Mass Spectrometry Analysis

Data “smoothing” is a very common problem in data science and statistics. We are often interested in examining the unknown relationship between a dependent variable (y) and an independent variable (x), under the assumption that the dependent variable has been imperfectly measured and has been contaminated by measurement noise. The model of reality that we use is,

$$y = f(x) + \varepsilon$$

where f(x) is some unknown, “true”, potentially non-linear function of x, and $\varepsilon \sim N(0, \sigma 2)$ is a random disturbance or error. This is called the problem of function estimation, and the process of estimating f(x) from the noisy measurements y is sometimes called “smoothing the data” (even if the resulting curve is not “smooth” in a traditional sense, it is less rough than the original data). In this question you will use several tools to try and estimate an underlying function in a simulated, but realistic example of a data science problem.

Mass Spectrometry Data Smoothing:

The file ms.train.csv contains n = 300 (simulated) measurements from a mass spectrometer. Mass spectrometry is a chemical analysis tool that provides a measure of the physical composition of a material. The outputs of a mass spectrometry reading are the intensities of various ions, indexed by their mass-to-charge ratio. The resulting spectrum usually consists of a number of relatively sharp peaks that indicate a concentration of particular ions, along with an overall background level. A standard problem is that the measurement process is generally affected by noise – that is, the sensor readings are imprecise and corrupted by measurement noise. Therefore, smoothing, or removing the noise is crucial as it allows us to get a more accurate idea of the true spectrum, as well as determine the relative quantity of the ions more accurately. However, we would ideally like for our smoothing procedure to not damage the important information contained in the spectrum (i.e., the heights of the peaks).

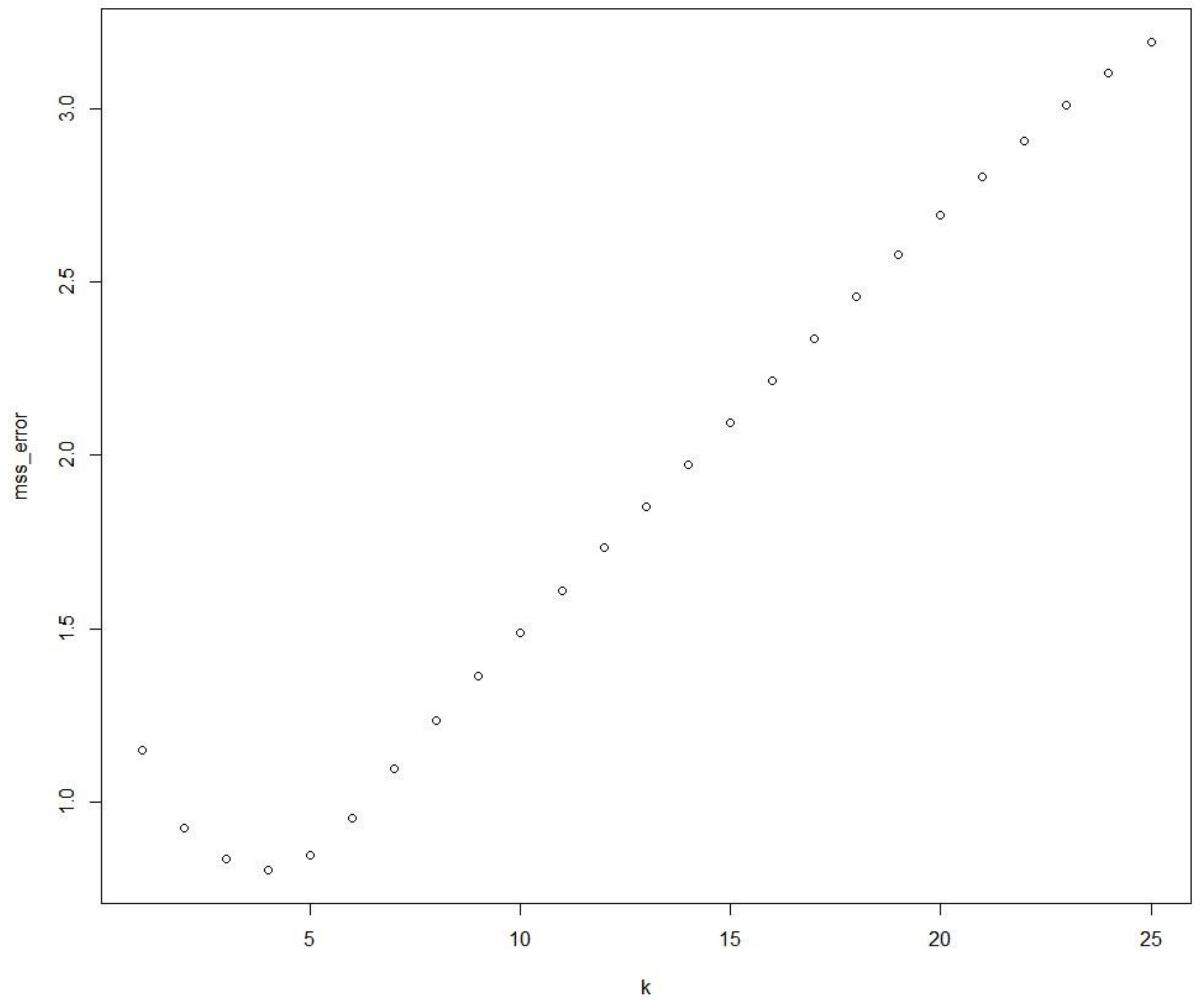
The file ms.train.csv contains the simulated measurements of our mass spectrometry reading; ms.train$MZ are the mass-to-charge ratios of various ions, and ms.train$intensity are the measured (noisy) intensities of these ions in our material. The file ms.test.csv contains n = 10, 000 different values of MZ along with the “true” intensity values, stored in ms.test$intensity. These true vales would be unknown in a real-life situation but can be used here to see how close your estimated spectrum is to the truth. For reference, the samples ms.train$intensity and the value of the true spectrum ms.test$intensity are plotted in Figure 1 against their respective MZ values.

