6404 Take-home Exam 3: Wavelets, Categorical Data Analysis, and Nonparametric Regression

Peter Williams, pwilliams60@gatech.edu

Date: 2018-12-01

Contents

1. Wavelets (50%)

Locate a one-dimensional data set that has sharp-changes like those presented in the recent lectures for applying the following wavelet procedures. It is best that the data size is larger than 512, and is in 2-factorial, e.g., 210 = 1024. Note that you need to locate proper R-package/codes to perform the tasks below.

For these exercises a dataset was located from the data science competition website, Kaggle (https://www.kaggle.com/datasets). The dataset consists of estimated energy consumption in Megawatts (MW) by hour provided by American Electric Power (AEP). AEP is a major investor-owned electric utility in the United States of America, which provides electricity to more than five million customers in 11 states including Texas and Ohio.

To simplify subsequent wavelet analysis, a subset of the original dataset was utilized. It consists of $n = 1024$ observations of hourly Megawatt consumption estimates beginning from 9:00am June 21st, 2018 to 12:00am August 3rd, 2018. A preview of the original dataset is shown below:

Table 1: AEP Hourly Energy Consumption in Megawatts

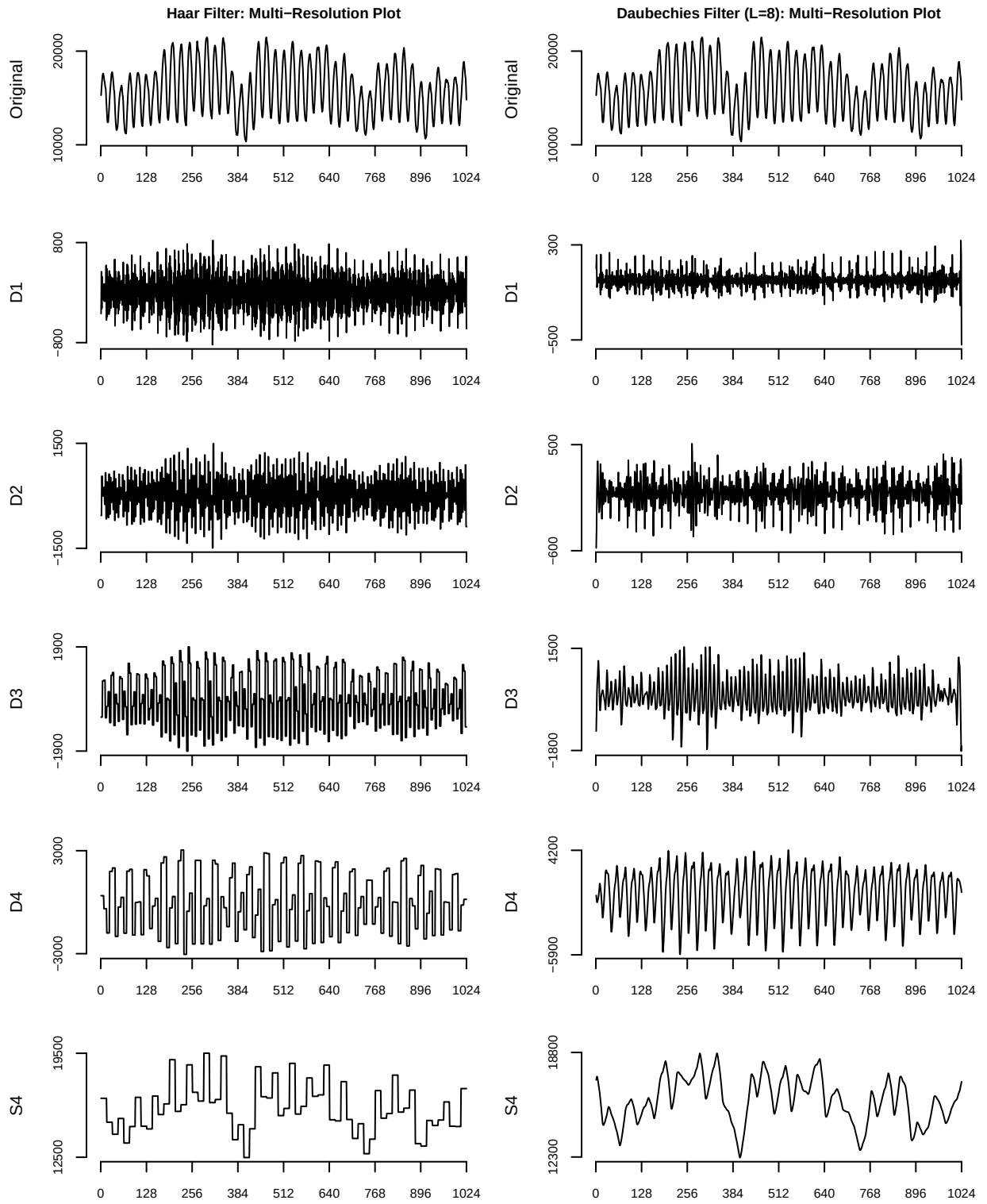
Hourly (Datetime)	Energy Usage (MW)
6/21/18 9:00	15,286
6/21/18 10:00	15,958
6/21/18 11:00	16,495
6/21/18 12:00	17,001
6/21/18 13:00	17,398
6/21/18 14:00	17,603

i) Select two families of wavelets for completing the two tasks ii) and iii) below, and make comparisons for the results impacted from distinct wavelet families.

ii) Show a multi-resolution plot of mother and father discrete wavelet-coefficients (DWTs), and make comments about their values.

A multi-resolution plot, including the original time series of energy consumption data, provides a lot of context on the data. The plot of the original data shown below highlights significant shifts in energy consumption in the time series, and a persistent daily cyclical pattern.

To contrast the multi-resolution plots below, we have selected both the 'Haar' and 'Daubechies' filters for comparison in each column. For visual simplicity and interpretation both plots rely on 4 levels, or depth in terms of signal reconstruction. The MRA plot quickly highlights, for both families, that for the first scaling level $S2$, we have a good representation of the original function, that at further depth the signals are much noisier. Also worth noting is the contrast in smoothness of the 'Daubechies' first level decomposition vs. the 'Haar'. In fact the first level for the 'Daubechies' family captures more of the original signal, as the magnitude, or scale of signals at lower levels is tighter relative to the 'Haar' decomposition, as shown below:



Note: To generate these plots, the 'waveslim' package was utilized in R, using the "mra" function with method "DWT" supplied as an argument which provides an additive decomposition of the original time series. Futher details are provided in the appendix.

