Spectral Time Series Analysis of Building Energy Data

Arash Khalilnejad

December 19, 2016

true

Introduction

Data streams from buildings and other sources will be used to explore relationships among independent and dependent variables, uncovering correlations, patterns and anomalies and revealing unique characteristics and behavior associated with specific buildings.

Data analytics applied to buildings have been used in the past to measure the energy savings associated with building retrofits and efficiency programs.

The investigation of power consumption data of buildings can reveal the hidden facts about its efficiency and can make a decent view of the possible improvements in better energy flow of the building. For this purpose, I will try to analyze the power consumption of a bunch of buildings. This investigation seems to be challenging knowing some facts listed below:

The available data are just active power consumption and there is not any evidence on reactive power consumption, so, the operation of equipment with motor drives is difficult to track. The electrical measurements of the building such as voltage amplitude and angle, voltage swings, sags, faults, flickers, interruptions, etc are not given. So, extracting the building power consumption operation without these data is difficult. The power consumption data may not be in the exact 15 minutes intervals. On the other hand, some of the data might be missing. The interpretation can be even more difficult knowing the fact that the measurement equipment might have some errors. Along with several procedures of data analysis to these data, the spectral time series analysis will be focused on this project. Although, the features such as trend, seasonal behavior and error terms can be derived by conventional time series analysis techniques, the focus of spectral analysis is to extract the features of the system in frequency domain. I assume that this method can give more reliable facts about the consumption trend.

Required Libraries

(data.table) (zoo) (extremevalues) (lubridate) (stringr) (ggplot2) (gridExtra) (imputeTS) (chron) (timeDate) (dplyr) (imputeTS) (xts) (forecast)

Process of Analysis

First step is reading the data and subsetting one of the point IDs to work with. Then we change the unit of measurement to factor. The date is converted to POSIXct class. Findig the time interval can make the code dynamic for any time interval. Next steps are:

-Rounding the date to unique interval
-Removing the duplicate data
-Converteing to zoo format
-Making sequnce of time in unique interval
-Merging the time sets to have the NA points in time
-Convert zoo to data table -Plot -Excluding the power data to fill the NAs with previous amounts -Replacing NAs of discriptive data with previous values
-roundata and newtime.z is useless
-Combindig the ENergy data and sorting the data appropriately
-And initial assumptions

-Iterative study on imputation

After preparing the dataset,it is ready for imputation. For that purpose, more than 30 iterations are being analyzed. In each step , the chunk of missing data from 1 to 24 hours which is 4 to 96 consecutive missing points for 15 minutes interval is evaluated and the results is given in terms of RMSE and detecting the switch on and off points.So, three methods of imputation are used which is based on filling the missing points with average of same points in 2 and 4 days in the neighbourhood and the average of same points in same days in 4 neighbour weeks. The code below is for the mentioned method:

After imputation a classical time series analysis is performed, then, the spectral time series with corresponding codes and evaluations are given.

Data book

Parameter	Definition
Facility ID	A building with several equipnets
Point ID	Point with Equipment in a Facility
NumberNAs	Number of missing points
PercentageNAs	Percentage missing points
naGapLongest	Maximum number of consecutive missing points
naGapMostOverallNAs	Most overall consecutive missing points
interval.minute	missing interval
PID	Number of Point IDs
sdata	Data splited to each point ID
MeterReordedTimeStampUTC	The time in POSIXct format
n	Time interval
rounddate	equally intervaled time
fdata	data dragged to fixed time interval
newtime	fixed time interval
fdata.z	fdata in zoo format

Parameter	Definition
fulldata	The data set with all variables
tsdata	the subset of data for time series analysis
Flag	if 1, the point is imputed, otherwise 0
meta	the subset of data for chunk missing points
Miss.Date	The data which points are missing
First.week	the imputation from the first week
Gather.week	gathering all the imputation points
MSE.itr	Data of MSE in iterative form
day.count	number of days for chunk imputations
MatchPercent	The accuracy of imputation in detecting

Code Book

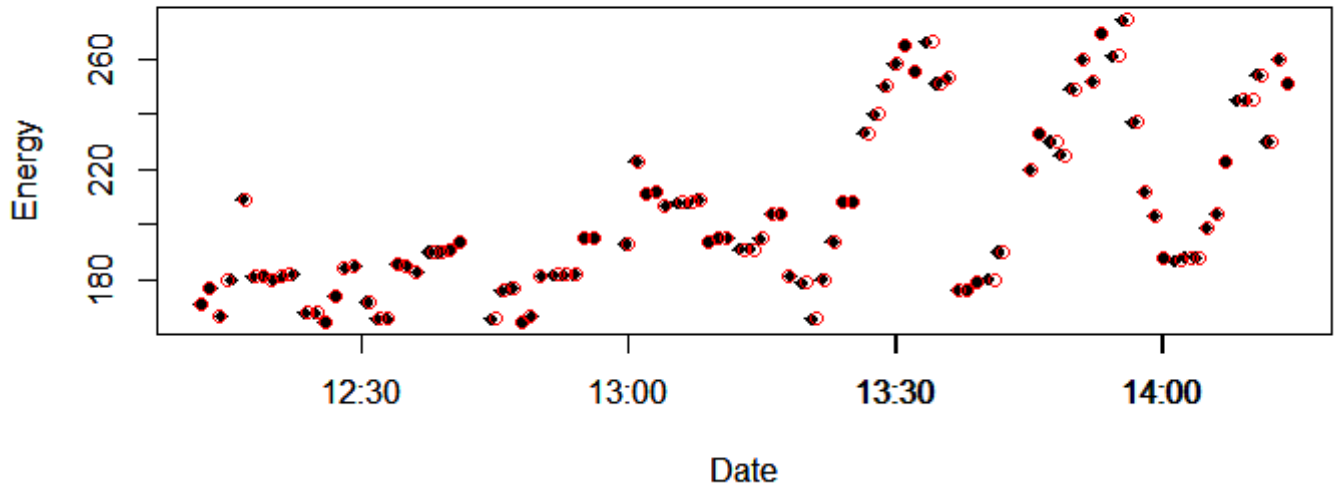
Table below is the functions that are used for this project, other than regular and simple functions for subseting and arranging dataset.

Code	Definition
na.interpolation	Interpolate the missing points with simple linear model
getOutliersI	wrapper function for getOutliersI
filterTSeriesSSA	Decompose a vector (i.e. time series) into spectral bands
ssa	Create a new SSA object
reconstruct	Perform a series reconstruction
spec.pgram	Estimate Spectral Density of a Time Series by a Smoothed Periodogram
wcor	Calculate the W-correlation matrix
strptime	Date-time Conversion Functions to and from Character
round_date	Round, floor and ceiling methods for date-time objects.
zoo	Z’s Ordered Observations
merge	merge zoo objects
rowSums	Sum values of Raster objects by row or column.
aggreagate	aggregation of spatial objects
ggplot	initializes a ggplot object.

Cleaning Data

Time interval

The data set does not have equal intervals. To do so, the points are dragged to the nearst uniform time point.