Please note that kknn and rpart packages are used in this task.

2.1)

Use the k-nearest neighbours method (k-NN) to estimate the underlying spectrum from the training data. Use the kknn package we examined in Studio 9 to provide predictions for the MZ values in ms.test, using ms.train as the training data. You should use the kernel = "optimal" option when calling the kknn () function. This means that the predictions are formed by a weighted average of the k points nearest to the point we are trying to predict, the weights being determined by how far away the neighbours are from the point we are trying to predict.

a) For each value of k = 1, . . ., 25, use k-NN to estimate the values of the spectrum for the MZ values in ms.test$MZ. Then, compute the mean-squared error between your estimates of the spectrum, and the true values in ms.test$intensity. Produce a plot of these errors against the various values of k.

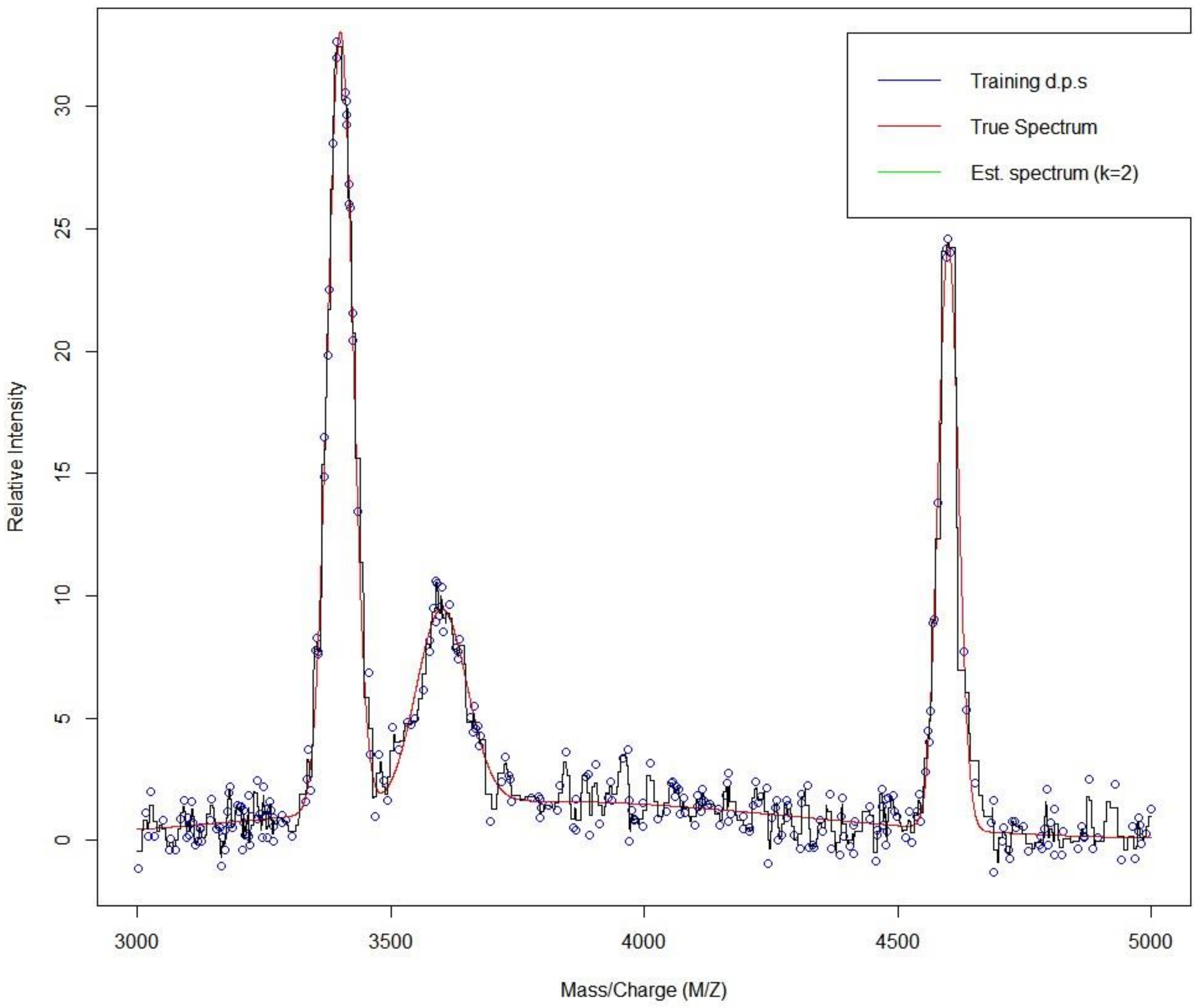


b)