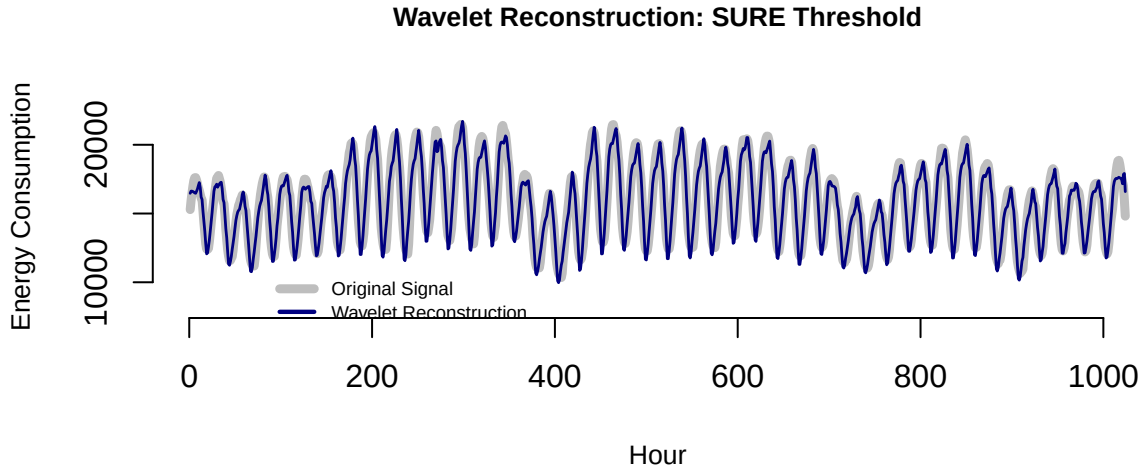
iii) Apply two thresholding/shrinkage methods to reduce number of non-zero DWT coefficients. Apply the IDWT to reconstruct the original data signal by using the thresholded DWTs. Comment on the quality of the reconstructions from plotting the original and reconstructed signals (by overlaying them in one figure). Compare the two methods about their data-reduction ratios and the MSEs.

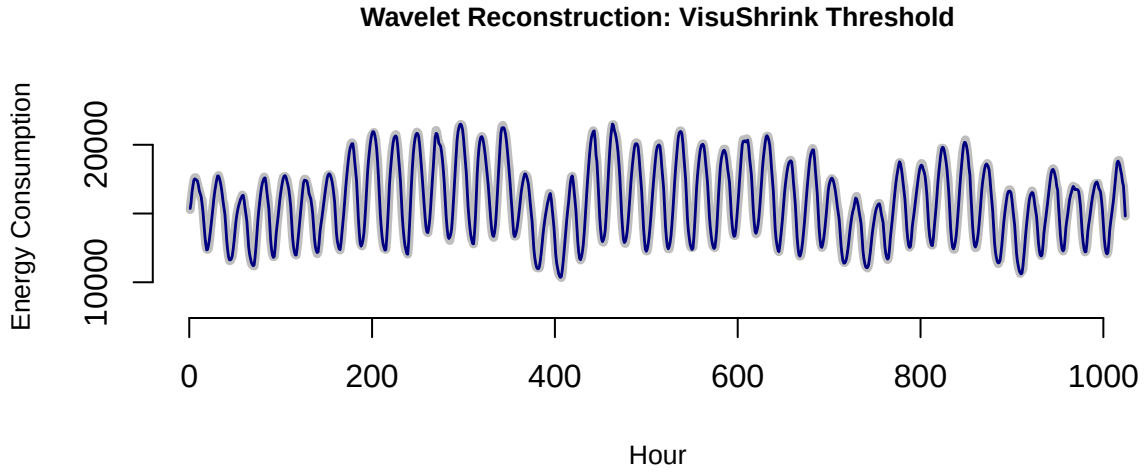
To reduce the number of non-zero DWT coefficients, we rely on two popular shrinkage methodologies to find smooth representations of our original series. We utilize the *SURE* and *VisuShrink* thresholding methodologies, described by Donoho and Johnstone in their 1995 pager *Adapting to Unknown Smoothness via Wavelet Shrinkage*. We apply thresholding on our DWT that utilized 4 levels, and the Daubechies Filter described above. A very brief of these thresholding methodologies is provided here:

The *VisuShrink* approach to thresholding relies on an estimate of the noise level of wavelet coefficients denoted $\hat{\sigma}$. $\hat{\sigma}$ is often implemented as a scaled median absolute deviation the realized wavelet coefficients. The thresholding itself is shown as, $*T_{UV} = \sqrt{2(log\ n)\hat{\sigma}}$ where n is the number of data points, and $\hat{\sigma}$ is as described above.

The *SURE* methodology utilizes Stein's unbiased risk estimation, as described by by Donoho and Johnstone in their paper. This methodology creates a threshold rule t_j for each resolution level j in a wavelet transform, and is found using cross-validation.

To constrast these two thresholding approaches, we plot the original signal in the figures below, against the reconstructed signal post-thresholding out DWT:





Visually it is quite clear the *VisuShrink* thresholding approach provides a more representative reconstruction of the original signal. The *SURE* approach is clearly less smooth, but has a slightly higher data reduction ratio which is shown in summary in the table below. We also provide summaries of the mean squared error (MSE), the root mean squared error (RMSE, which can be more easily interpreted on the original data scale), along with the data reduction ratio (%), which is computed as $100 \cdot (1 - \frac{\text{\#non-zero DWT Coefficients}}{\text{\#Original Observations}})$:

