)[1]){
215   if(sd.dataT[i] < avr.sd){
216     indices2 <- rbind(indices2,i)
```

```
217   }
218 }
219
220 # Make new dataset that have the rows that are meat
221 U.indices <- union(indices1,indices2)
222
223 # Matrix with the mean spektrum for the last 10.000 rows
224 dataM <- matrix(colMeans(dataT[30000:40000,]),nrow=40000,ncol=500,
      byrow=TRUE)
225 dataM[U.indices,] <- dataT[U.indices,]
226
227 # Make dataM into an r x c x h array where each h is a row in
      dataM
228 dataM.arr <- array(0, dim = c(dim(data.arr)[1],dim(data.arr)
      [2],500))
229 i <- 1
230 r <- 0
231 while(i < dim(data.mat)[1]){
232   r <- r + 1
233   for(c in 1:dim(data.arr)[2]){
234     dataM.arr[r,c,] <- dataM[i,]
235     i <- i + 1
236   }
237 }
238
239 # Make dataM.arr into a 80.000 x 500 matrix where each row is h in
       dataM.arr
240 dataM.mat <- matrix(0,dim(data.mat)[1],500)
241 i = 1
242 while(i < dim(data.mat)[1]){
243   for(r in 1:dim(data.arr)[1]){
244     for(c in 1:dim(data.arr)[2]){
245       dataM.mat[i,] <- dataM.arr[r,c,]
```

```
246       i <- i + 1
247     }
248   }
249 }
250
251 # Determine which wavelength to look at
252 dataM.arr.mean <- matrix(0,500,1)
253 for(i in 1:500){
254   dataM.arr.mean[i] <- mean(dataM.arr[,,i])
255 }
256 a <- which.max(dataM.arr.mean)
257 #image.plot(dataM.arr[,,a])
258
259 ## Clustering method
260 clusters <- kmeans(dataM.mat[,a], 2)
261 dataM.clust <- matrix(0,dim(data.mat)[1],500)