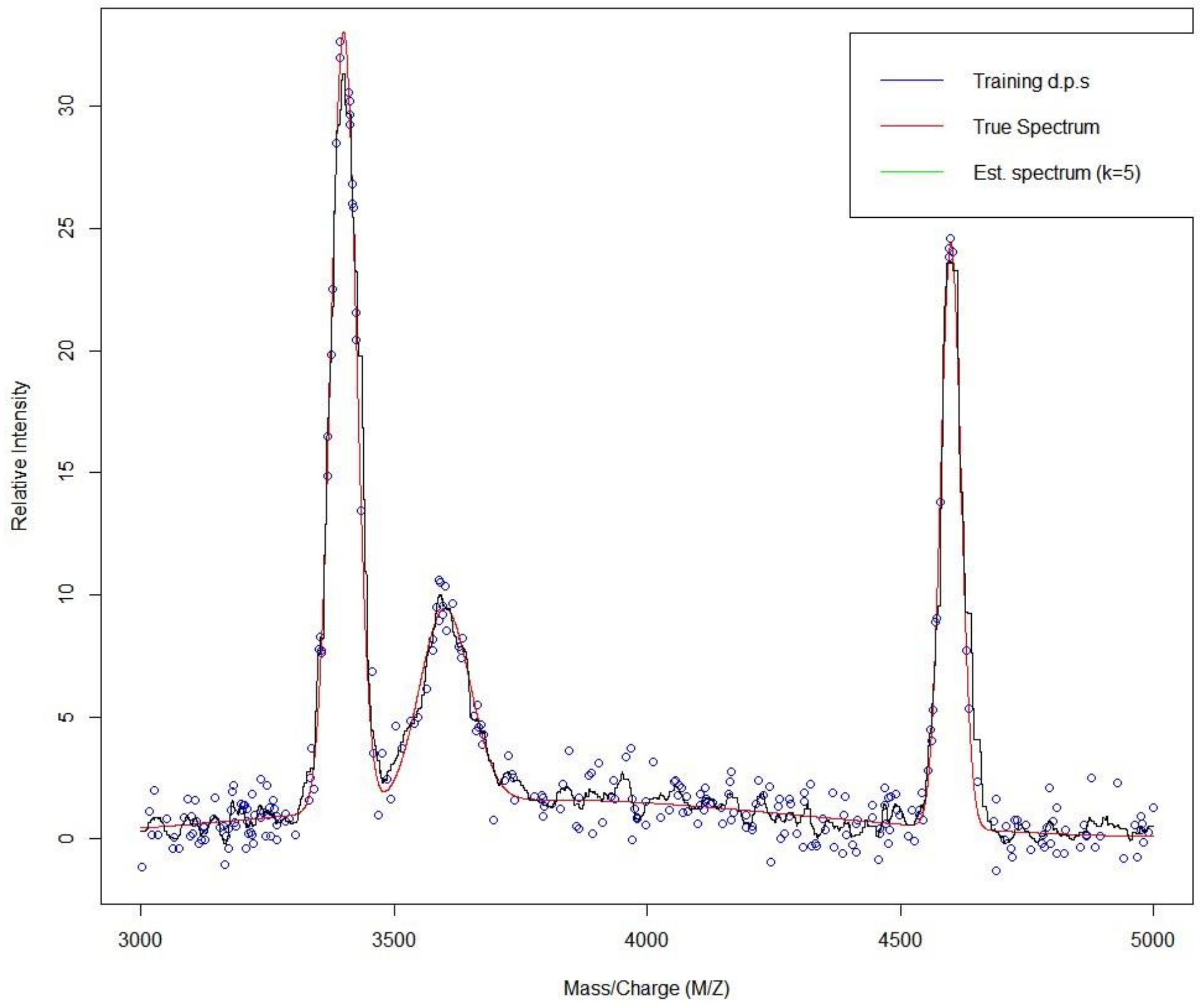
Produce four graphs, each one showing:

(i) the training data points (ms.train$intensity), (ii) the true spectrum (ms.test$intensity) and (iii) the estimated spectrum (predicted intensity values for the MZ values in ms.test.csv) produced by the k-NN method for four different values of k; do this for k = 2, k = 5, k = 10 and k = 25. Make sure the graphs have clearly labelled axis' and a clear legend. Use a different colour for your estimated curve.

