Baseline Algorithm Draft #1.1: locally Weighted Regression

Miles Bennett

September 14, 2018

### Local Regression Model Algorithm

The data used to train is attached in the email. please install tidyverse and lubridate first *The code below may be used to download and read in data from local file (please install tidyverse and lubridate)*

setwd("C:/Users/Miles/Desktop/COI")  
data <- read.csv("meter\_C30545.csv")  
library(tidyverse)  
library(lubridate)

### Data Subset for Training (after settting up connection this may/probably not necessary)

# necessary to easily parse date\_time variable later

data$date\_time <- as.character(data$date\_time)  
data$date\_time <- dmy\_hms(data$date\_time)  
  
  
#Test/Train subset  
aug17\_aug23 <- data %>%   
 filter(date\_time >= "2018-08-17 00:00:00" & date\_time <= "2018-08-23 03:59:00")%>%  
 select(date\_time, kwh\_diff)

### Data Pre-Processing Necessary to Develop Model

*For the model to determine which data to train, it will need the ‘event’ column below to populate in environment*

# Extracting Date Objects from date\_time also we can easily add more variables   
day <- as.data.frame(as.factor(as.integer(wday(aug17\_aug23$date\_time, label = FALSE))))#levels(day) "Sun" "Mon" "Tue" "Wed" "Thu" "Fri" "Sat" = 1:7   
month <- as.data.frame(month(aug17\_aug23$date\_time, label = FALSE))  
hour <- as.data.frame(hour(aug17\_aug23$date\_time))  
hour <- hour + 1  
minute <- as.data.frame(minute(aug17\_aug23$date\_time))  
minute <- minute + 1  
year <- as.data.frame(as.factor(year(aug17\_aug23$date\_time)))  
  
  
# Creating event variable, necessary to develop control flows for algorithm. Should populate in the model environment  
event <- as.data.frame(rep(0, times = nrow(aug17\_aug23))) #new data is a column of 0's and 1's

### TEST CONTROL FLOW 1

#########################TEST#####################  
###Prior to test set up ran code blocks 1:3  
  
#####Test 1 for control Flow 1  
  
aug17 <- data %>%   
 filter(date\_time >= "2018-08-17 00:00:00" & date\_time <= "2018-08-17 12:00:00" )%>%  
 select(date\_time, kwh\_diff)  
nrow(aug17)  
  
event\_17 <- rep(1, times = nrow(aug17))  
non\_event <- rep(0, times = (nrow(aug17\_aug23) - length(event\_17)))  
length(event\_17) + length(non\_event) == nrow(aug17\_aug23)  
  
event <- as.data.frame(c(event\_17, non\_event))  
  
####################### END TEST######################

### Properly Structured Data for Model: M0

# Combining variables into data-frame and naming columns  
structured\_data <- cbind(aug17\_aug23, day, month, hour, minute, year, event)  
colnames(structured\_data) <- c("date\_time", "kwh\_diff", "day", "month", "hour", "minute", "year", "event")  
  
  
#visualize data frame   
head(structured\_data, 10)

## date\_time kwh\_diff day month hour minute year event  
## 1 2018-08-17 04:00:00 0.72 6 8 5 1 2018 0  
## 2 2018-08-17 04:01:00 0.72 6 8 5 2 2018 0  
## 3 2018-08-17 04:02:00 0.72 6 8 5 3 2018 0  
## 4 2018-08-17 04:03:00 0.36 6 8 5 4 2018 0  
## 5 2018-08-17 04:04:00 0.72 6 8 5 5 2018 0  
## 6 2018-08-17 04:05:00 0.72 6 8 5 6 2018 0  
## 7 2018-08-17 04:06:00 0.72 6 8 5 7 2018 0  
## 8 2018-08-17 04:07:00 0.72 6 8 5 8 2018 0  
## 9 2018-08-17 04:08:00 0.72 6 8 5 9 2018 0  
## 10 2018-08-17 04:09:00 0.72 6 8 5 10 2018 0

### Control Flow 1 to determine appropriate subset of data to train model on (all non curtailment event days)

if(any(structured\_data$event == "1")){  
 train <- subset(structured\_data, event =="0")  
} else {   
 train <- structured\_data  
}  
  
#Visualize current training data  
head(train)

## date\_time kwh\_diff day month hour minute year event  
## 1 2018-08-17 04:00:00 0.72 6 8 5 1 2018 0  
## 2 2018-08-17 04:01:00 0.72 6 8 5 2 2018 0  
## 3 2018-08-17 04:02:00 0.72 6 8 5 3 2018 0  
## 4 2018-08-17 04:03:00 0.36 6 8 5 4 2018 0  
## 5 2018-08-17 04:04:00 0.72 6 8 5 5 2018 0  
## 6 2018-08-17 04:05:00 0.72 6 8 5 6 2018 0

### Training the Model

For capacity/demand kW should be the dependent variable.