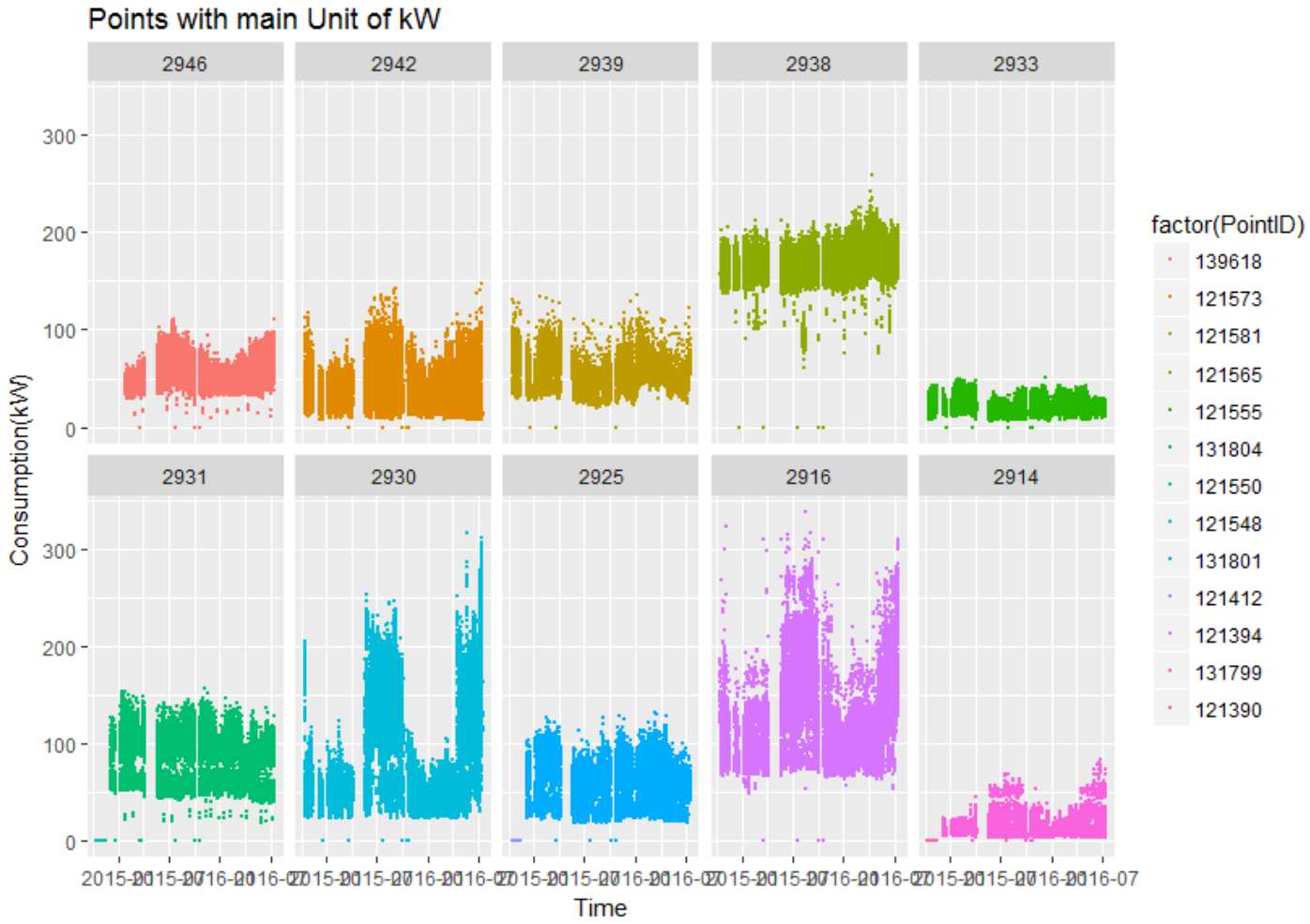


Dragging the points to unique time interval

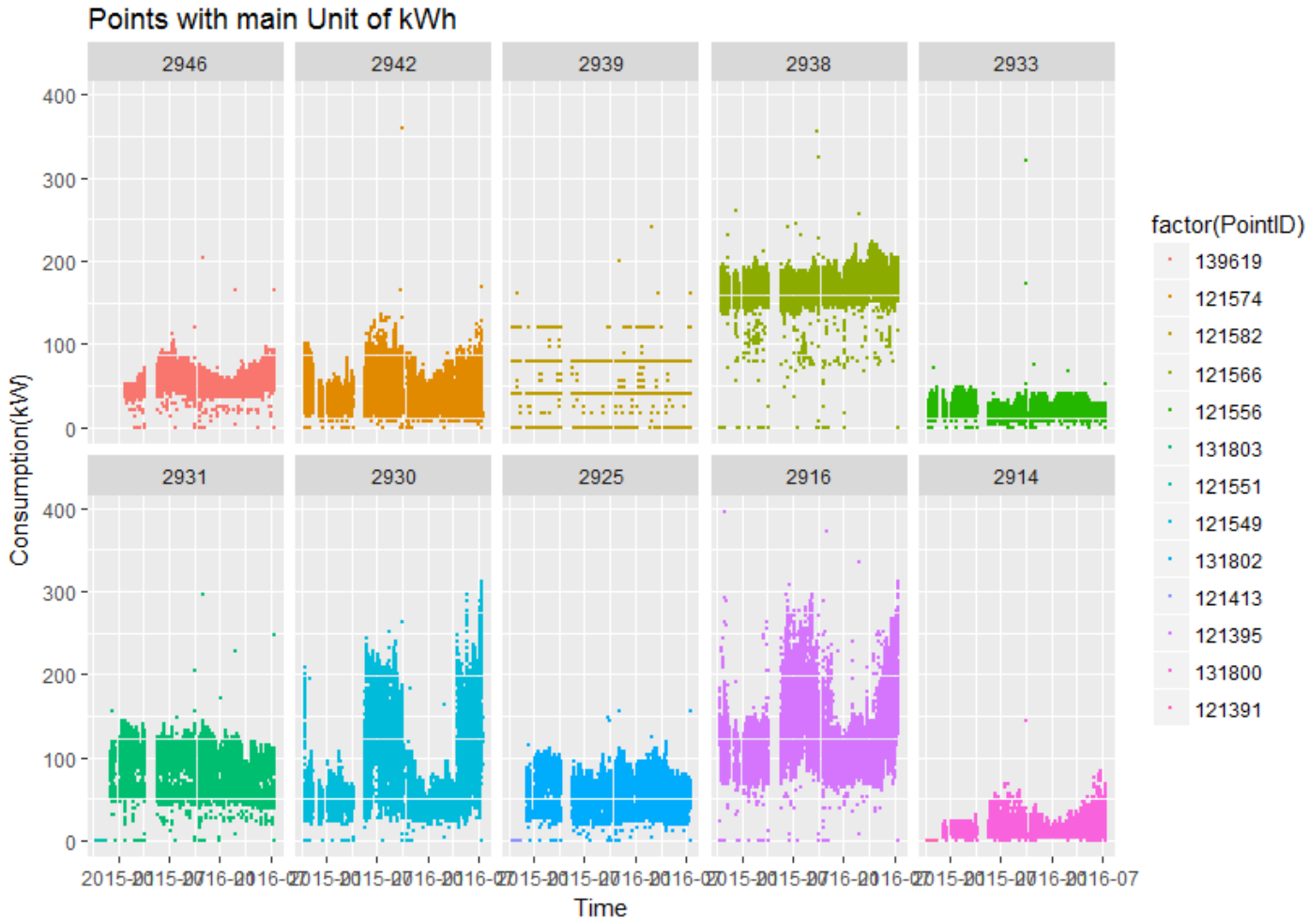
Unit of Data

There are two major units in the meters as “kW” and “kWh”. The units should be changed to unique format, which I choosed to be kW.

This is the data set for a set of buildings in Kent which have two measurements in kW and KWh. After converteing the data set unit of kWh to kW it can be seen that both meters are measuring the same equipment.



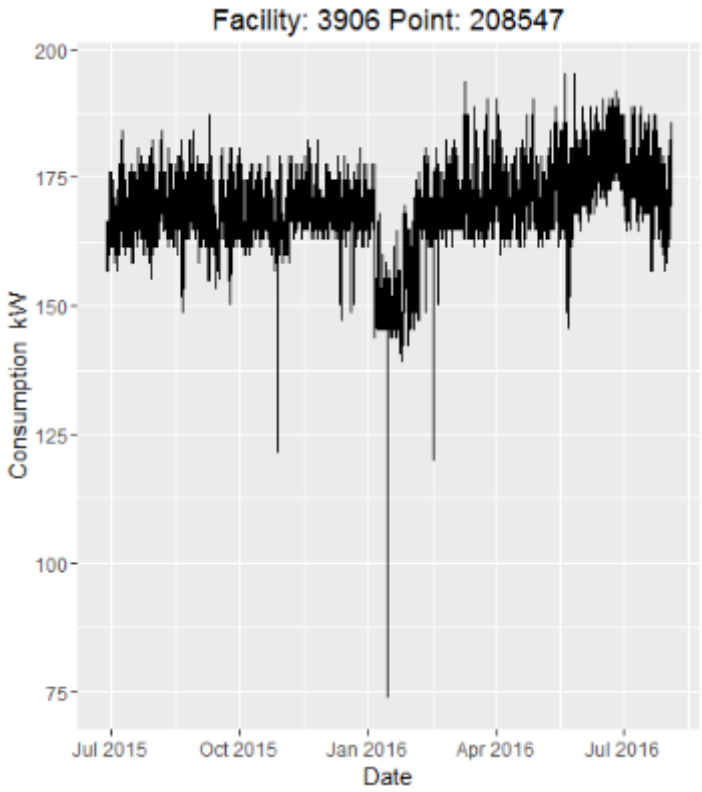
Consumption Data of Kent Buildings in kW



Consumption Data of Kent Buildings in kW converted from kWh

Outliers

The data has obviously some ouliters. Figure below has some of them that we want to take them out.

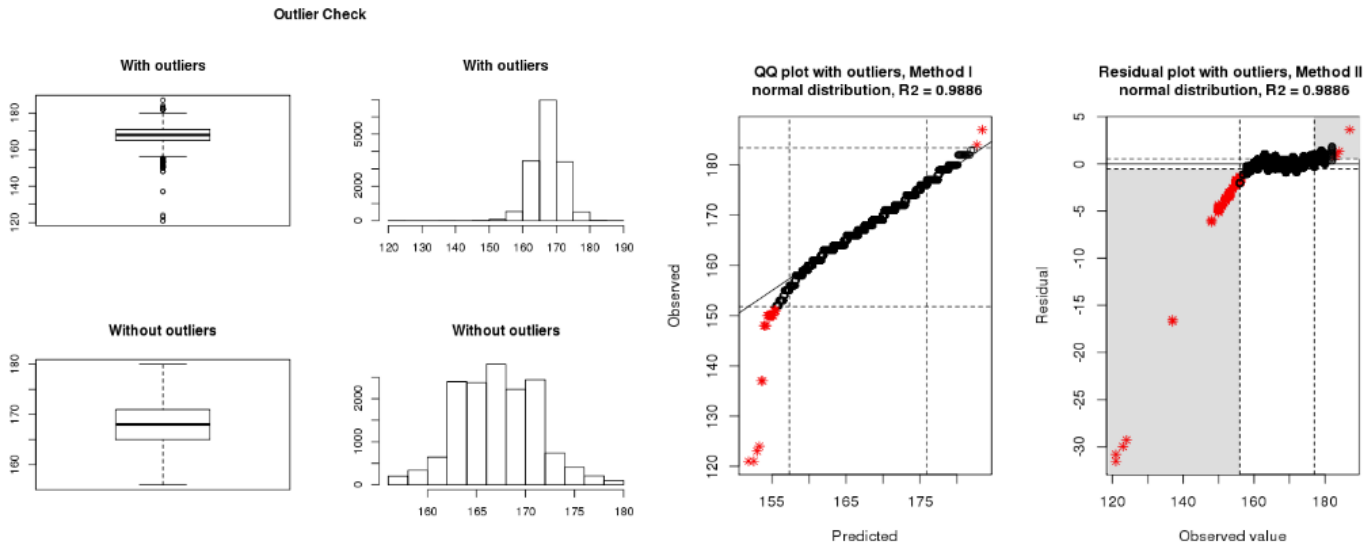


Data set with outliers

The left figure is for taking the outlier based on the points above and below the first and third quartiles which we will lose a lot data that necessarily they are not missing.