K=2,

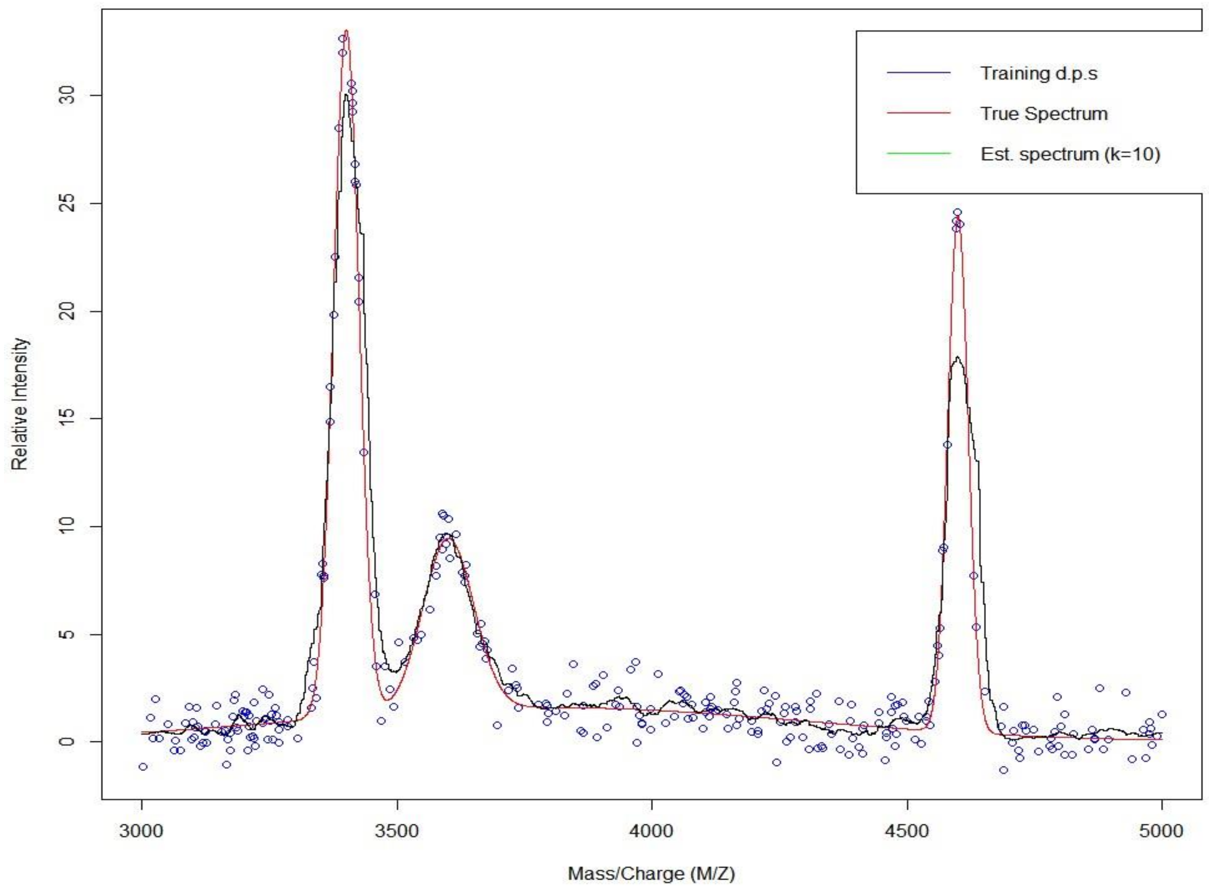

Training d.p.s
True Spectrum
Est. spectrum (k=2)
Relative Intensity
Mass/Charge (M/Z)
0
5
10
15
20
25
30
3000
3500
4000
4500
5000

K=5.