262 clust.th <- clusters$cluster[1]
263 for(i in 1:dim(data.mat)[1]){
264   if(clusters$cluster[i] != clust.th){
265     dataM.clust[i,] <- dataM[i,]
266   }
267 }
268
269 plot(dataM.mat[,a], col = clusters$cluster)
270 abline(mean(dataM.mat[,a]),0,col = "green")
271
272 # Make dataM into an r x c x h array where each h is a row in
      dataM.clust
273 clust.arr <- array(0, dim = c(dim(data.arr)[1],dim(data.arr)
      [2],500))
274 i <- 1
275 j <- 0
276 while(i < dim(data.mat)[1]){
```

```
277   j <- j + 1
278   for(c in 1:dim(data.arr)[2]){
279     clust.arr[j,c,] <- dataM.clust[i,]
280     i <- i + 1
281   }
282 }
283 image.plot(clust.arr[,,251])
```

A.4 Analysis code

```
###PACKAGES###
library(fields)
library(ggplot2)
library(adagio)
library(fastICA)
library(abind)
####

# Data Pork chops ----------

dataCube <- read.csv("hyperspectralDataCube4_24h.csv", header=F,
   sep=",")
dataCube <- dataCube[2:dim(dataCube)[1],2:216]
data <- as.matrix(dataCube)

rows <- 250


#Make matrix into summed array
data.arr <- array(0, dim = c(rows,400))
i <- 1
r <- 0
while(i < dim(dataCube)[1]){
  r <- r + 1
  for(c in 1:400){
    data.arr[r,c] <- sum(data[i,])
    i <- i + 1
  }
}

par(mfrow=c(1,2))
```

```
31 image.plot(data.arr, col = grey.colors(100))
32
33 #Make matrix into not summed array
34 data.arr1 <- array(0, dim = c(rows,400,215))
35 i <- 1
36 r <- 0
37 while(i < dim(dataCube)[1]){
38   r <- r + 1