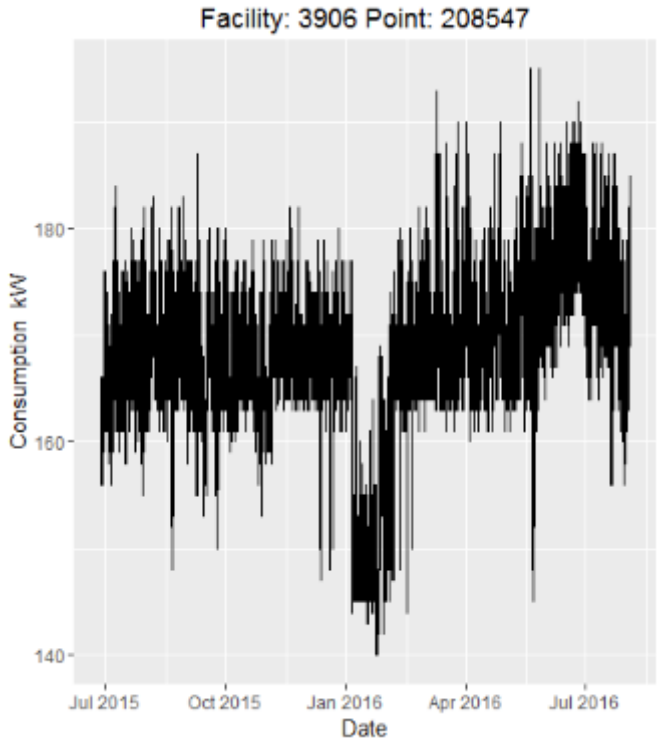
So, the other methd based on interval of disribution is given.

L <- getOutliersI(Energy,rho=c(1,1), FLim=c(0.01,0.99),distribution="normal")
outlierPlot(tsdata.noNA$Energy,L,mode="qq") Energy[which(Energy<L$limit[1]),] <-NA
Energy[which(Energy>L$limit[2]),] <-NA



Outlier treatment

The data set after taking out outliers is shown below.

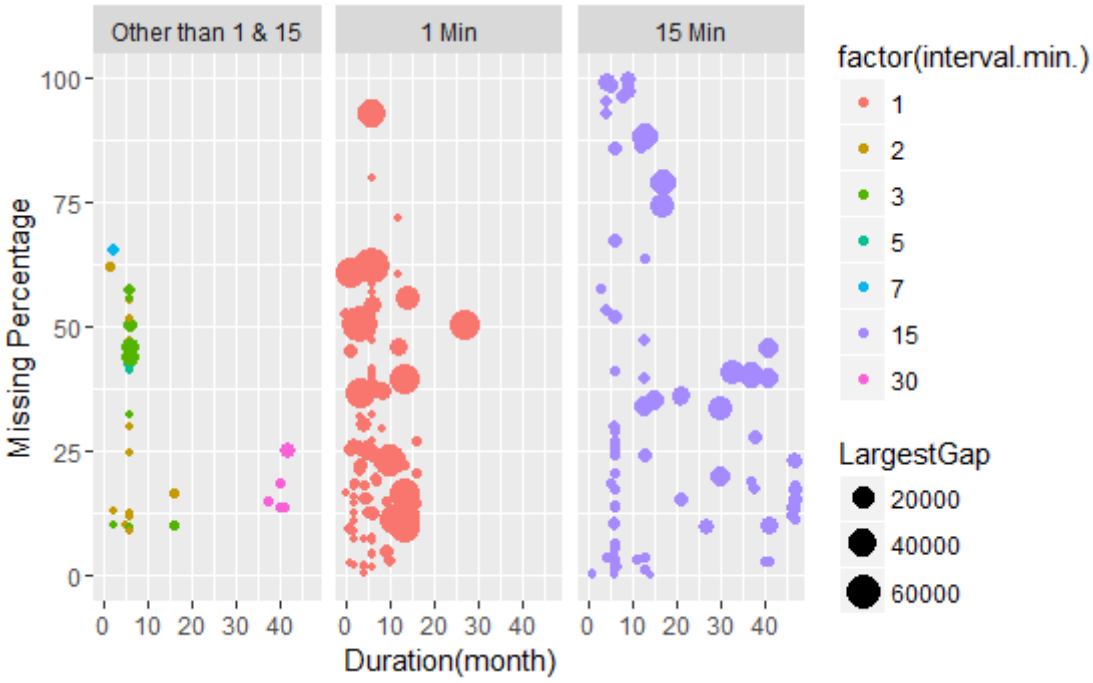


Dataset without outliers

Missing Data

```
##    V1 Facility.ID Point.ID Duration.Month. Length NumberNAs percentageNAs
## 1:  1        2914   121390               1   3115        14        0.449%
## 2:  2        2914   121391               1   3115        12        0.385%
## 3:  3        2914   131799              19  56477      7878         13.9%
## 4:  4        2914   131800              19  56477      7830         13.9%
## 5:  5        2916   121394              21  61567      9818         15.9%
## 6:  6        2916   121395              21  61567      9817         15.9%
##    naGapLongest naGapMostFrequent naGapMostOverallNAs interval.minute.
## 1:            4                 1                   4               15
## 2:            4                 1                   4               15
## 3:         4141                 1                4141               15
## 4:         4141                 1                4141               15
## 5:         4141                 1                4141               15
## 6:         4141                 1                4141               15
```

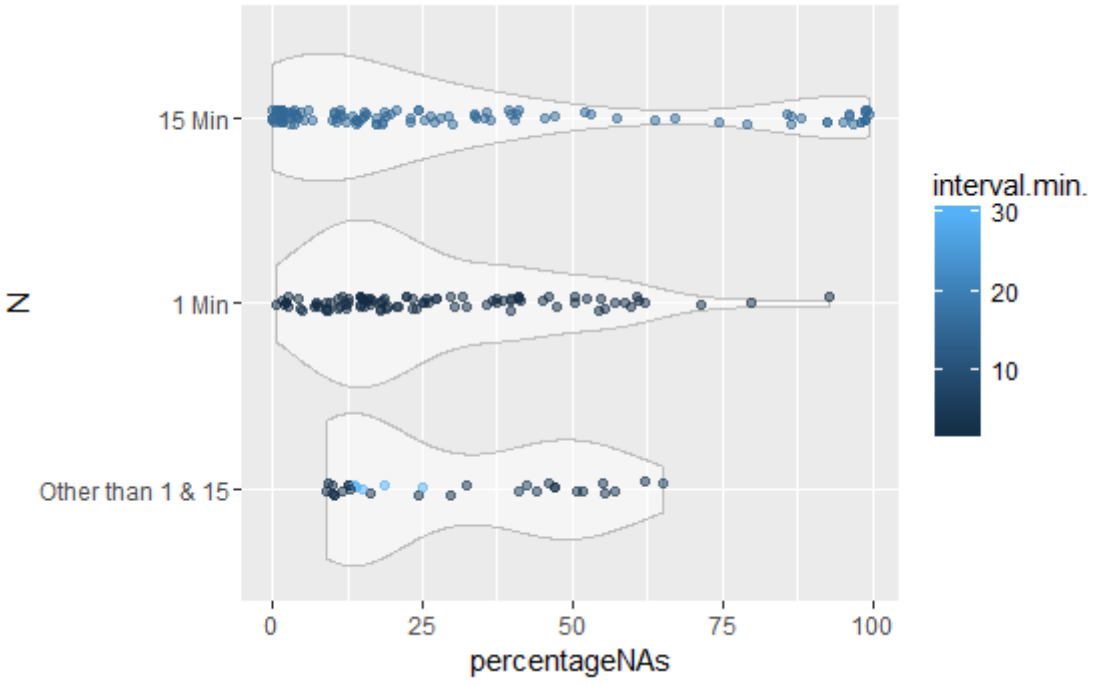
The figure below shoes the missing percentage in each data set. The size of each point shows the gap in the data set and the x axis is the dataset duration.



Missing Points

As it can be seen, we have a lot of missing data. In missing data, 30 minutes in pink are 12 to 25 percent missing points which are single points, and in almost 40 month purid, with 30 minutes interval. Also, the one minute data set are mostly less than one year period.

The figure below shoews the missing percentage distiribution. As it can be seen 15 minutes data have the most rate of missing percentage above 40.