Table 2: MSE Results

threshold	mse
SURE	285763.0
VisuShrink	8507.9

Table 3: RMSE Results

threshold	rmse
SURE	534.6
VisuShrink	92.2

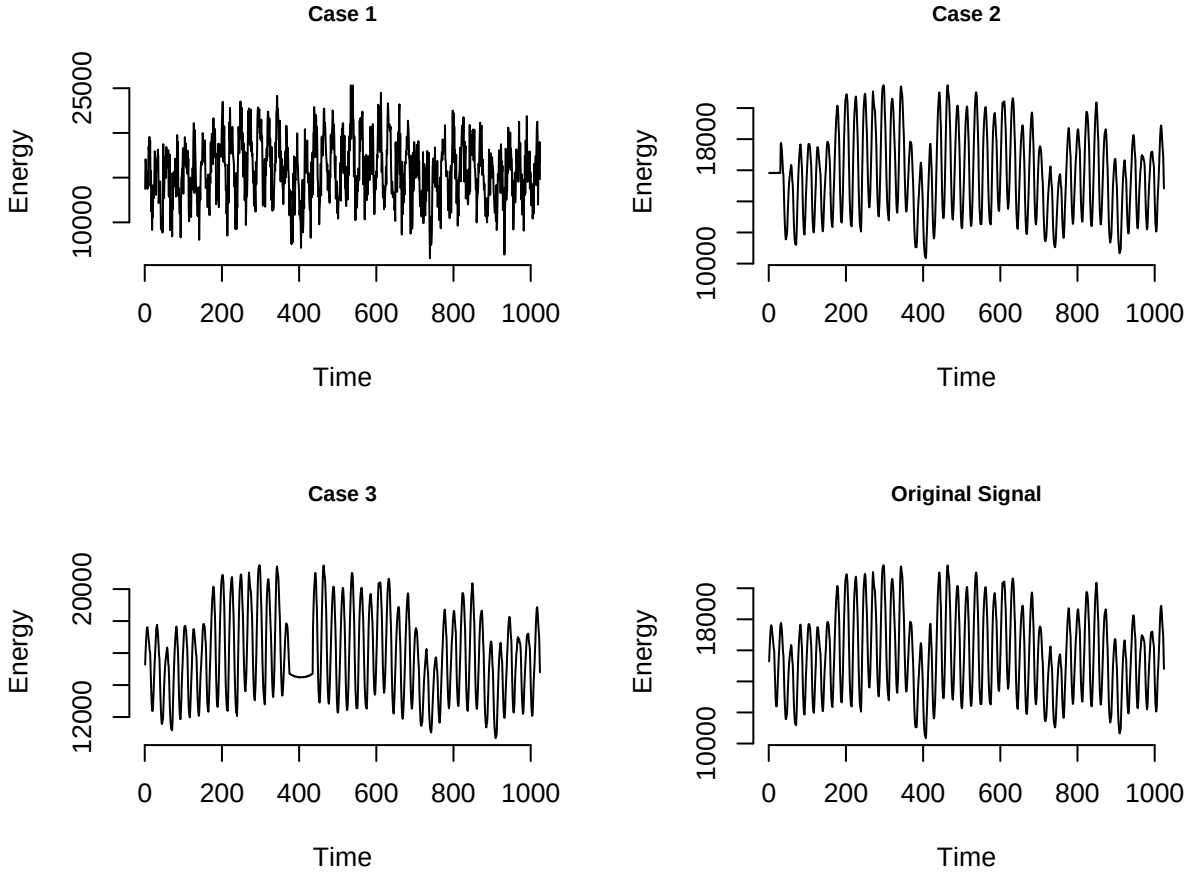
Table 4: Reduction Ratio

threshold	ratio (%)
SURE	74.90
VisuShrink	68.55

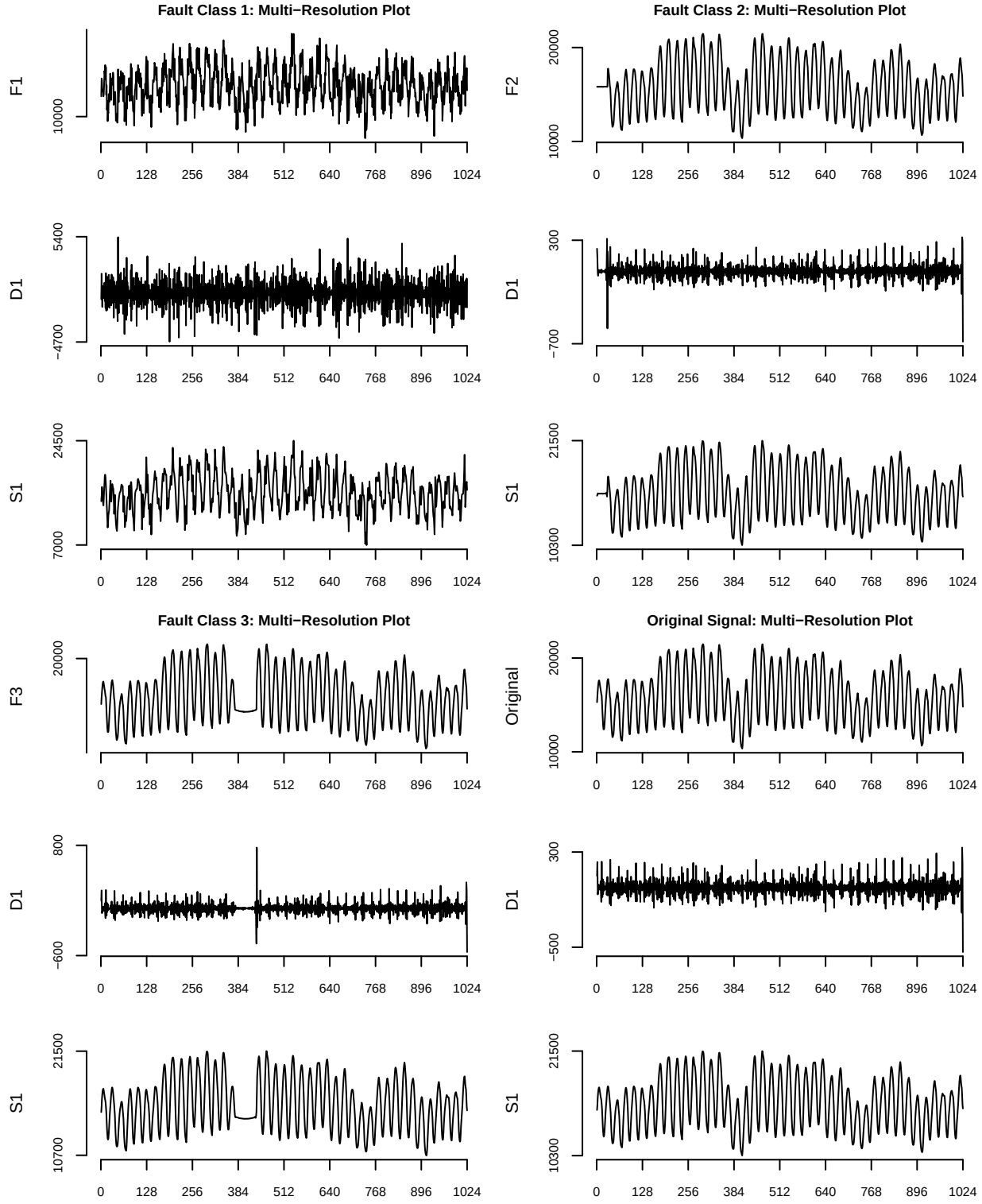
Comment on the results from the tables above.

iv) Artificially alter the data in one "local-region" and one large-size "global region" for creating 3 distinct "fault-class" data sets. See lecture presentation about details of this task. Apply the best thresholding method (and the best wavelet-family) to model the data from all FOUR classes (one from the original data and the other 3 fault-class data). Use the multi-resolution plots of thresholded-DWTs to see how they are different in these FOUR classes of data signals.