39   for(c in 1:400){
40     data.arr1[r,c,] <- data[i,]
41     i <- i + 1
42   }
43 }
44 image.plot(data.arr1[,,50])
45 # Set measure part to zero (the black and white squares at the
     edge of the data)
46 data.arr1[1:30,,] <- 0
47 image.plot(data.arr1[,,50])
48
49 #Arr to mat
50 data.mat <- matrix(0,dim(dataCube)[1],215)
51 i = 1
52 while(i < dim(dataCube)[1]){
53   for(r in 1:rows){
54     for(c in 1:400){
55       data.mat[i,] <- data.arr1[r,c,]
56       i <- i + 1
57     }
58   }
59 }
60
61
62 #Make new matrix into summed array
```

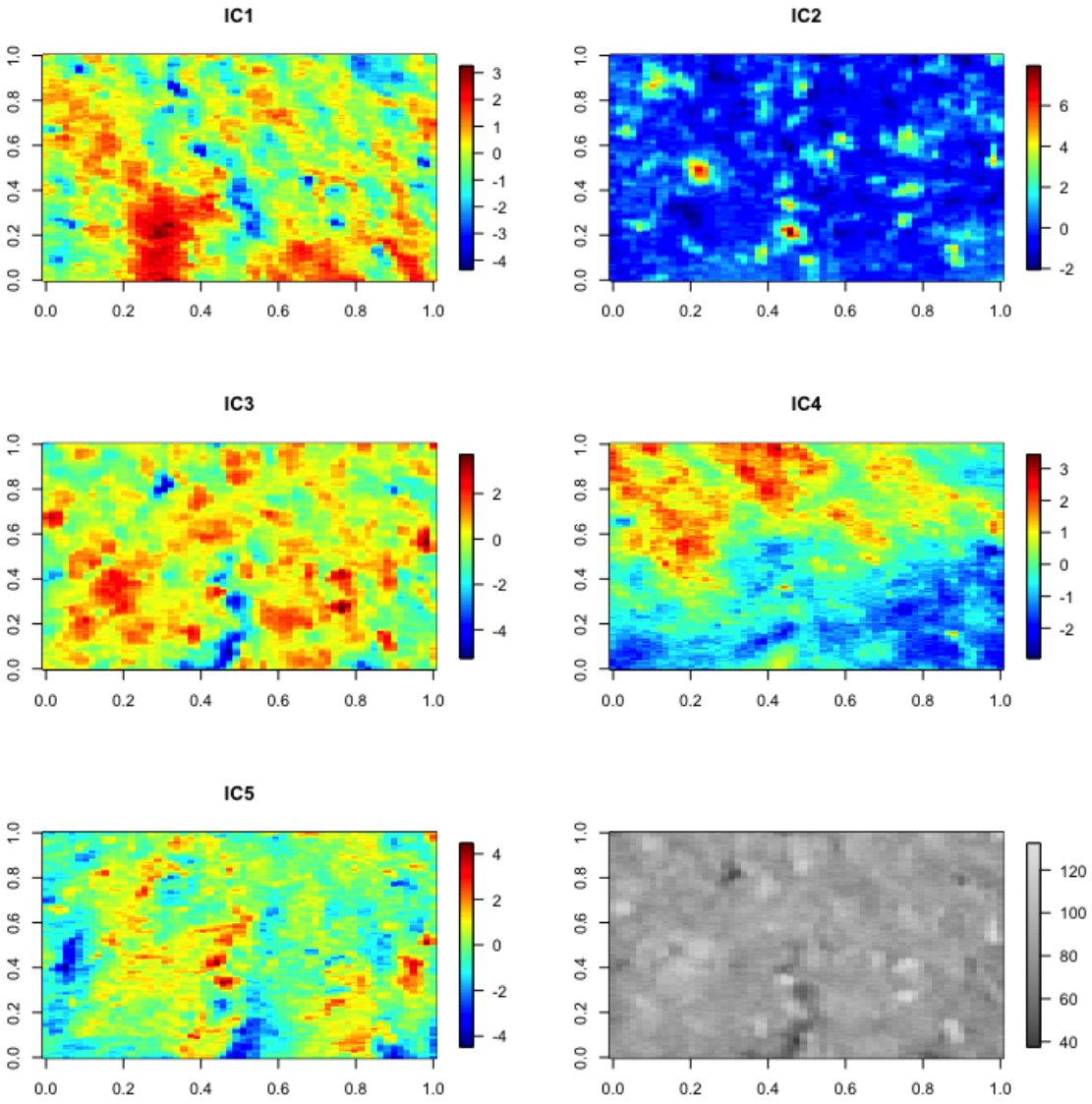
```
63 data.arr2 <- array(0, dim = c(rows,400))
64 i <- 1
65 r <- 0
66 while(i < dim(dataCube)[1]){
67   r <- r + 1