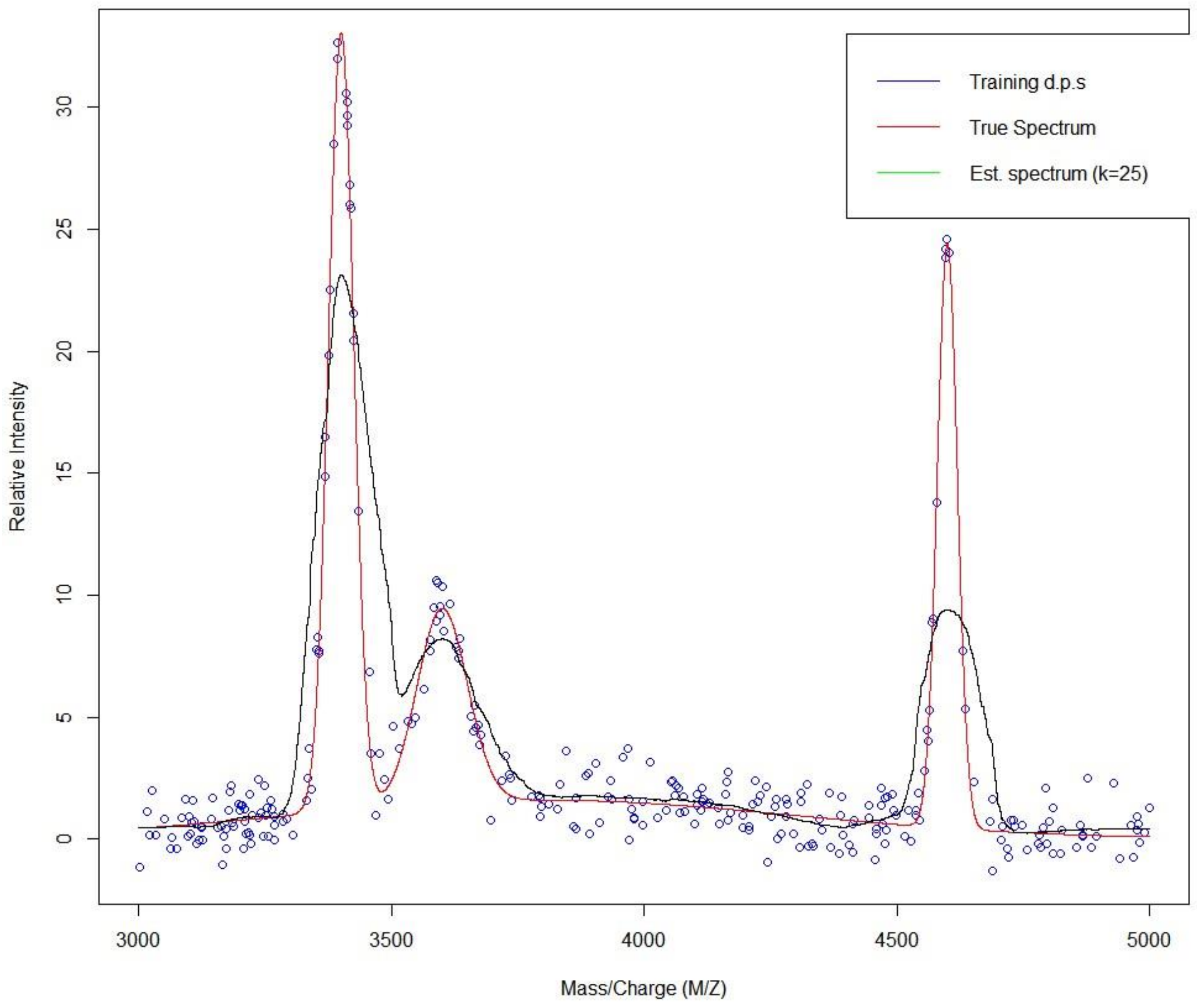


Training d.p.s
True Spectrum
Est. spectrum (k=5)
Relative Intensity
Mass/Charge (M/Z)
0
5
10
15
20
25
30
3000
3500
4000
4500
5000

K=10,


Training d.p.s
True Spectrum
Est. spectrum (k=10)
Relative Intensity
Mass/Charge (M/Z)
0
5
10
15
20
25
30
3000
3500
4000
4500
5000

K=25,



c)

Discuss, qualitatively, and quantitatively (in terms of mean-squared error on the true spectrum) the four different estimates of the spectrum.

```
> mss_error_2
[1] 0.9250907
> mss_error_5
[1] 0.8468406
> mss_error_10
[1] 1.486122
> mss_error_25 [1]
3.193641
```

The 4 mean-squared error on the 4 different estimates of the spectrum using k = 2, k = 5, k = 10 and k = 25 are 0.9250907, 0.8468406, 1.486122 and 3.193641, respectively.

According to these values, the estimated spectrum that uses k = 5 has the lowest mean-squared error and the estimated spectrum that uses k = 25 has the highest mean-squared error. This indicates that the estimated spectrum that uses k = 5 has the least average squared deviation from the true spectrum, as seen in the graph, while the spectrum that uses k = 25 deviates the most from the true spectrum (in terms of average squared deviation). Furthermore, we can also see that there is an optimal value of k.

where the average squared deviation is the lowest and getting closer to this value of k leads to less deviation from the true spectrum, while getting further will cause more deviation from the true spectrum. According to the sequence of values, that optimum value of k seems to be between 2 and 5.

2.2)

Use the cross-validation functionality in the kknn package to select an estimate of the best value of k (make sure you still use the optimal kernel). What value of k does the method select? How does it compare to the (in practice, unknown) value of k that would minimise the actual mean-squared error (as computed in Q2.1a)?