To create our fault class datasets, we replicated our original data and added some artificial alterations. As shown in the figure below for 'Case 1', we added random noise to the original signal, specifically $N(0, \sigma^2)$, with $\sigma = 2000$, given the scale of our original data. For 'Case 2', we shifted our original series by 25 time units to the right, added a small amount of noise $\sigma = 10$, and appended a sequence of values equal to the sample mean to the front of the series. Finally for 'Case 3', we added a small 'U' shaped curve from observations $375 - 436$, and added a small amount of noise. Comparisons of our altered series and the original series are shown below:



To explore how these artificial faults classes differ in their wavelet decomposition we can look at their respective multi-resolution plots, all using 2 resolution levels to simplify and condense visualizations:



Note: DWT decomposition performed using 'Daubechies' filter.

The multi-resolution plots above provide visual indications of differences across fault classes. For example, at the 'D1' resolution level, both the shifted fault class, 'Fault Class 2', and the artificially altered series

'Fault Class 3' show significant spikes in coefficient estimate level around the area of the signal where the series was altered. 'Fault Class' which was altered with a larger magnitude of noise, has significantly larger estimates across the entire signal for wavelet coefficients at level 'D1'. These visual difference highlight a possibility of classifying fault classes based on features that can be extracted from values of the wavelet coefficient estimates are various levels. This topic will be explored further in the next section.

v) Discuss the possibility and steps for developing a rigorous decision-making procedure to detect faulty-signals against the original data, and distinguish classes of faulty signals with reduced-size data presented by thresholded DWTs.

In the paper *Wavelet-Based Data Reduction Techniques for Process Fault Detection* (2006), Jeong and Lu et al. provide in-depth discussion on decision-making procedures for distinguishing classes of faulty signals with reduced size dataset. Based on the discussion in that and paper and our results from above, a decision-making procedure to detect and distinguish classes of faulty-signals should leverage:

- A structured data generation process to support a diverse, and representation or population of fault-type classifications
- In order to detect and distinguish classes of faulty signal, data should be generated that form a diverse set of classes representative of what faulty signals may look like
- A variety of techniques of generating faulty-classes are described by Jeong and Lu et al. include adding random noise existing datasets, adding artificial features and curves, and shifting time series etc.
- Features of the original data that are estimated from the multi-resolution decomposition, for example:
- Coarse level wavelet coefficients can be used to identify faults that relate to global faults related to the shape of the original data series relative to new signals
- Granular wavelet coefficients, represented at lower levels of the decomposition can be used to identify faults that related local and global faults representing faults smaller in magnitude relative to the dominant signal patterns

To undertake these tasks described programmatically, Jeong and Lu et al. utilize a classification and regression tree (CART) to classify datasets into fault classes by relying on estimates of wavelet coefficients at different resolution levels in the signal decomposition of both faulty, and original signals. Undertaking a decision-making process

2. Categorical Data Analysis (25%)

Chapter 9 of the textbook on categorical data analysis includes 6 sections with various problems/data and methods. Locate three sets of problems/data matching three distinct methods taught in lectures. Apply proper statistical software (preferred to be in R-codes) to analyze the data. Provide in-depth comments about the findings in your statistical analyses.

1. Chi-square, goodness of fit

To discuss the Chi-square goodness of fit procedure, a dataset published by the *National Oceanic and Atmospheric Administration* was located that tracks the number of hurricanes that have made landfall on the continental United States by decade over the last ~ 60 years. Links to the actual source data is linked below. To get a sense of what the dataset looks like a brief preview is shown below, where the expected count of hurricanes is computed as the grand mean of observed landfalls across the entire dataset:

Table 5: NOAA: Continental United States: Hurricane Impacts/Landfalls

Decade	Count of Impacts	Expected Impacts
1951-1960	18	15.667
1961-1970	15	15.667
1971-1980	12	15.667

Decade	Count of Impacts	Expected Impacts
1981-1990	16	15.667
1991-2000	14	15.667
2000-2010	19	15.667

Source: Continental United States Hurricane Impacts/Landfalls by decade as reported by the NOAA http://www.aoml.noaa.gov/hrd/hurdat/All_U.S._Hurricanes.html

Given the overall national interest in climate change, and its impact on weather patterns, it is of interest to many researchers if the frequency of hurricane landfalls in recent years in increasing. Given the dataset shown above, we can test this hypothesis in a crude way utilizing differences in observed vs. expected frequency of hurricane landfall. Where n_i refers to observed counts across $i = 1, ..., k$ decades measured, and $N = \sum_{=1}^{k} n_i$, is the total number of observed landfalls. We can then set up our test:

$$H_0 : F_X(x) = F_0(x),\ H_\alpha : F_X(x) \neq F_0(x)$$

We then take our test statistic T to be, $T = \sum_i^r \frac{(n_i - np_i)^2}{np_i}$, which is approximately distributed $T \sim \chi^2_{r-1}$.

Jumping into actual calculations, we have $T = \frac{(18-np_1)^2}{np_1} + \frac{(15-np_2)^2}{np_2} + ... + \frac{(19-np_6)^2}{np_6} = 2.128$, where $np_i = 15\frac{2}{3}$, for $i = 1, ..., 6$. Under our H_0, our p-value is 0.831, not sufficient evidence to reject our H_0 and conclude that hurricane landfalls are different across decades, but perhaps a contentious talking point in conversation. Obviously the scope and depth of the analysis here isn't sufficient to add meaningfully to the overall discussion about climate change.