68   for(c in 1:400){
69     data.arr2[r,c] <- sum(data.mat[i,])
70     i <- i + 1
71   }
72 }
73
74 image.plot(data.arr2,col = grey.colors(100))
75
76 #ICA
77 C <- 3
78 ica <- fastICA(data.mat,C)
79
80 ica.arr <- array(0, dim = c(rows,400,C))
81 i <- 1
82 r <- 0
83 while(i < dim(dataCube)[1]){
84   r <- r + 1
85   for(c in 1:400){
86     ica.arr[r,c,] <- ica$S[i,]
87     i <- i + 1
88   }
89 }
90
91 par(mfrow=c(2,2))
92 for(i in 1:C){
93   image.plot(ica.arr[,,i], main = paste0("IC", i))
94 }
95 image.plot(data.arr2,col = grey.colors(100))
```

```
96
97  # Data minced beef -------
98  data.section <- data.sec45
99  D <- dim(data.section)[1]*dim(data.section)[2]
100
101 data.mat <- matrix(0,D,500)
102 i = 1
103 while(i < D){
104   for(r in 1:dim(data.section)[1]){
105     for(c in 1:dim(data.section)[2]){
106       data.mat[i,] <- data.section[r,c,]
107       i <- i + 1
108     }
109   }
110 }
111
112 # ICA
113
114 C <- 5
115 ica <- fastICA(data.mat,C)
116
117 ica.arr <- array(0, dim = c(dim(data.section)[1],dim(data.section)
        [2],C))
118 i <- 1
119 r <- 0
120 while(i < D){
121   r <- r + 1
122   for(c in 1:dim(data.section)[2]){
123     ica.arr[r,c,] <- ica$S[i,]
124     i <- i + 1
125   }
126 }
127
```

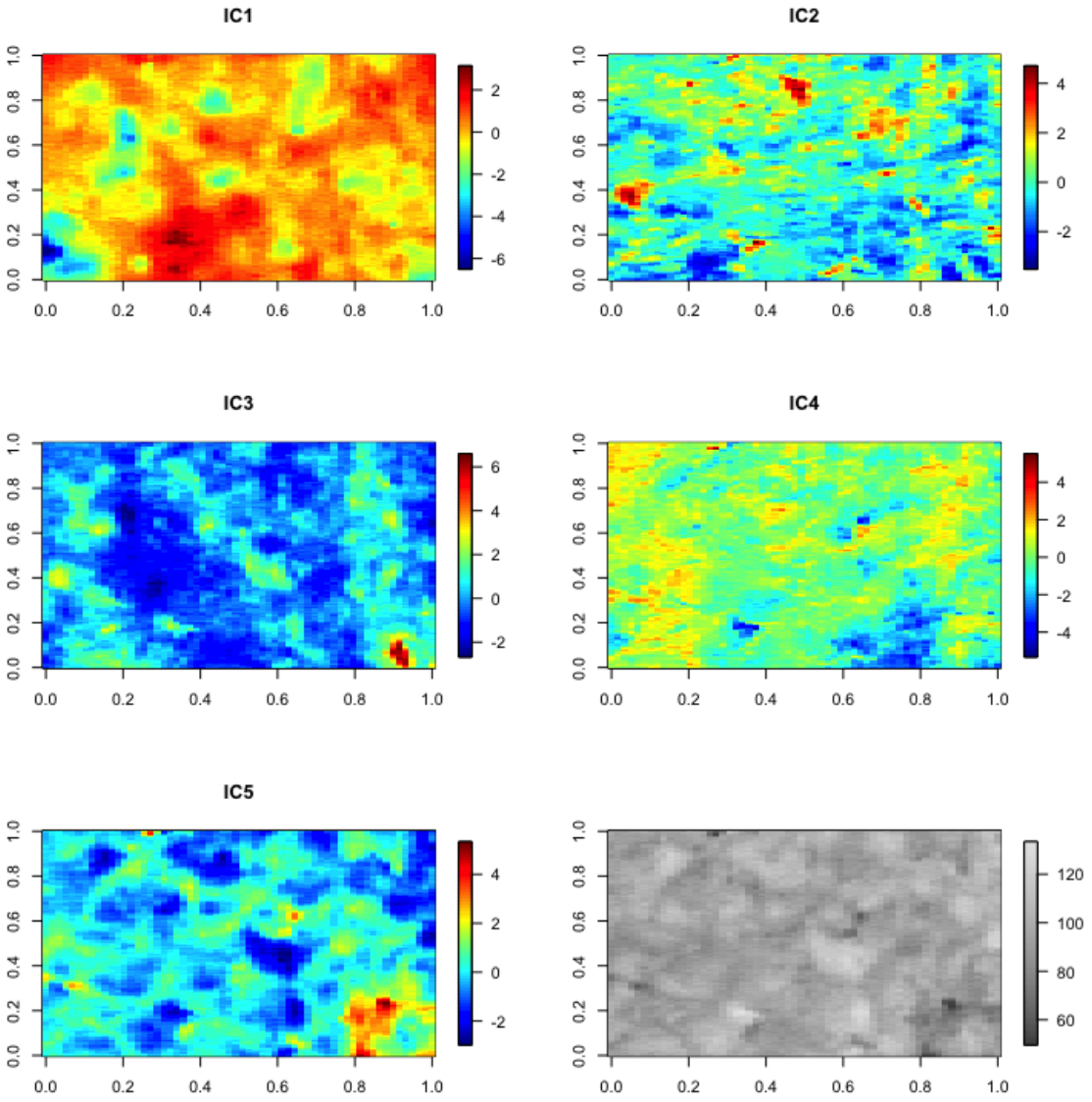
```
128 par(mfrow=c(3,2))
129 for(i in 1:C){
130   image.plot(ica.arr[,,i], main = paste0("IC", i))
131 }
132 image.plot(data.section[,,250], col = grey.colors(100))