Using the cross-validation functionality in the kknn package in R script, the value of k selected is k = 3. Compared to the value of k in 2.1a that minimizes the actual mean-squared error, which is k = 4, the cross-validation functionality almost selects a correct k value, as it still selects a k value that minimizes the squared error value by a lot (but not the most minimization), which indicates that either the data or kmax provided might be increased to provide a more accurate k value.

2.3)

Using the estimates of the curve produced in the previous question, see if you can provide an estimate of the variance of the sensor/measurement noise that has corrupted our intensity measurements.

The variance found using the method of the average variance with the squared value of difference of mean with value divided by number of rows gives: 32.026271. It seems like the noise has corrupted the intensity with a quite a decent value.

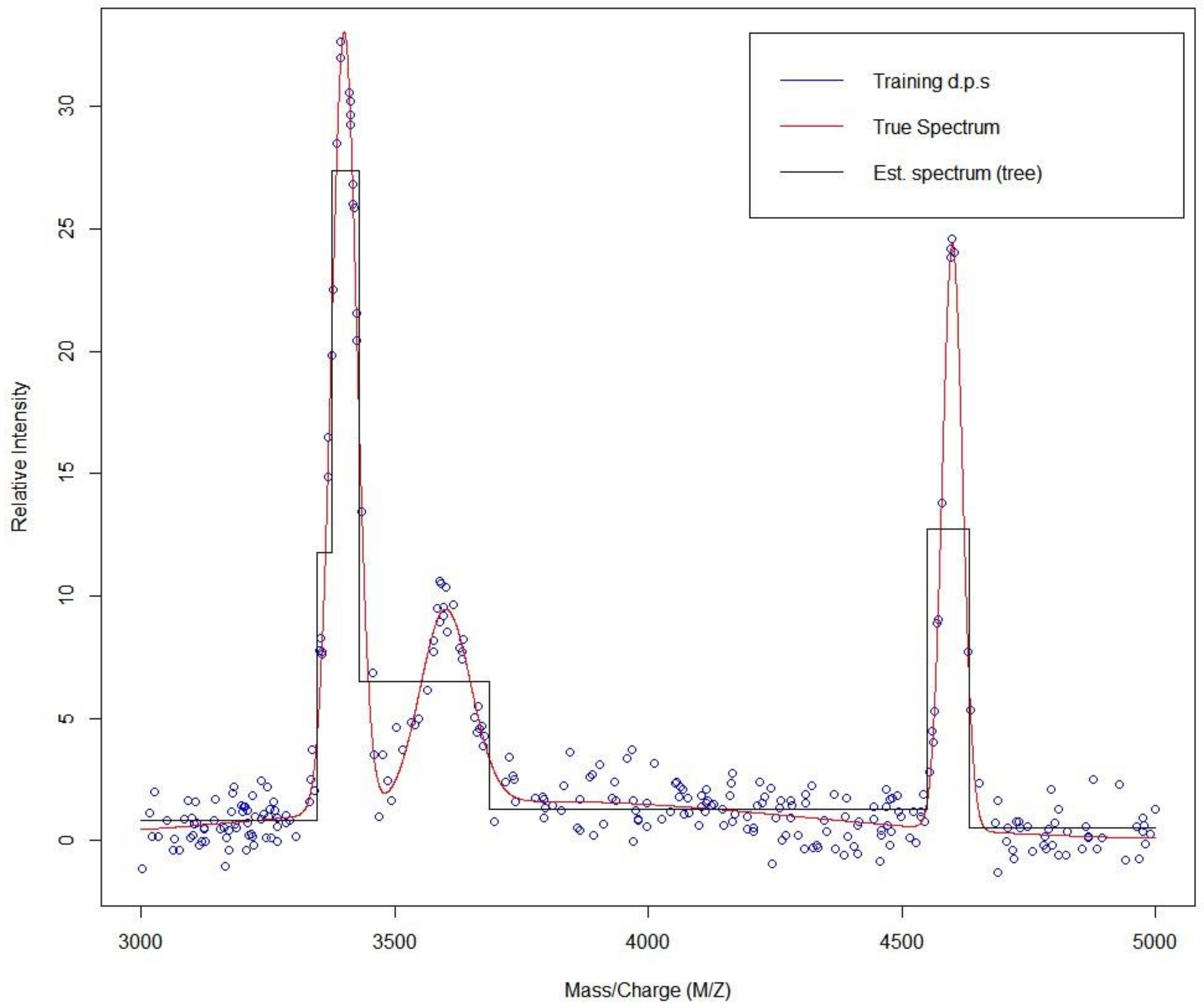
2.4)