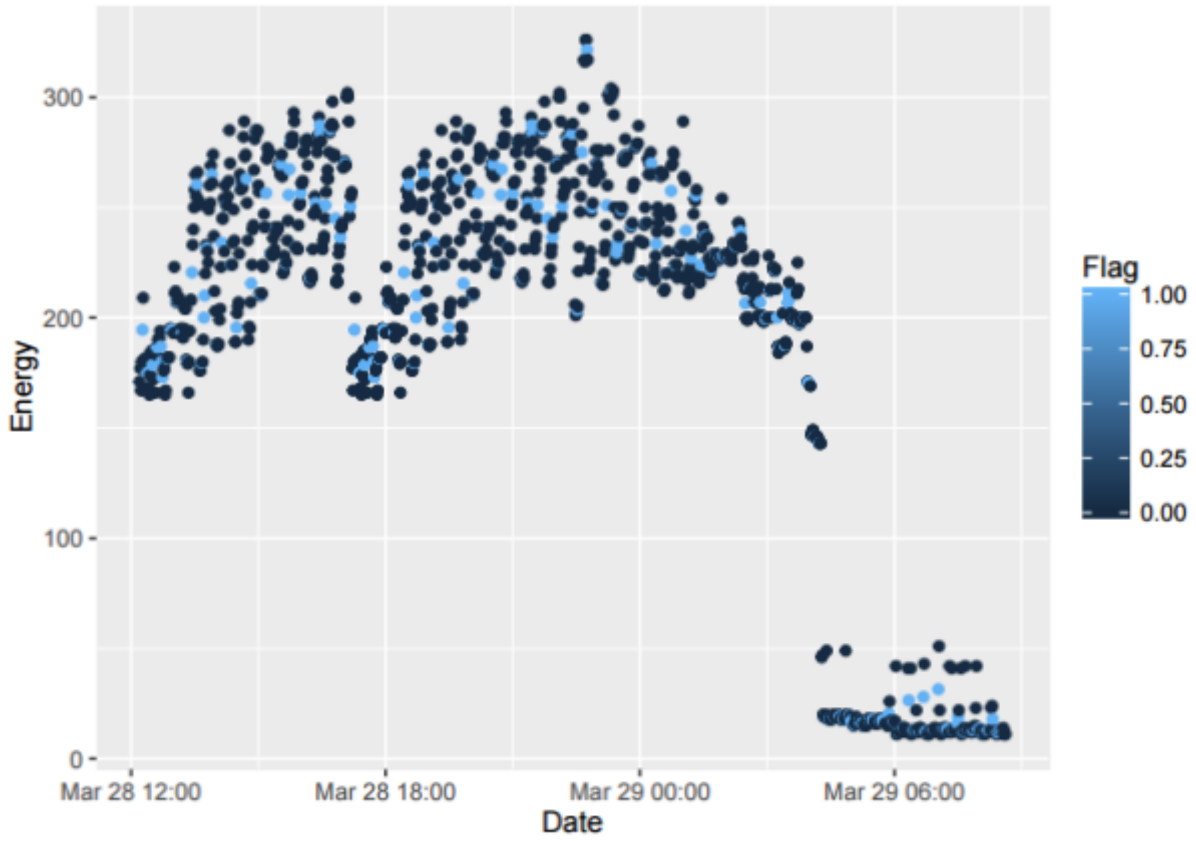


Missing points

In missing points we have two issues, first the single points and then, chunk missing points.

Single point missing

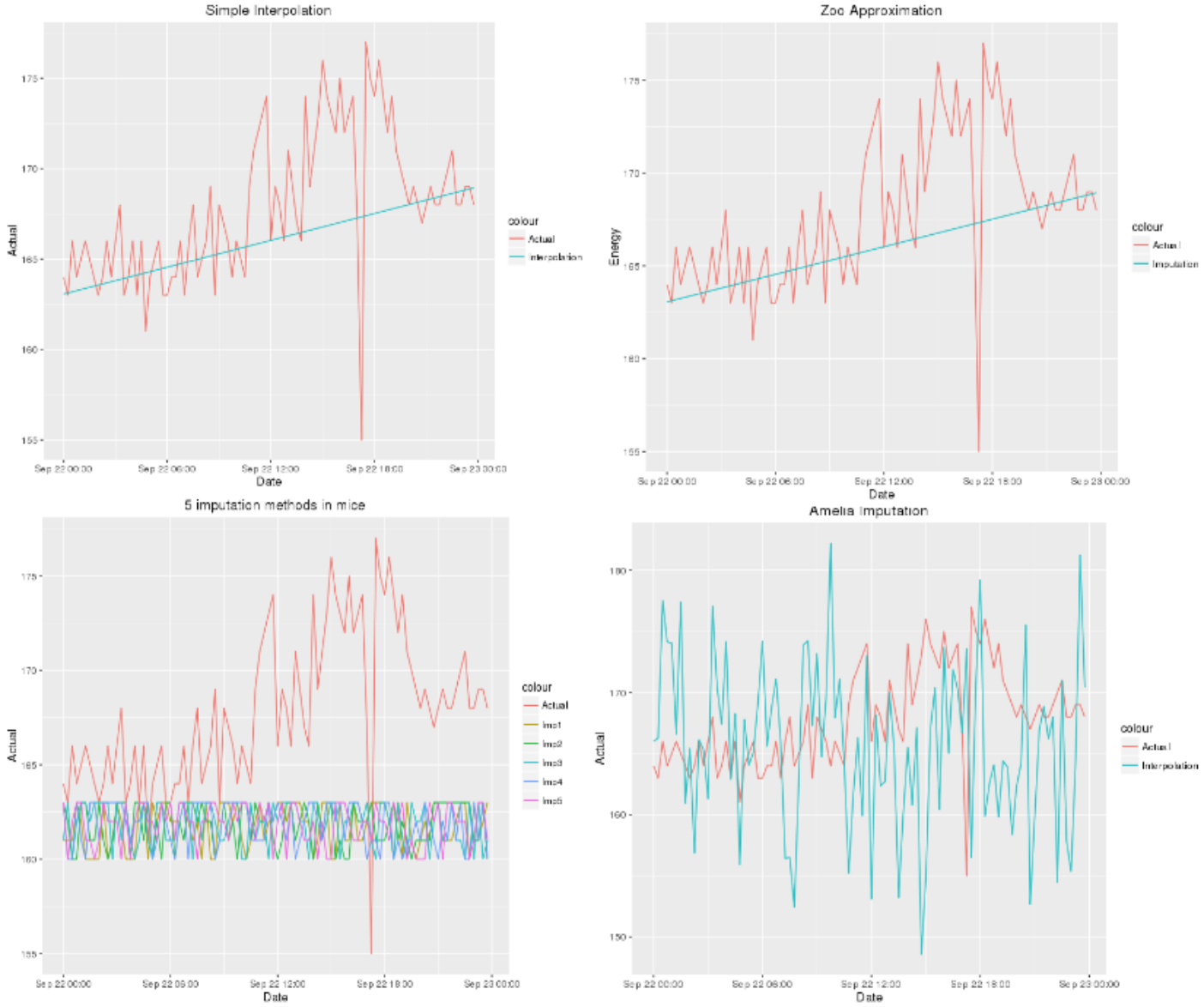
As discussed in report 2, the simle interpolation is a reliable methodology of imputation



Simple Imputation

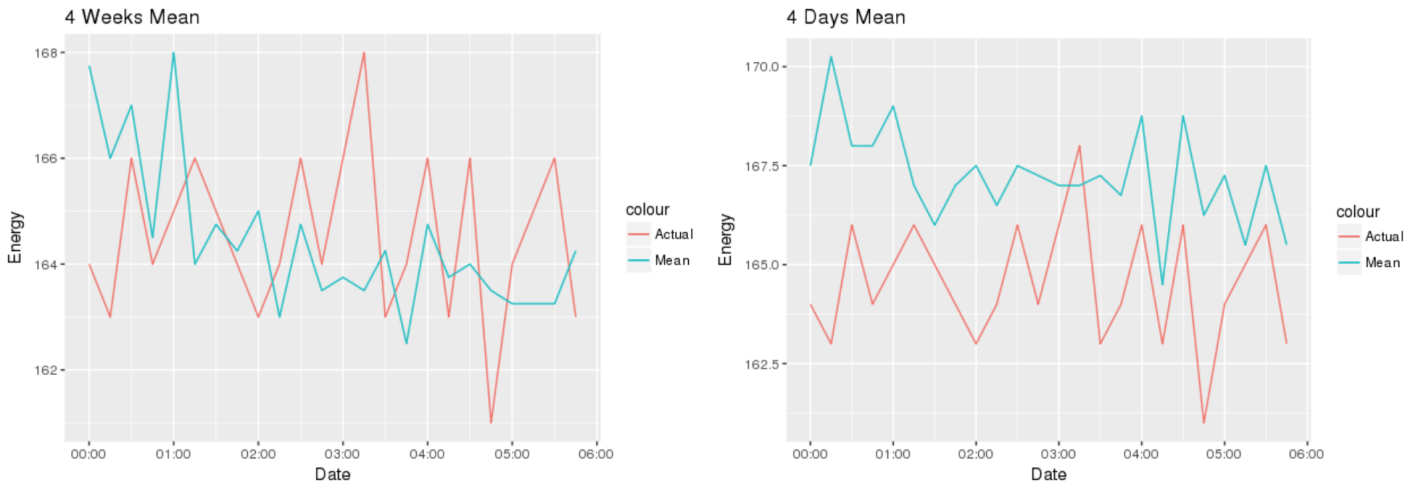
Chunk missing data

The other issue is chunk and bulk missing points. As shown in below figures, the r packages cannot solve this problem.