2. Contingency tables, testing for homogeneity and independence

To demonstrate usage of contigency tables and associated tests using *R*, an article published by medical researchers that followed $N = 6272$, Swedish men for 30 years to see whether there was any association between the amount of fish in their diet and prostate cancer was located. According to authors of the research, the original study actually used pairs of twins. This approach allowed researchers to share their findings with more confidence. In the table below, we show frequency counts among research respondents across their fish diet type, and whether or not they contracted prostate cancer across the 30 year study:

Table 6: Occurence of Prostate Cancer, By Amount of Fish Consumption, N = 6272

	Large	Moderate	Small	Never
No	507	2769	2420	110
Yes	42	209	201	14

Source: Lancet, June 2001, "Fatty Fish Consumption and Risk of Prostate Cancer"

The table itself does not communicate readily any differences in cancer rates by diet type. To evaluate further how diet type may impact the rate of cancer, we utilize the chi-square test of independence, which is readily available via the function *chisq.est* in base *R*. Given a contingency table of consisting of m levels of a factor, along with an additional factor with k levels of observations, we can construct a matrix $N = (n_{ij}), i = 1, ..., m,\ j = 1, ..., k$, where each entry in N is an observed frequency, or count. The main gist of the chi-square test, is to study observed frequencies in N, n_{ij}, and their potential differences from the expected (under the hypothesis of independence) frequencies, denoted: $\hat{n_{ij}} = \frac{n_{i.} \cdot n_{j.}}{n_{..}}$, where $n_{i.} = \sum_{j=1}^k n_{ij}$, and $n_{j.} = \sum_{i=1}^m n_{ij}$, and compute our test statistic $T = \sum_{i=1}^m \sum_{j=1}^k \frac{(n_{ij} - \hat{n_{ij}})^2}{\hat{n_{ij}}}$, where $T \sim \chi^2$. The results of the test for our fish diet dataset are summarized in the table below:

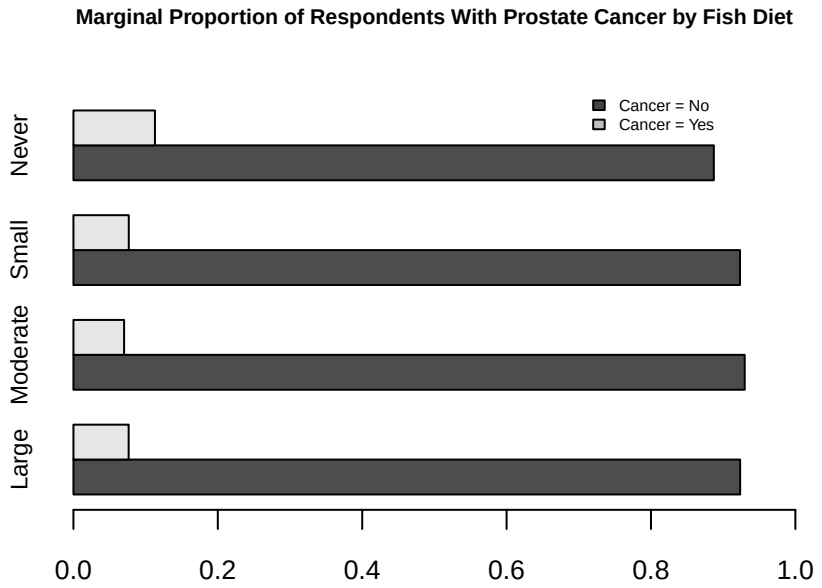
Table 7: Chi-square Test for Independence Test, R (base), function: "chisq.test (df = 3)"

Result	Value
Test Statistic (Chi)	3.677
P-value	0.298

As shown, our resulting P-value 0.298 does not provide evidence for us to conclude that the observed frequency of prostate cancer by fish diet type, is unexpected by itself. While this finding may be interesting, this test alone isn't sufficient to rule out diet's role in respondent's contraction of prostate, as there are potentially many other factors which can contribute to prostate cancer in addition to diet alone.

3. Fisher exact test

Taking our analyis of prostate cancer, and fish diet further, visual inspection of the marginal proportion of prostate cancer by fish diet classification shows that those who never consumed fish in their diet, may have had slightly higher rates of prostate cancer, which was identified and estimated in the *Lancet* paper referenced above. Here is a basic barplot to visualize difference in rate of cancer by diet:



Since, as shown in the text, the Fisher exact test is based on the null hypothesis that *two* factors, each with *two* factor levels, are independent, conditional on fixing marginal frequencies for both factors. However, the fish diet dataset described above is 2×4, therefore for the purposes of this question, we subset our data to just those with 'Never', and 'Large' fish diets for comparison.

As shown below, relying on the only the *fisher.test* function to compute *Fisher's Exact Test* in *R (base)* to detect any differences in counts across diet categories. The test shown below, relies on the odds ratio between the occurence of cancer between groups ('Never' and 'Large'):

Table 8: Fisher's Exact Test: Fish Diet Comparison (Confidence = 0.95)

Result	Value
Odds Ratio	1.535
Upper Confidence Level	0.747
Lower Confidence Level	2.989
P-Value	0.207

Again, our resulting P-value from *Fisher's Exact Test* does not provide evidence for us to conclude that the observed frequency of prostate cancer by fish diet type, is unexpected. As highlighted before, there are potentially many other factors which can contribute to prostate cancer in addition to diet alone. The research referenced above in *Lancet* takes a deeper dive into this particular dataset and concludes that diet, in fact, did play a role describing respondents' rate of contracting prostate cancer, however, relying on more sophisticated statistical techniques.

3. Nonparametric Regression (25%)

Locate a data set suitable for both kernel and spline regressions. The data should include at least 3 x-variables. It is okay to focus on additive-models discussed in lectures.

To consider procedures for both kernel and spline regression, we utilize the well-known automobile MPG dataset that is available in the UCI Machine Learning library. Link here: https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data. The dataset was used in the 1983 American Statistical Association Exposition, and is also well referenced in the following paper:

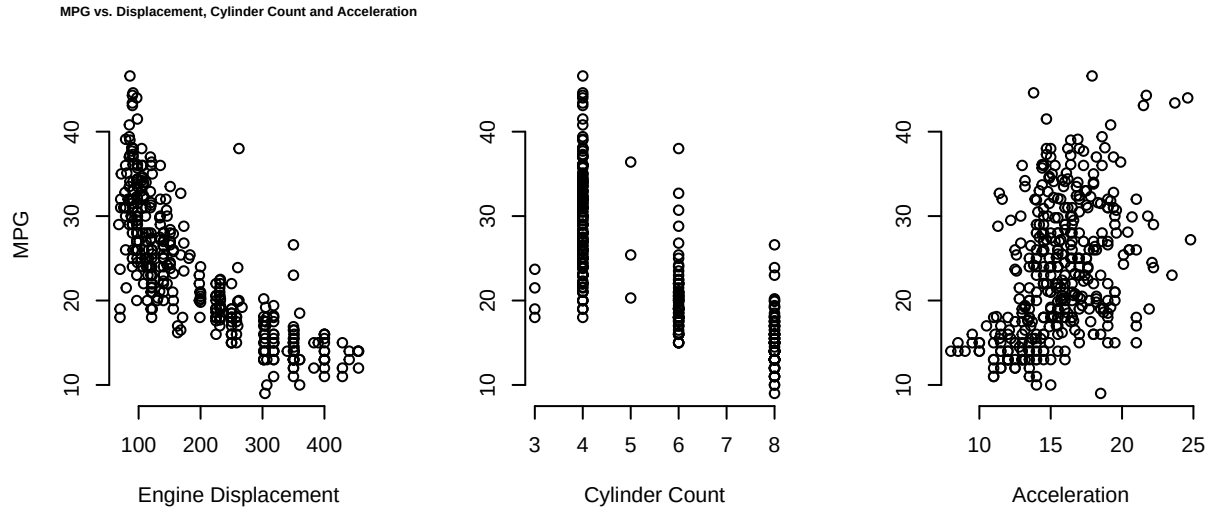
Quinlan,R. (1993). Combining Instance-Based and Model-Based Learning. In Proceedings on the Tenth International Conference of Machine Learning, 236-243, University of Massachusetts, Amherst.