```

B Minced Beef results

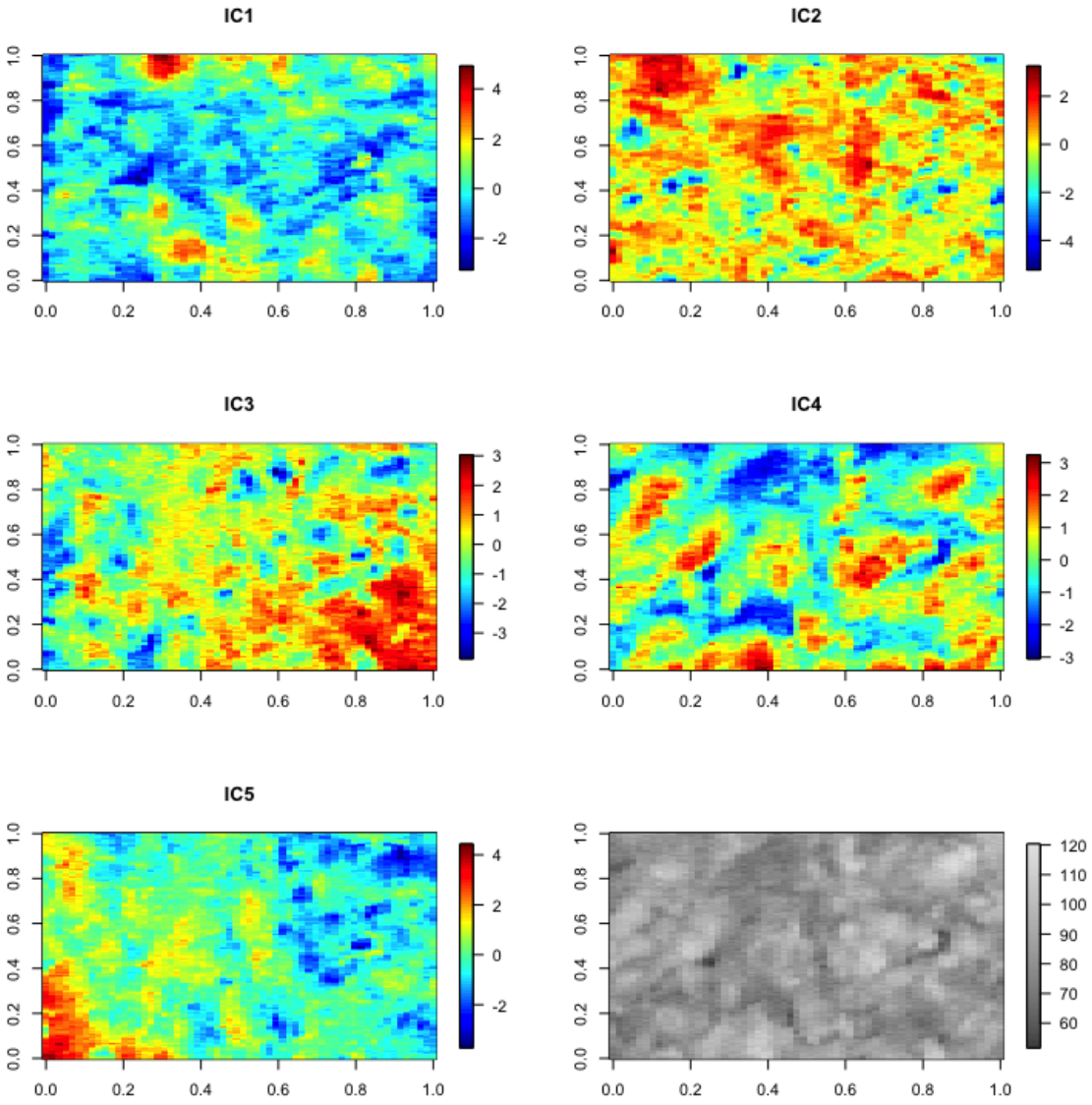
B.1 Minced beef experiment 18 - 48h



B.2 Minced beef experiment 19 - 24h

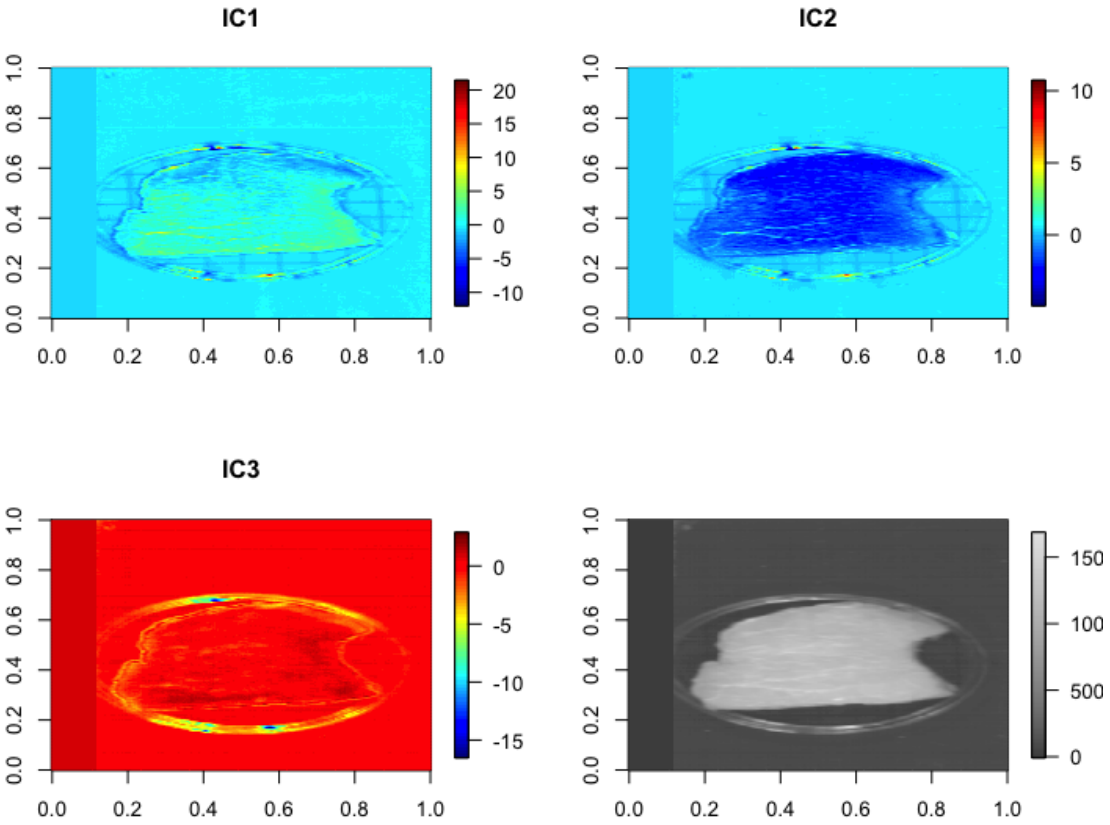


B.3 Minced beef experiment 19 - 48h

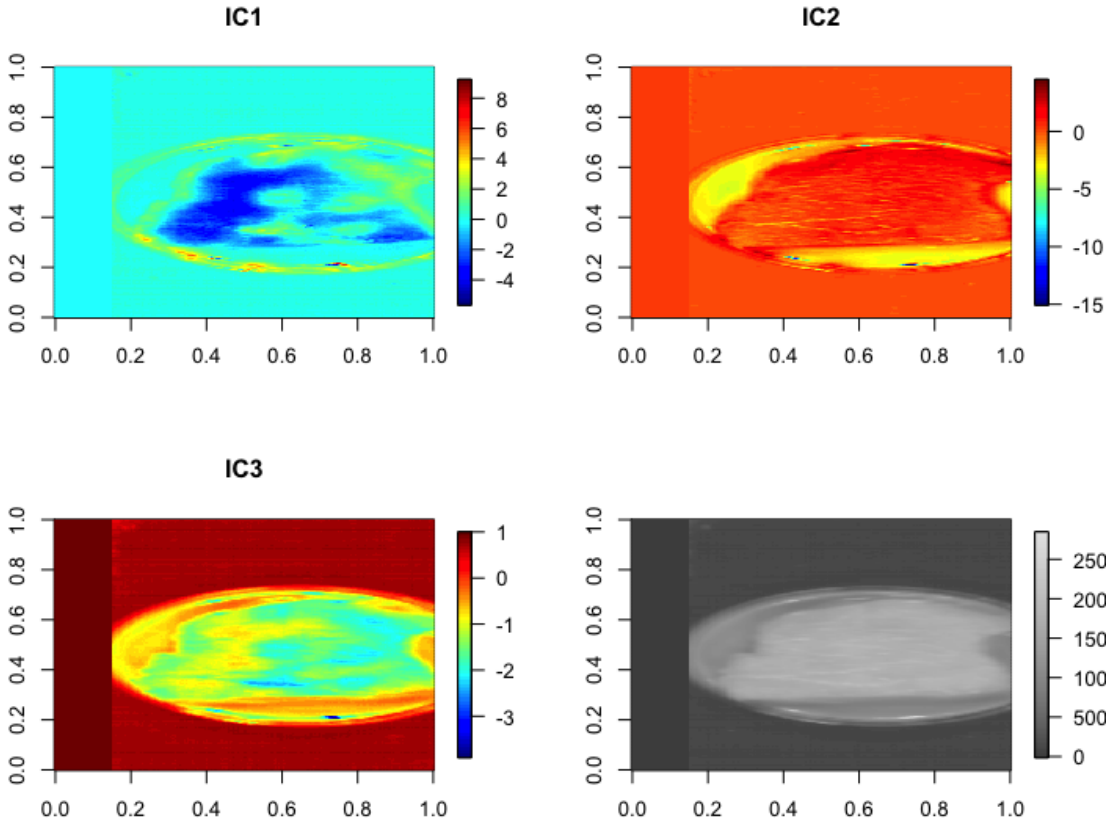


C Pork chop results

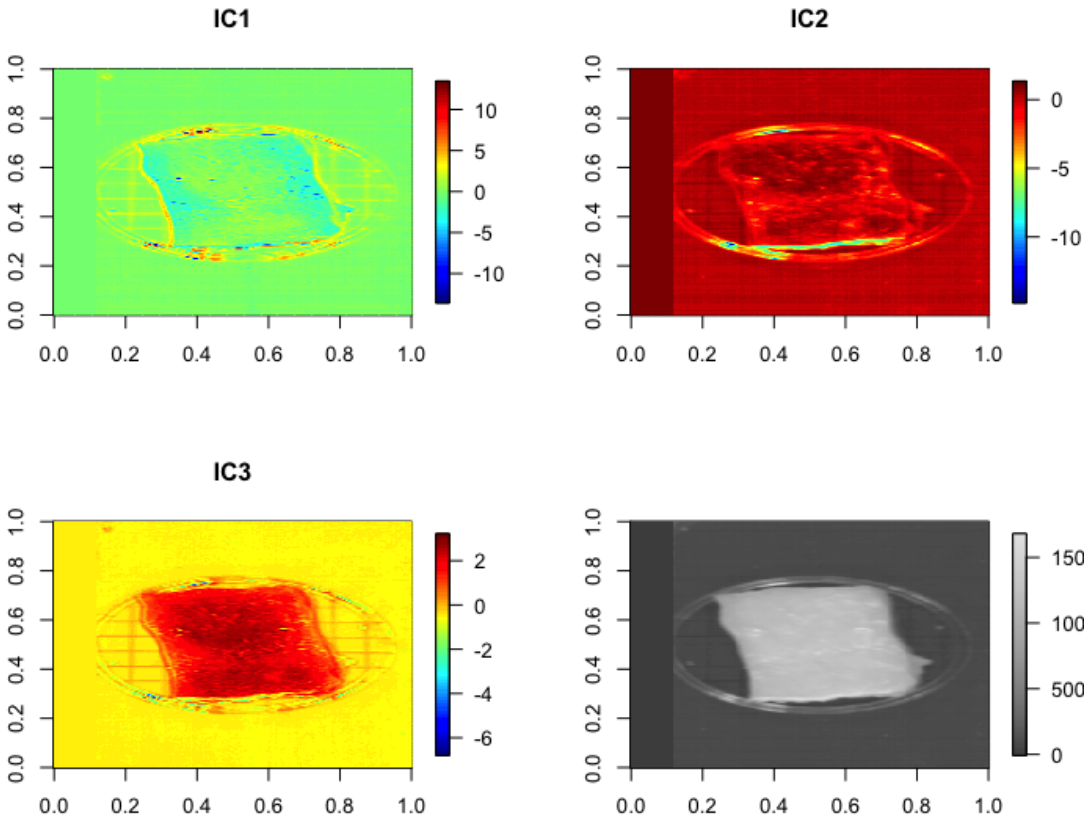
C.1 Pork chop dataset 2 - 0h



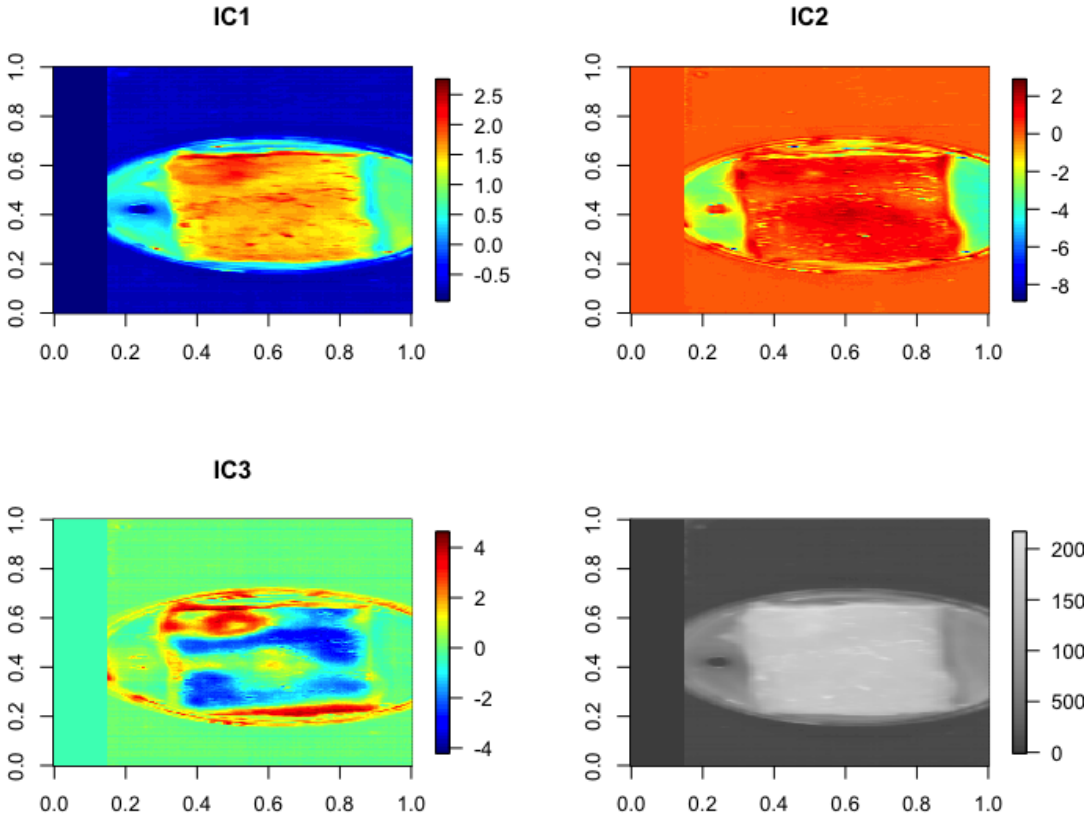
C.2 Pork chop dataset 2 - 24h



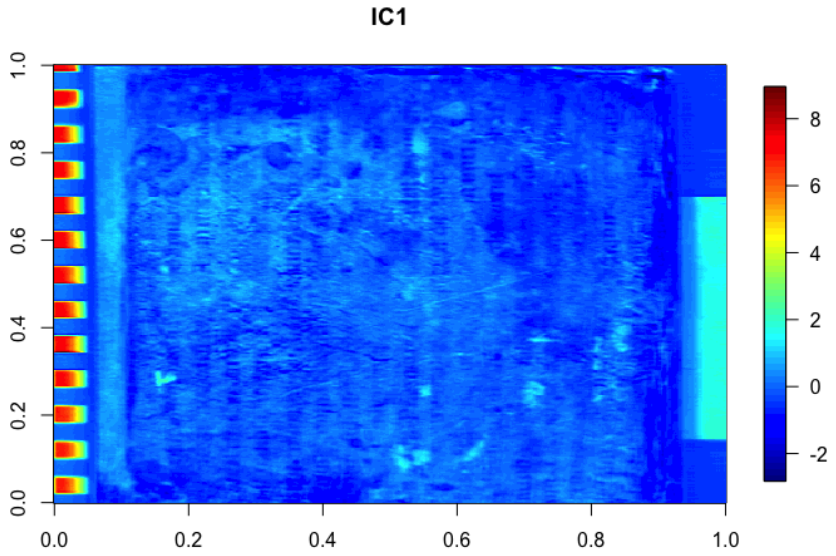
C.3 Pork chop dataset 3 - 0h

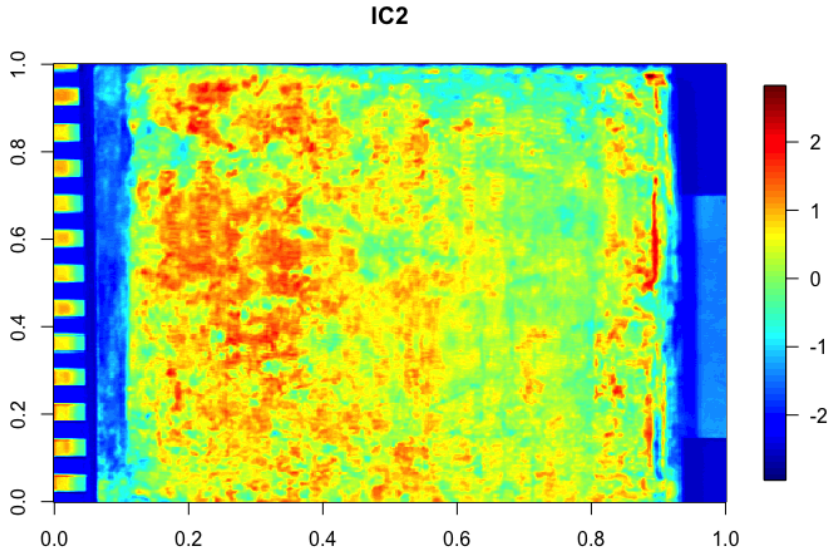


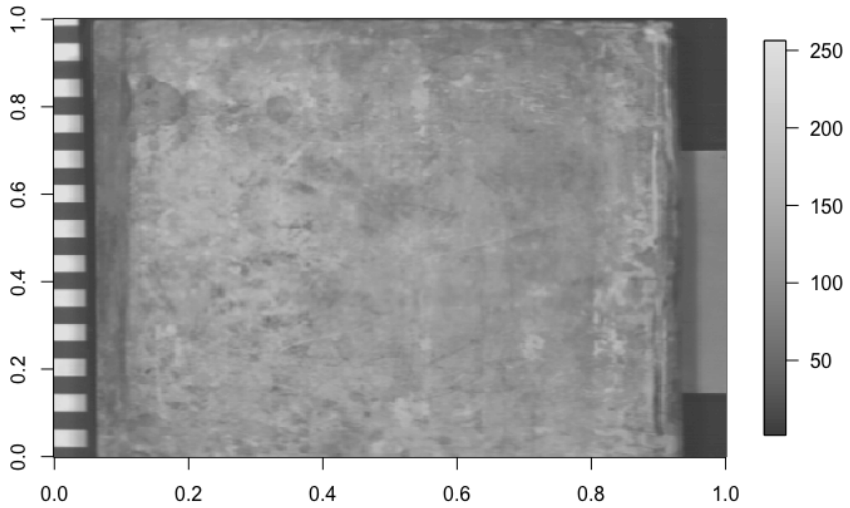
C.4 Pork chop dataset 3 - 24h



D Book






1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
250
200
150
100
50