Imputation methods on bulk data

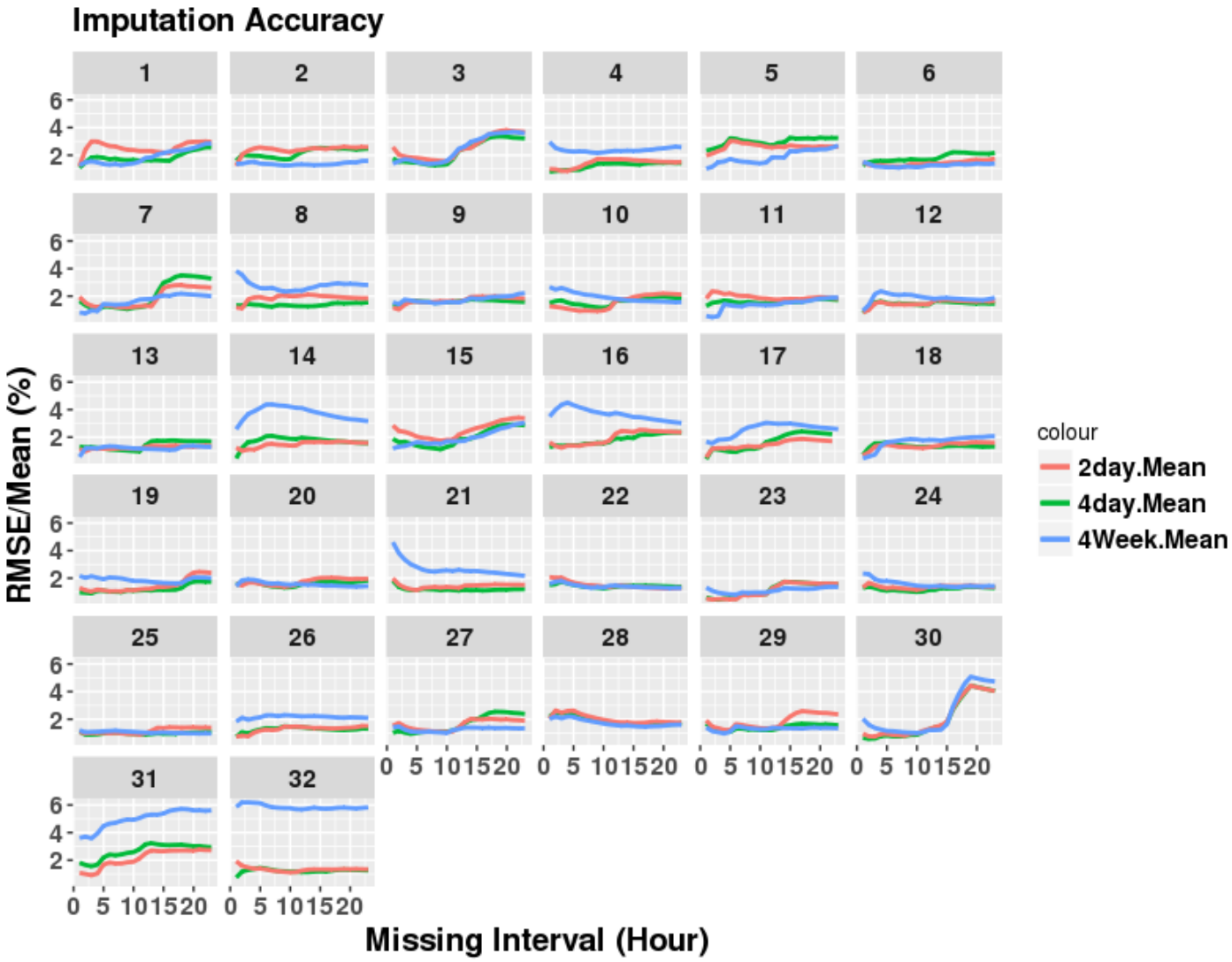
So, we used the neighbour days or weeks in fill in the missing points.



Imputation methods on bulk data

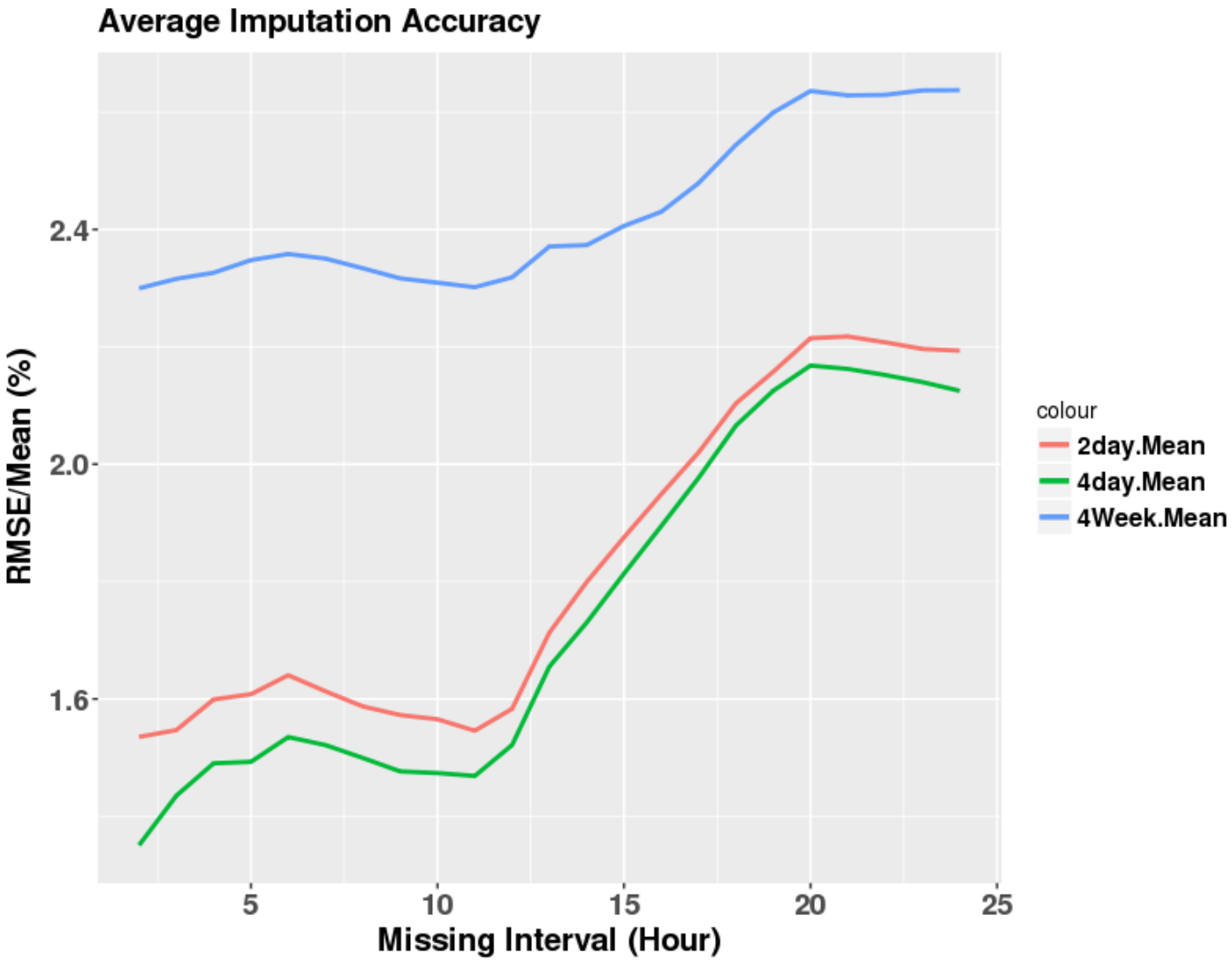
Which method is better

For that purpose, an iterative study on a data set can be helpful. T know which method is good, three different techniques of imputation based on 2 and 4 neighbour days, and 4 neighbour weeks are analyzed in more than thirty itterations.