As highlighted by Quinlan in the paper above, this dataset is concerned with city-cycle fuel consumption in miles per gallon, to be predicted in terms of a number of attributes of each car. Each car in the dataset ($N = 392$) was manufactured sometime in between 1970 and 1982. For convenience purposes we have removed observations with missing values. Based on the exercise specification we will rely on a car's *Mileage Per Gallon* (mpg), as our outcome of interest, and utilize 3 co-variates, namely the car's *Cylinder* count (cyl), *Engine Displacement* (disp), and *Acceleration* (accel). A preview of the data is provided below:

Table 9: Preview of Automobile MPG Dataset (Published 1993)

MPG	Displacement	Cylinder Count	Acceleration	Model
18	307	8	12.0	chevrolet chevelle malibu
15	350	8	11.5	buick skylark 320
18	318	8	11.0	plymouth satellite
16	304	8	12.0	amc rebel sst
17	302	8	10.5	ford torino
15	429	8	10.0	ford galaxie 500

One aspect of this dataset that makes it interestingly suite to kernel and spline regression, is that visually, there is evidence of non-linear relationship between the outcome of interest (*MPG*) and *Engine Displacement*. The relationship between *MPG*, *Cylinder* count and *Acceleration* is less clear, and shown below in a series of scatter plots:



1) Go through one-variable-at-a-time kernel- and spline-regression fits to the data for all 3 x-variables. Compare the fitting results using the leave-one-out cross-validation procedure.

Iterating over the auto-mpg set, we fit both a spline, and kernel regression model, removing one target observation, and compared the prediction error generated out of sample for each model. Summary statistics resulting from the leave-one-out computation are presented below:

Table 10: Regression Model Error Comparisons: Leave-One-Out Procedure

Statistic	Kernel	Spline
Median Error	-0.30	-0.36
Error: 75th Percentile	2.09	2.03

Statistic	Kernel	Spline
Error: 25th Percentile	-2.50	-2.45
R-Squared	0.71	0.69
Mean Absolute Percentage Error	13.05	13.26

The summary statistics from this table clearly highlight that both models perform similarly across a number of simple error statistics. While the kernel regression approach may have a slight edge across a few statistics, in general, both models provided results with symmetric, generally unbiased out-of-sample prediction error for each leave-out observation. While the leave-one-out procedure doesn't highlight any significant differences in performance, further analysis and computation below does show some of the practical benefits of and limitations of each approach to solving the regression problem of predicting a car's *MPG* based on its characteristics.

Note that computational details for the leave-one-out procedure are detailed in the code appendix.

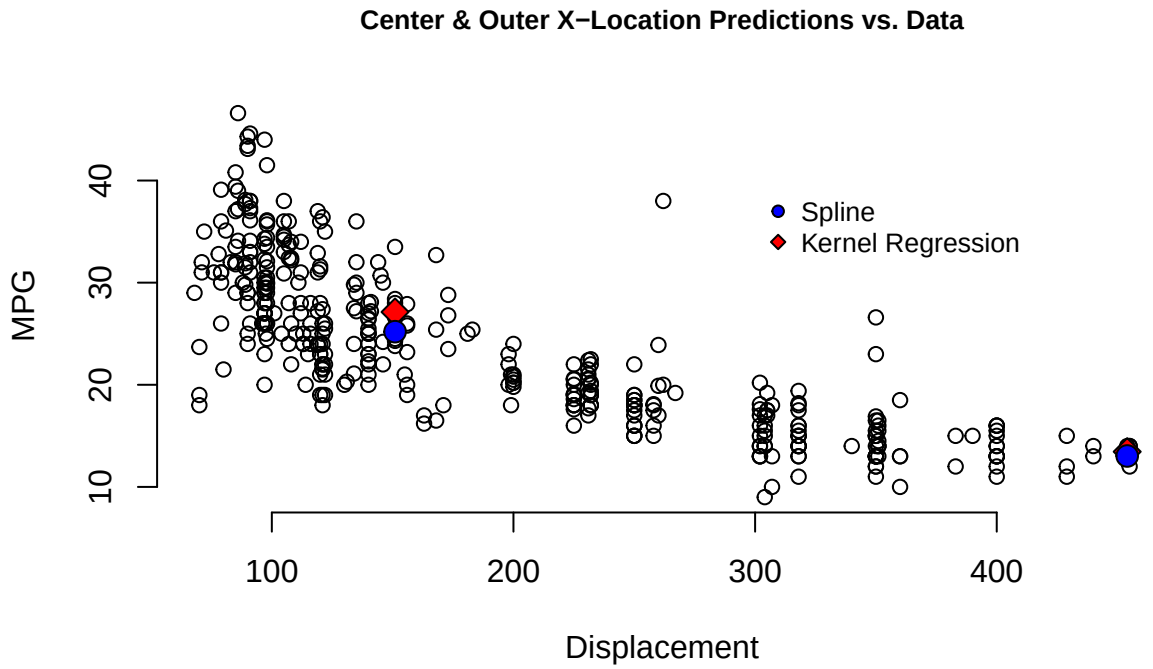
2) Select 2 sets of x-locations, e.g., (1st set: x1 = 3, x2 = 5, x3 = 2, where 3, 5, 2 are values within x-data range). These 2 sets of x-locations should be from a location close to the center of x-data-range and another location closer to the edge. Make predictions of Y at these x-data. Compare the predictions from kernel and spline methods, and also comment on the impact from x-data-locations.