Currently this model should provide an estimated value for each given day at that particular minute of the day in the case of a curtailment event, for example if an event happened on Friday it would use the all previous Friday’s and average the amount of kw power usage at each particular minute to estimate the load of a customer on a non-curtailment day to determine the baseline for the customer. This model may be the most accurate for baseline as we can include other variables such as temperature-humidity index to increase the accuracy of the model.

#Trained Model  
M0 <- loess(kwh\_diff ~ as.numeric(day) + as.numeric(hour) + as.numeric(minute), data = train[, c(2, 3, 5, 6)], span = 0.75, degree = 2, family = “symmetric”)  
summary(M0)

## Call:  
## loess(formula = kwh\_diff ~ as.numeric(day) + as.numeric(hour) +   
## as.numeric(minute), data = train[, c(2, 3, 5, 6)])  
##   
## Number of Observations: 8880   
## Equivalent Number of Parameters: 17.72   
## Residual Standard Error: 0.2455   
## Trace of smoother matrix: 21.49 (exact)  
##   
## Control settings:  
## span : 0.75   
## degree : 2   
## family : symmetric  
## surface : interpolate cell = 0.2  
## normalize: TRUE  
## parametric: FALSE FALSE FALSE  
## drop.square: FALSE FALSE FALSE

### Test Objects (sun == 1 : sat == 7)

#test data Objects  
sunday <- train[, c(3, 5, 6)] %>% filter(day == "1")  
test\_sunday <- unique(sunday)  
  
monday <- train[, c(3, 5, 6)] %>% filter(day == "2")  
test\_monday <- unique(monday)  
  
tuesday <- train[, c(3, 5, 6)] %>% filter(day == "3")  
test\_tuesday <- unique(tuesday)  
  
wednesday <- train[, c(3, 5, 6)] %>% filter(day == "4")  
test\_wednesday <- unique(wednesday)  
  
thursday <- train[, c(3, 5, 6)] %>% filter(day == "5")  
test\_thursday <- unique(thursday)  
  
friday <- train[, c(3, 5, 6)] %>% filter(day == "6")  
test\_friday <- unique(friday)  
  
saturday <- train[, c(3, 5, 6)] %>% filter(day == "7")  
test\_saturday <- unique(saturday)

### Control Flow 2 Determines which day to provide baseline estimates given accurate values for event variable

*The baseline estimate for each day and the estimate put into a data frame with its test set should be returned given a curtailment event were to take place that day*

#Control flow 2 for test data   
if(any(structured\_data$event == "1" & day == "1")) {  
 sunday\_baseline <- as.data.frame(predict(M0, newdata = test\_sunday))  
 output <-cbind(test\_sunday, sunday\_baseline)  
 colnames(output) <- c("day", "hour", "minute", "estimate")  
} else if(any(structured\_data$event == "1" & day == "2")) {   
 monday\_baseline <- as.data.frame(predict(M0, newdata = test\_monday))  
 output <- cbind(test\_monday, monday\_baseline)  
 colnames(output) <- c("day", "hour", "minute", "estimate")  
} else if(any(structured\_data$event == "1" & day == "3")) {   
 tuesday\_baseline <- as.data.frame(predict(M0, newdata = test\_tuesday))  
 output <- cbind(test\_tuesday, tuesday\_baseline)  
 colnames(output) <- c("day", "hour", "minute", "estimate")  
} else if(any(structured\_data$event == "1" & day == "4")) {   
 wednesday\_baseline <- as.data.frame(predict(M0, newdata = test\_wednesday))  
 output <- cbind(test\_wednesday, wednesday\_baseline)  
 colnames(output) <- c("day", "hour", "minute", "estimate")  
} else if(any(structured\_data$event == "1" & day == "5")) {   
 thursday\_baseline <- as.data.frame(predict(M0, newdata = test\_thursday))  
 output <- cbind(test\_thursday, thursday\_baseline)  
 colnames(output) <- c("day", "hour", "minute", "estimate")  
} else if(any(structured\_data$event == "1" & day == "6")) {  
 friday\_baseline <- as.data.frame(predict(M0, newdata = test\_friday))  
 output <- cbind(test\_friday, friday\_baseline)  
 colnames(output) <- c("day", "hour", "minute", "estimate")  
} else if(any(structured\_data$event == "1" & day == "7")) {  
 saturday\_baseline <- as.data.frame(predict(M0, newdata = test\_saturday))  
 output <- cbind(test\_saturday, saturday\_baseline)  
 colnames(output) <- c("day", "hour", "minute", "estimate")  
} else {  
 fit\_all <- as.data.frame(predict(M0, train[, c(2, 3, 5, 6)]))  
 output <- cbind(train, fit\_all)  
 colnames(output) <- c("date\_time", "kwh\_diff", "day", "month", "hour", "minute", "year", "event", "estimate")  
}

### 

output %>%  
 ggplot(aes(x = date\_time, y = estimate)) + geom\_line()