Iterative study in Imputation methods on bulk data

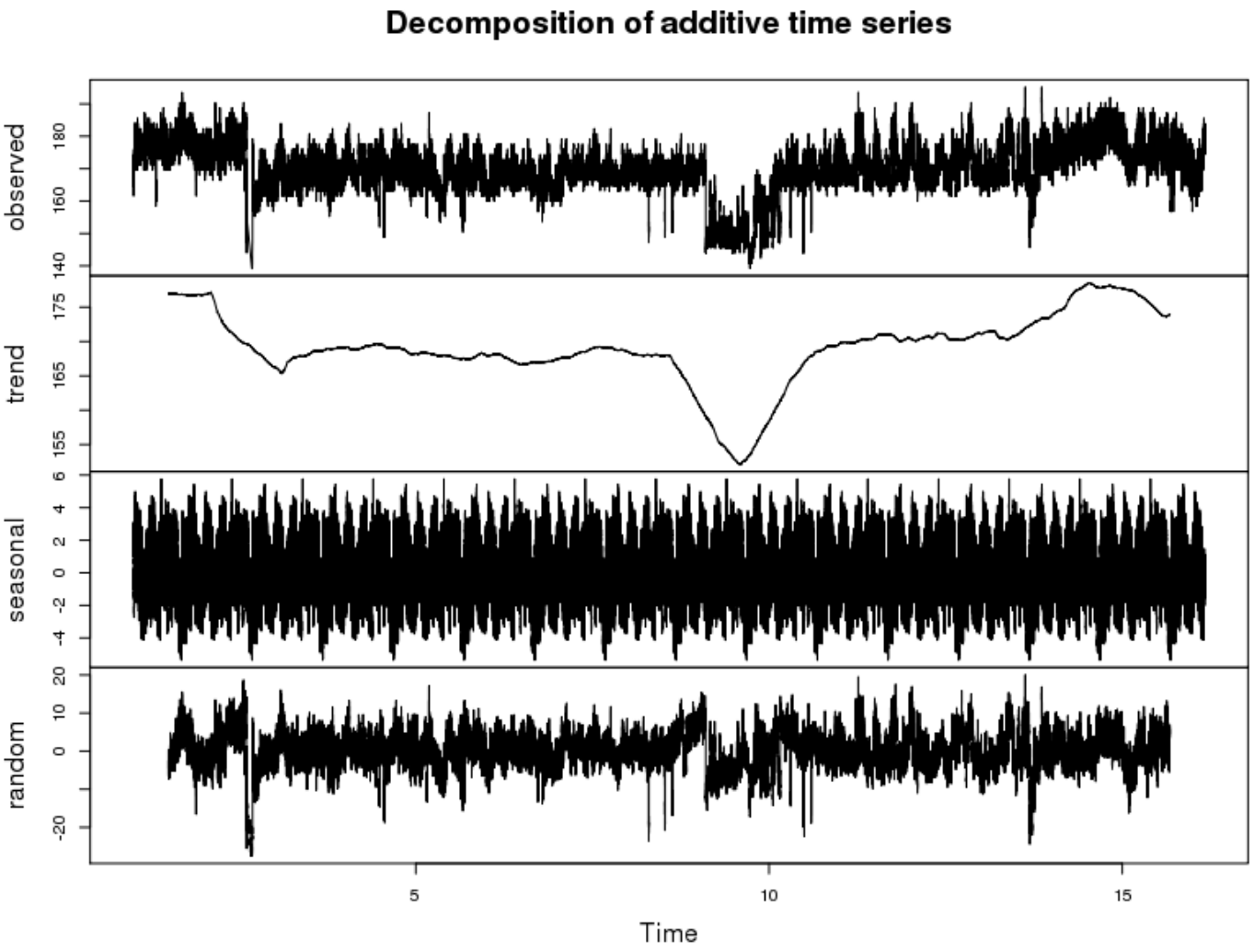
Claculating the sqaure root of mean square error, we can know how much accurate these methods are, which shows that 4 day missing are more accurate but what if we are on Friday and the neighbourday is Satureday whcih the behaviours of those days are extremely different.



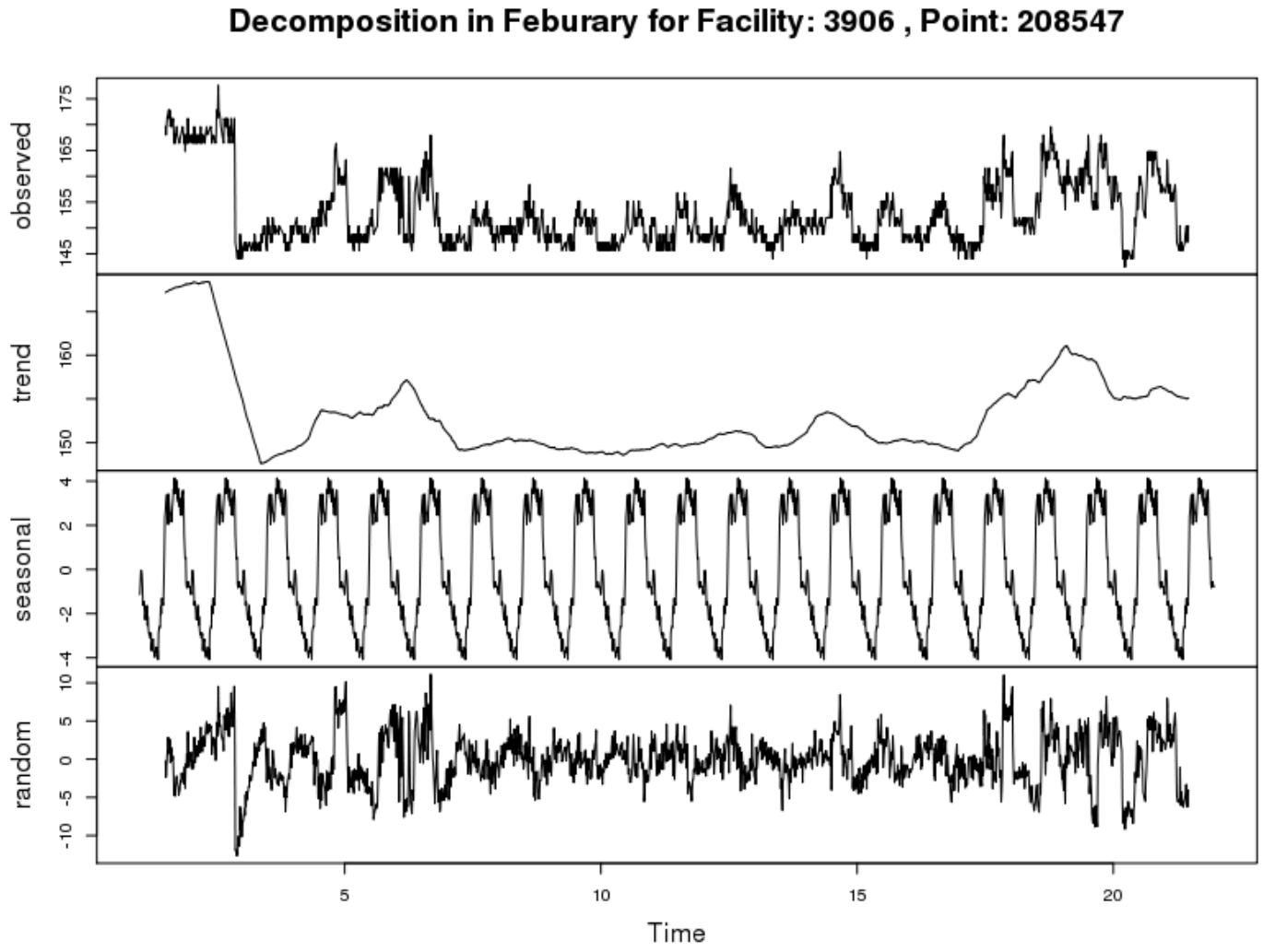
The average of th iterations shows that, the 4 day average has the best accuracy i terms of imputatio accuracy and changing point match and two day mean has the least changing point accuracy. This is because the metorological condition is so close to eachother in neighbourdays. However, it should be mentioned that, the uncertainities can increase in the imputation with average days, as the trend of the operation in weekends are totaly different from the other days. So, as 4 neighbour weeks average give reasonable error accuracy of 2.4 percent, we could accept that, to take care of both accuracy and noise. As it can be seen, the error for increasing the hours from one day to 5 days is almost fixed ad this is because, the single points are judged not based on their close points which could be subject to error, but based on the single points i the other weeks.

Classical Time series

The classical time series decomposition, the trend, seasonality, and error terms can be extracted. As shown below for yearly data set and a month of the sam data.



Time Series Decomposition with monthly seosenality



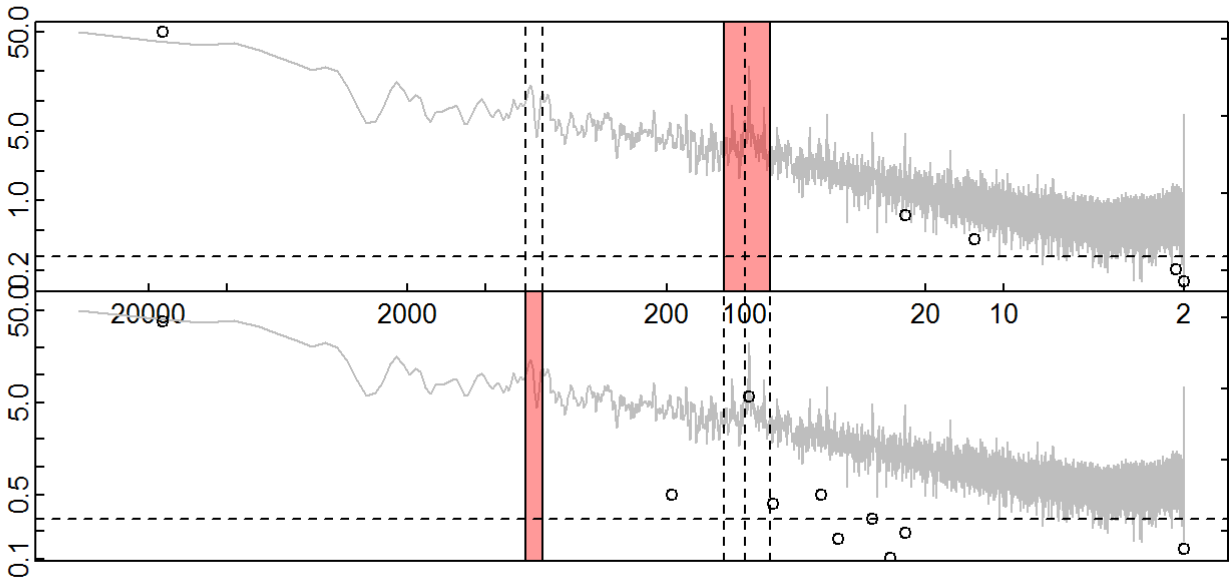
Time Series Decomposition with Daily seasonality

Spectral Analysis

In linear algebra, an eigenvector or characteristic vector of a linear transformation is a non-zero vector that does not change its direction when that linear transformation is applied to it. More formally, if T is a linear transformation from a vector space V over a field F into itself and v is a vector in V that is not the zero vector, then v is an eigenvector of T if T(v) is a scalar multiple of v.