To select *X*-locations, we took the median across all *X* variables, i.e. *Displacement*, *Cylinder Count*, and *Acceleration* as one *X*-location, denoted as the "Center Point" in subsequence analysis. We then also created an "Outside Point" which is based on the maximum across all *X* variables, less 1. That is a point very close to the maximum across all co-variates. The table below summarizes the exact location of these points, and also provides associated predicted values from both kernel regression and spline models that were fit to the full auto-mpg dataset:

Table 11: Comparison of Predictions from Spline & Kernel Regression Methods

Displacement	Cylinder	Acceleration	X-Location	Kernel Prediction	Spline Prediction
151	4	15.5	center point	27.13	25.20
454	7	23.8	outside point	13.45	13.03

It is clear based on the table above that the spline provides a more conservative prediction of car MPG for both the center and outside points. To get a sense of how these predictions fare relative to our actual data, the scatter plot below shows our key predictor *Displacement* on the x-axis, against our outcome of interest *MPG* on the y-axis. The predicted MPG given each model is then shown in the colored points below:



The scatter plot again shows that the spline provides a slightly more conservative predictin vs. the kernal regression. However, both predictions do lie in a reasonable place relative to actual data points observed.

3) Construct the 90% Bias-Corrected Bootstrap CIs for the predictions at the selected 2-sets of x-locations. Show details of the bias-correction process. Compare the CIs at two x-locations, and also from two nonparametric regression methods.

Continuing our analysis of prediction at our two X-locations, we conduct a bootstrap sampling analysis. The procedure conducted consists of:

1. Taking a boostrap sample, or a uniform sample of observations (selected with replacement)
2. Fitting a spline or kernel regression model to the boostrap sample data, and
3. Predicting *MPG* at the given X-locations using the models fit in step 2

To create a more robust bootstrap estimate, we used bias-correction, or *BCa*, a technique similar to a percentile approach to boostrap sampling. However the bootstrap percentiles,but adjusted to account for bias and skewness. Underlying the the *BCa* approach is a notion of acceleration and skewness informing the parameter intervals, and also a connection to the normal distribution, as *BCa* relies on the CDF of a standard normal random variable to compute percentile estimates. The results of the bootstrap analysis are summarized in the table below:

Table 12: Bias-Corrected Bootstrap CIs - 90%, By Model Type and X-location

Point	Prediction	Lower	Upper	Method
Center Point	27.13	25.88	30.47	Kernel Regression
Outside Point	13.45	-0.22	16.08	Kernel Regression
Center Point	25.20	24.73	25.62	Spline
Outside Point	13.03	10.47	15.90	Spline

Note: Instability of the kernel regression fits from the np package, and instability of fits from smooth.spline in R (base) were both problematic due to sample size, and computation speed constraints further analysis would have likely yielded more stable results.

A key point to highlight from this table, is the instability of the kernel regression fits given our dataset. Bootstrap replication actually created non-informative confidence intervals for our kernel regression model,

given poor fits of in some boostrap replications. In contrast results from the spine model were more informative, and much narrower, and perhaps precise in comparison to kernel regression. The stability of the spline model in comparison does highlight its utility in this very specific case.

The bootstrap computational details were somewhat lengthy relative to other routines, and are provided in the *Code Appendix* at the end of this document.

Code Appendix

Multi-resolution Plot Generation Example (R package: *waveslim*)

The code below demonstrates procedures to plot the additive components of a wavelet decomposition using the *waveslim* package in *R*. Note this code depends on usage of a time series exactly 1024 observations in length:

```
x <- as.numeric(aep$AEP_MW) #input megawatts time series

n.levels <- 4
aep.haar <- mra(x = x, "haar", n.levels, "dwt")
nplots <- n.levels + 2
par(mfcol=c(nplots,2), pty="m", mar=c(5-2.5,4,4-2,1.5), cex.main = 0.9)
plot.ts(x, axes=F, main="Haar Filter: Multi-Resolution Plot", ylab = 'Original', bty='n')
axis(side=1, at=seq(0,1024,by=128),
     labels=seq(0,1024,by=128))
axis(side=2, at=c(round(min(x),digits = -4),round(max(x),digits = -4)),
     labels=c(round(min(x),digits = -4),round(max(x),digits = -4)))
for(i in 1:(n.levels +1)){
  plot.ts(aep.haar[[i]], axes=F, ylab=names(aep.haar)[i],bty='n')
  axis(side=1, at=seq(0,1024,by=128),
     labels=seq(0,1024,by=128))
  axis(side=2, at=c(round(min(aep.haar[[i]]),digits = -2),round(max(aep.haar[[i]]),digits = -2)),
       labels=c(round(min(aep.haar[[i]]),digits = -2),round(max(aep.haar[[i]]),digits = -2)))
}

#family daubechies
aep.la8 <- mra(x = x, "la8", n.levels, "dwt")
plot.ts(x, axes=F, main="Daubechies Filter (L=8): Multi-Resolution Plot", ylab = 'Original', bty='n')
axis(side=1, at=seq(0,1024,by=128),
     labels=seq(0,1024,by=128))
axis(side=2, at=c(round(min(x),digits = -4),round(max(x),digits = -4)),
     labels=c(round(min(x),digits = -4),round(max(x),digits = -4)))
for(i in 1:(n.levels +1)){
  plot.ts(aep.la8[[i]], axes=F, ylab=names(aep.la8)[i],bty='n')
  axis(side=1, at=seq(0,1024,by=128),
       labels=seq(0,1024,by=128))
  axis(side=2, at=c(round(min(aep.la8[[i]]),digits = -2),round(max(aep.la8[[i]]),digits = -2)),
       labels=c(round(min(aep.la8[[i]]),digits = -2),round(max(aep.la8[[i]]),digits = -2)))
}
```

Leave-One-Out Procedure, Kernel and Spline Regression

Below is exemplary code utilized in this report to generate additive model fits for both spline, and kernel regression models, along with the leave-one-out procedure. To speed up computation, which can be slow for kernel regression, the routine is distributed across CPUs utilizing the *parallel* library in *R*:

```
#leave one out procedure - spline and kernel regression
# CPU distributed (10 cores here)
spl_res <- rbindlist(mclapply(1:nrow(adat), function(i){
  train <- adat[-i]
  test <- adat[i]
  #additive spline fits
  spl_fit_x1 <- smooth.spline(train$disp,train$mpg, cv= F,df = 3)
```

```
  spl_fit_x2 <- smooth.spline(train$cyl,residuals(spl_fit_x1), cv= F,df = 3)
  spl_fit_x3 <- smooth.spline(train$accel,residuals(spl_fit_x2), cv= F,df = 3)
  test <- transform(test, prediction = predict(spl_fit_x1, data.frame(test$disp))$y +
                      predict(spl_fit_x2, data.frame(test$cyl))$y + predict(spl_fit_x3, data.frame(test$
  test$error <- test$mpg - test$prediction #actual vs. estimate
  test
}, mc.cores=10))

#this takes 1-2 minutes with 10 cores
kr_res <- rbindlist(mclapply(1:nrow(adat), function(i){
  train <- adat[-i]
  test <- adat[i]
  #additive kernel reg fits - simplified predict if data.frame is called
  kr_fit_x1 <- npreg(mpg~disp,gradients = TRUE, data = train)
  train$e1 <- residuals(kr_fit_x1)
  kr_fit_x2 <- npreg(e1~cyl,gradients = TRUE, data = train)
  train$e2 <- residuals(kr_fit_x2)
  kr_fit_x3 <- npreg(e2~accel,gradients = TRUE, data = train)
  test$prediction <- predict(kr_fit_x1, newdata = data.frame(disp = test$disp)) +
                      predict(kr_fit_x2, newdata = data.frame(cyl = test$cyl)) +
                      predict(kr_fit_x3, newdata = data.frame(accel = test$accel))
  test$error <- test$mpg - test$prediction #actual vs. estimate
  test
}, mc.cores=10))
```