Do any of the estimated spectra achieve our aim of providing a smooth, low noise estimate of background level as well as accurate estimation of the peaks? Explain why you think the k-NN method is able to achieve, or not achieve, this aim.

As from the previous values, the various estimated spectra, particularly for k values 2 and 5, provide an accurate estimate of our peaks, but these estimates are smooth and contain some noise. Whereas the estimated spectra for k values 10 and 25 provide estimates with lower noise than the previous 2, but are quite inaccurate in their estimates, especially for k = 25, which has the most deviation from the true spectrum's peak values. Despite the presence of some noise in the estimated spectra for k values 2 and 5, I believe that the k-NN method cannot achieve our aim of providing an accurate, smooth and low noise estimate of background level and peaks, as the presence of accurate peak estimation will result in a rough, high noise estimate of background level (k=2 and k=5), while a smooth and low noise estimate of the background level will in turn provide us estimates of the peaks that are inaccurate and have high deviation from the true values (k=10 and k=25).

2.5)

We can also use a decision tree to smooth our data. Learn a decision tree on ms.train to predict intensity given MZ using CV. Produce a plot, as above, with the predicted intensity values for the MZ values in ms.test.csv using the final tree found by CV. Compare this qualitatively, and quantitatively (in terms of mean-squared error on the true spectrum) to the k-NN method.



The mean squared error for the tree above found is 2.173358, which is higher than the lowest meansquared error that the k-NN method could obtain with an optimal value of k. This indicates that there is more deviation in the tree's predicted intensity values from the true values than the estimated spectra values obtained using k-NN with an optimal value of k, and thus mean that the decision tree provides an

estimation of the peaks that is less accurate than the k-NN's estimated spectra values in tree have more deviation from true values when compared to values of estimated spectra from k-NN method.

2.6)

Compare and contrast the decision tree approach with the k-NN approach on this one-dimensional problem. Thinking about the way in which both methods work, discuss how they are similar, and they are different.

The decision tree method works by splitting the predictor space (the various M/Z levels) such that the intensity values are evenly split between each predictor and to obtain an optimal number of 'leaf nodes', while the k-NN method "measures" each M/Z level and classifies its intensity according to the nearby M/Z levels.

Both methods work by grouping together M/Z levels with 'similar' attributes together, but the k-NN method is different from the decision tree method, as it determines which classification an individual belongs to (its intensity) from its neighbor's classifications, whereas decision trees aim to split the 'nodes' (intensity) evenly. So, decision tree split it into two values with the probability of heart-disease is present in patient or not while KNN method used neighbor's classification to determine if there is any chance of disease.

2.7)

Apply the bootstrap procedure to the spectrum estimated by k-NN using the value of k found in Q2.2. Using the outputs of the bootstrap function boot (), plot an approximate 95% confidence interval for the estimated spectrum for the MZ values in ms.test along with the true spectrum in ms.test and the average of the bootstrap estimates of the spectra (to keep things clean do not include the measurements on this plot). Run it for at least 100 bootstrap iterations. Discuss the confidence interval in comparison to the true spectrum. (hint: the $t field of the object returned by boot () gives you access to the bootstrap estimates for each bootstrap iteration as a matrix)

The estimated MZ value was found 0.7643 before while the confidence interval is found (0.45, 0.87) with 95% being confident. And It does certainly contain the value which is in the range.

we can see that the predicted classification accuracy does fall into the calculated confidence interval, which means that we can be quite sure that the MZ values with the given data does indeed fall in the interval above. And it seems to be accurate.