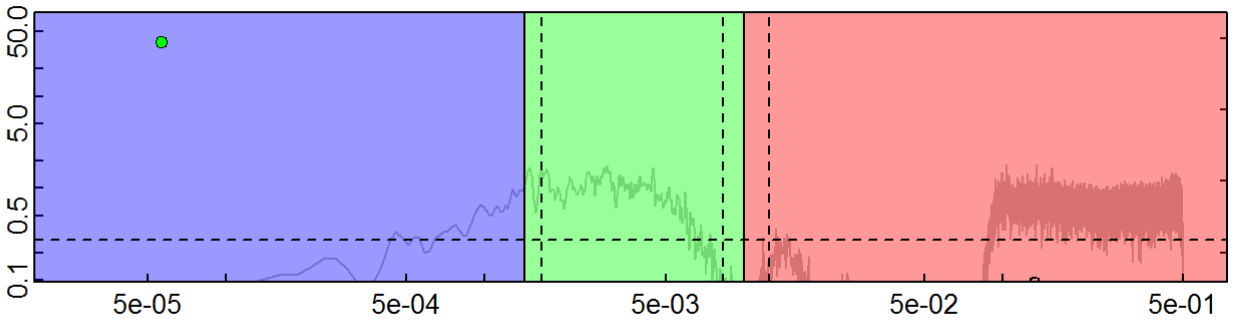
```
FALSE 2016-12-20 11:20:42 : Spectral decomposition: Preparing calculations.
FALSE 2016-12-20 11:20:42 : Spectral decomposition: Starting step 1 of 3.
FALSE 2016-12-20 11:20:42 : Spectral decomposition: Running SSA.
FALSE 2016-12-20 11:20:43 : Spectral decomposition: Determining frequencies.
FALSE 2016-12-20 11:20:43 : Spectral decomposition: Summing groups in wavlength bands.
FALSE 2016-12-20 11:20:43 : Spectral decomposition: Plotting pseudospectrum.
```

```
FALSE 2016-12-20 11:20:44 : Spectral decomposition: Starting step 2 of 3.
FALSE 2016-12-20 11:20:44 : Spectral decomposition: Running SSA.
FALSE 2016-12-20 11:20:46 : Spectral decomposition: Determining frequencies.
FALSE 2016-12-20 11:20:46 : Spectral decomposition: Summing groups in wavlength bands.
FALSE 2016-12-20 11:20:46 : Spectral decomposition: Plotting pseudospectrum.
```

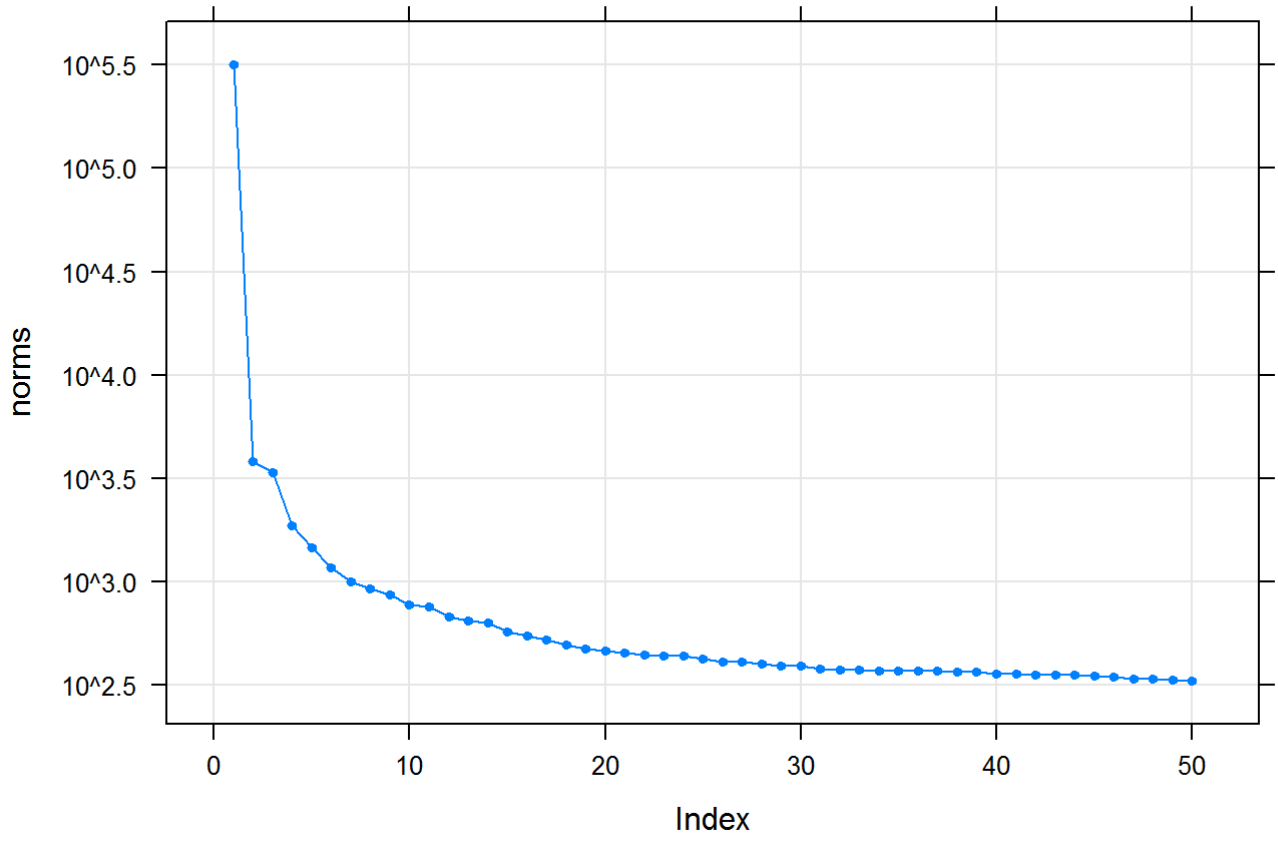



```
FALSE 2016-12-20 11:20:46 : Spectral decomposition: Starting step 3 of 3.
FALSE 2016-12-20 11:20:46 : Spectral decomposition: Running SSA.
FALSE 2016-12-20 11:20:48 : Spectral decomposition: Determining frequencies.
FALSE 2016-12-20 11:20:48 : Spectral decomposition: Summing groups in wavlength bands.
FALSE 2016-12-20 11:20:51 : Spectral decomposition: Determining frequencies.
FALSE 2016-12-20 11:20:51 : Spectral decomposition: Summing groups in wavlength bands.
FALSE 2016-12-20 11:20:54 : Spectral decomposition: Determining frequencies.
FALSE 2016-12-20 11:20:55 : Spectral decomposition: Summing groups in wavlength bands.
FALSE 2016-12-20 11:20:55 : Spectral decomposition: Plotting pseudospectrum.
```

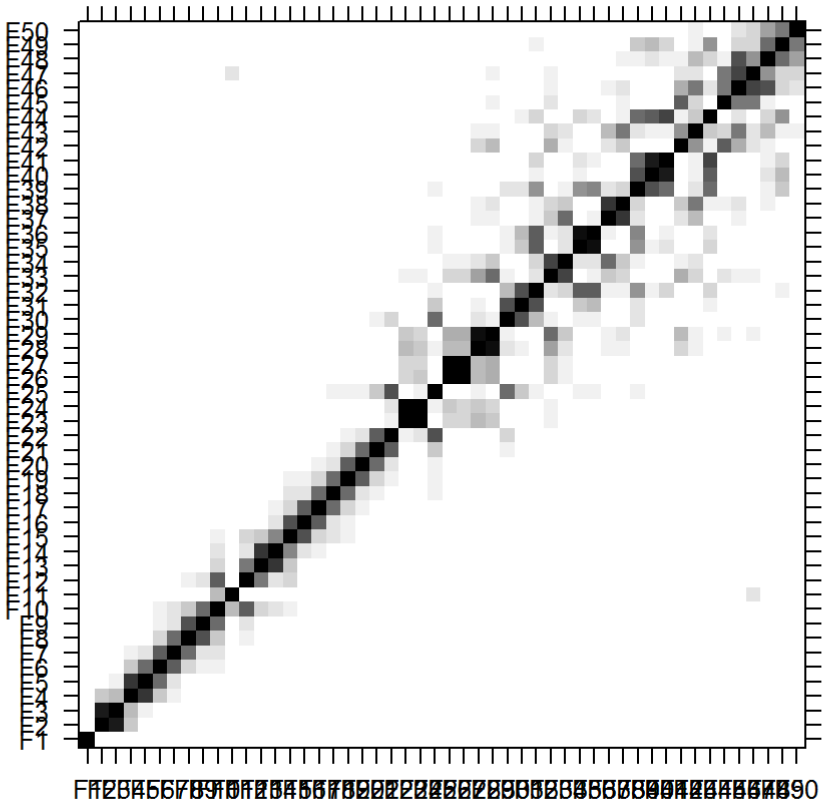
```
FALSE 2016-12-20 11:20:55 : Spectral decomposition: Process finished!
```