Bootstrap Confidence Intervals

Below is exemplary code utilized in this report to generate boostrap confidence intervals using the boot package, relying on the *boot* and *np* packages. Bootstrap sampling procedures were CPU distributed where possible to speed up computation:

```
B <- 10000 #bootstrap replication count
#estimator function
adat <- readRDS('data/auto_mpg.rds')
cpt <- data.table(disp = median(adat$disp),cyl = median(adat$cyl),
                  accel = median(adat$accel), type = 'center point')
opt <- data.table(disp = max(adat$disp) - 1,cyl = max(adat$cyl)-1,
                  accel = max(adat$accel) - 1, type = 'outside point')
spl_boot_cpt <- function(data, indices){
  #indices <- sample(1:nrow(adat),replace = T)
  bs_dat <- data[indices]
  sfit_x1 <- smooth.spline(bs_dat$disp,bs_dat$mpg, cv= F,df = 3)
  sfit_x2 <- smooth.spline(bs_dat$cyl,residuals(sfit_x1), cv= F,df = 3)
  sfit_x3 <- smooth.spline(bs_dat$accel,residuals(sfit_x2), cv= F,df = 3)
  prediction <- as.numeric(predict(sfit_x1, data.frame(cpt$disp))$y +
                             predict(sfit_x2, data.frame(cpt$cyl))$y +
                             predict(sfit_x3, data.frame(cpt$accel))$y)
  prediction
}

spl_boot_opt <- function(data, indices){
  #indices <- sample(1:nrow(adat),replace = T)
  bs_dat <- data[indices]
```

```
  sfit_x1 <- smooth.spline(bs_dat$disp,bs_dat$mpg, cv= F,df = 3)
  sfit_x2 <- smooth.spline(bs_dat$cyl,residuals(sfit_x1), cv= F,df = 3)
  sfit_x3 <- smooth.spline(bs_dat$accel,residuals(sfit_x2), cv= F,df = 3)
  prediction <- as.numeric(predict(sfit_x1, data.frame(opt$disp))$y +
                             predict(sfit_x2, data.frame(opt$cyl))$y +
                             predict(sfit_x3, data.frame(opt$accel))$y)
  prediction
}

spl_cpt_res <- boot(adat, statistic = spl_boot_cpt,
                    R=B)
spl_opt_res <- boot(adat, statistic = spl_boot_opt,
                    R=B)
#bootstrap results - spline
cpt_results <- boot.ci(boot.out = spl_cpt_res,type = 'bca',conf = 0.9)
opt_results <- boot.ci(boot.out = spl_opt_res,type = 'bca',conf = 0.9)

boot_spl_bca <- data.table(
  Point = c('Center Point','Outside Point'),
  Prediction = c(cpt_results[[2]],opt_results[[2]]),
  Lower = c(cpt_results$bca[4],opt_results$bca[4]),
  Upper = c(cpt_results$bca[5],opt_results$bca[5]))



kr_boot_opt <- function(data, indices){
  #indices <- sample(1:nrow(adat),replace = T)
  kdat <- data[indices]
  kfit_x1 <- npreg(mpg~disp,gradients = TRUE, data = kdat)
  kdat$e1 <- residuals(kfit_x1)
  kfit_x2 <- npreg(e1~cyl,gradients = TRUE, data = kdat)
  kdat$e2 <- residuals(kfit_x2)
  kfit_x3 <- npreg(e2~accel,gradients = TRUE, data = kdat)
  prediction <- predict(kfit_x1, newdata = data.frame(disp = opt$disp)) +
    predict(kfit_x2, newdata = data.frame(cyl = opt$cyl)) +
    predict(kfit_x3, newdata = data.frame(accel = opt$accel))
  prediction
}

kr_boot_cpt <- function(data, indices){
  #indices <- sample(1:nrow(adat),replace = T)
  kdat <- data[indices]
  kfit_x1 <- npreg(mpg~disp,gradients = TRUE, data = kdat)
  kdat$e1 <- residuals(kfit_x1)
  kfit_x2 <- npreg(e1~cyl,gradients = TRUE, data = kdat)
  kdat$e2 <- residuals(kfit_x2)
  kfit_x3 <- npreg(e2~accel,gradients = TRUE, data = kdat)
  prediction <- predict(kfit_x1, newdata = data.frame(disp = cpt$disp)) +
    predict(kfit_x2, newdata = data.frame(cyl = cpt$cyl)) +
    predict(kfit_x3, newdata = data.frame(accel = cpt$accel))
  prediction
}
```

```
kr_cpt_res <- boot(adat, statistic = kr_boot_cpt,
                   R=B,parallel = 'multicore',
                   ncpus = 11)
kr_opt_res <- boot(adat, statistic = kr_boot_opt,
                   R=B,parallel = 'multicore',
                   ncpus = 11)

#bootstrap results - kernel regression
cpt_kresults <- boot.ci(boot.out = kr_cpt_res,type = 'bca',conf = 0.9)
opt_kresults <- boot.ci(boot.out = kr_opt_res,type = 'bca',conf = 0.9)

#results summary
boot_kr_bca <- data.table(
  Point = c('Center Point','Outside Point'),
  Prediction = c(cpt_kresults[[2]],opt_kresults[[2]]),
  Lower = c(cpt_kresults$bca[4],opt_kresults$bca[4]),
  Upper = c(cpt_kresults$bca[5],opt_kresults$bca[5]))

boot_kr_bca$Method <- 'Kernel Regression'
boot_spl_bca$Method <- 'Spline'
boot_sum <- rbindlist(list(boot_kr_bca,boot_spl_bca))
```