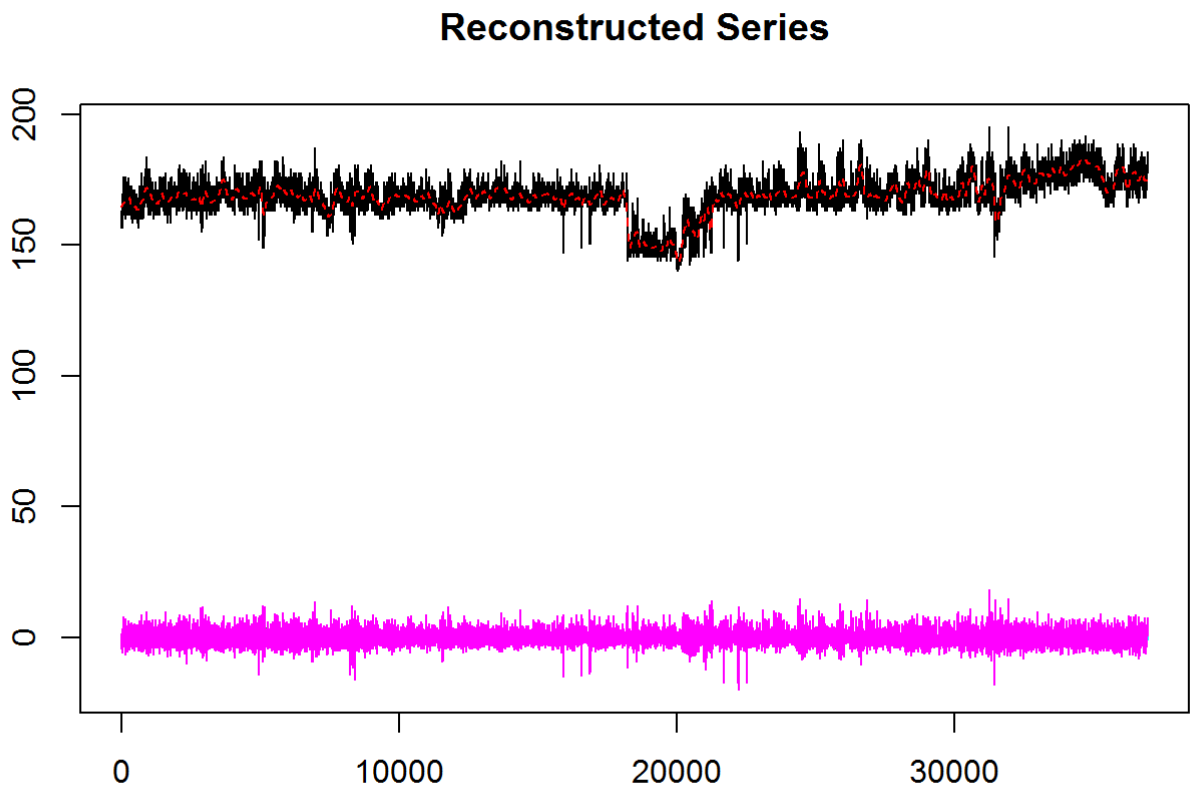

```
s <- ssa(Data, L=96)
plot(s)
```



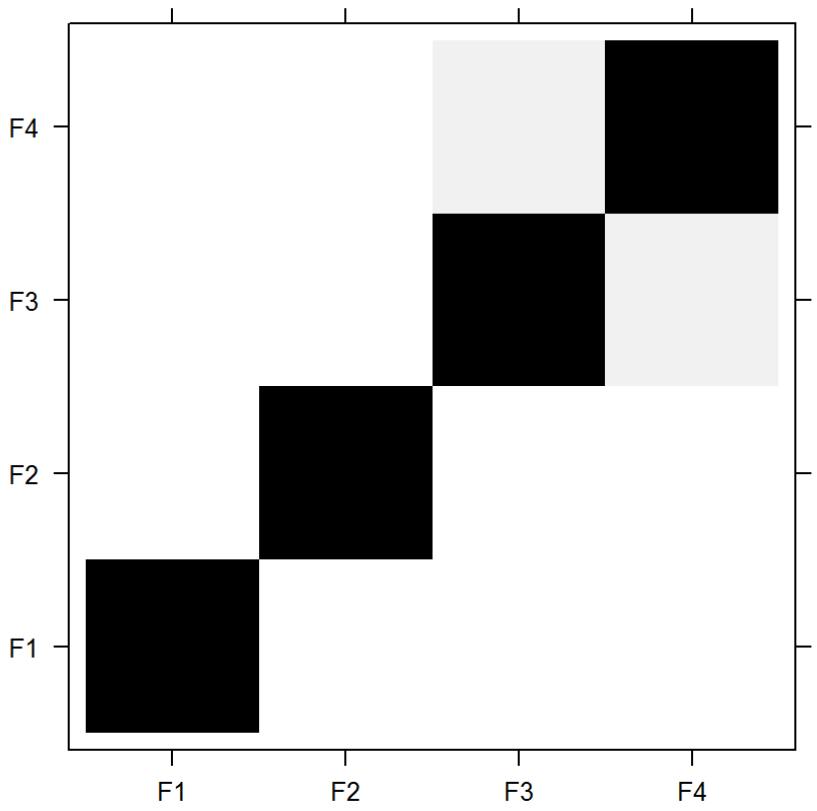

```
#W-correlation matrix

plot(wcor(s))
```



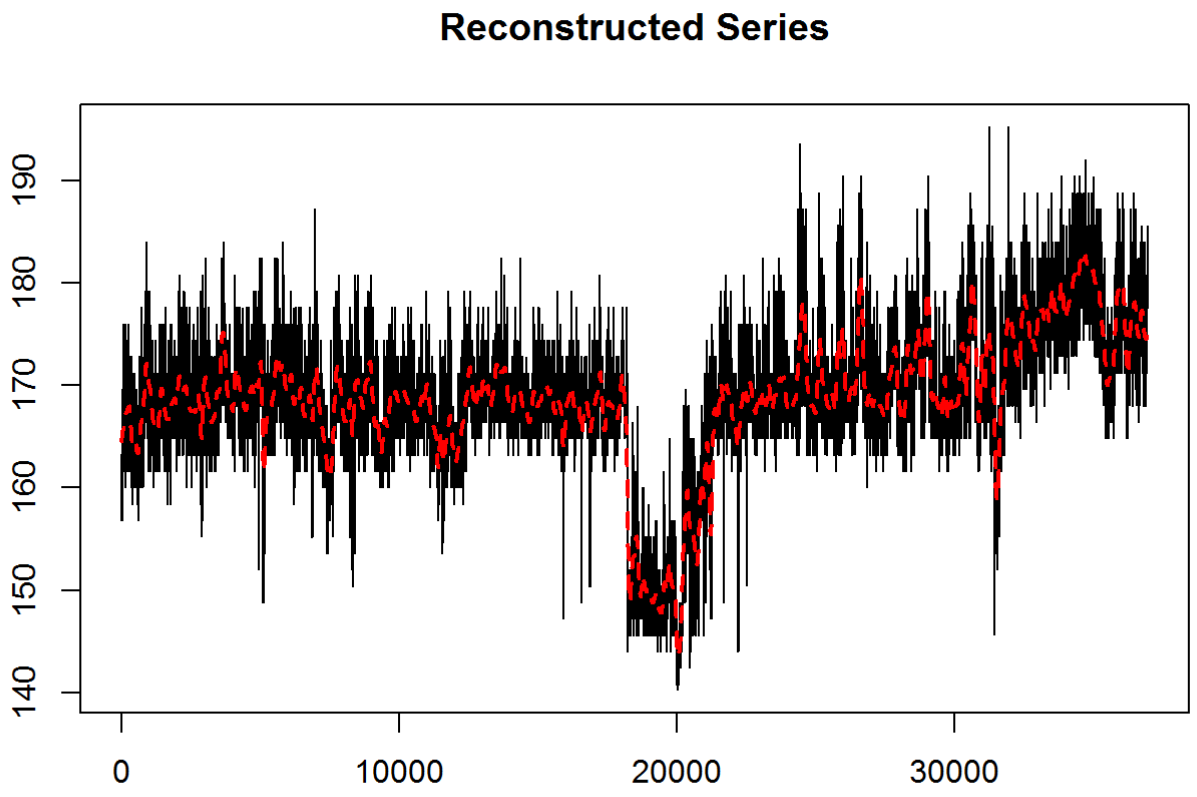

```
recon <- reconstruct(s, groups = list(c(1),c(2,24), c(25,32), c(33,50)))

plot(recon)
```



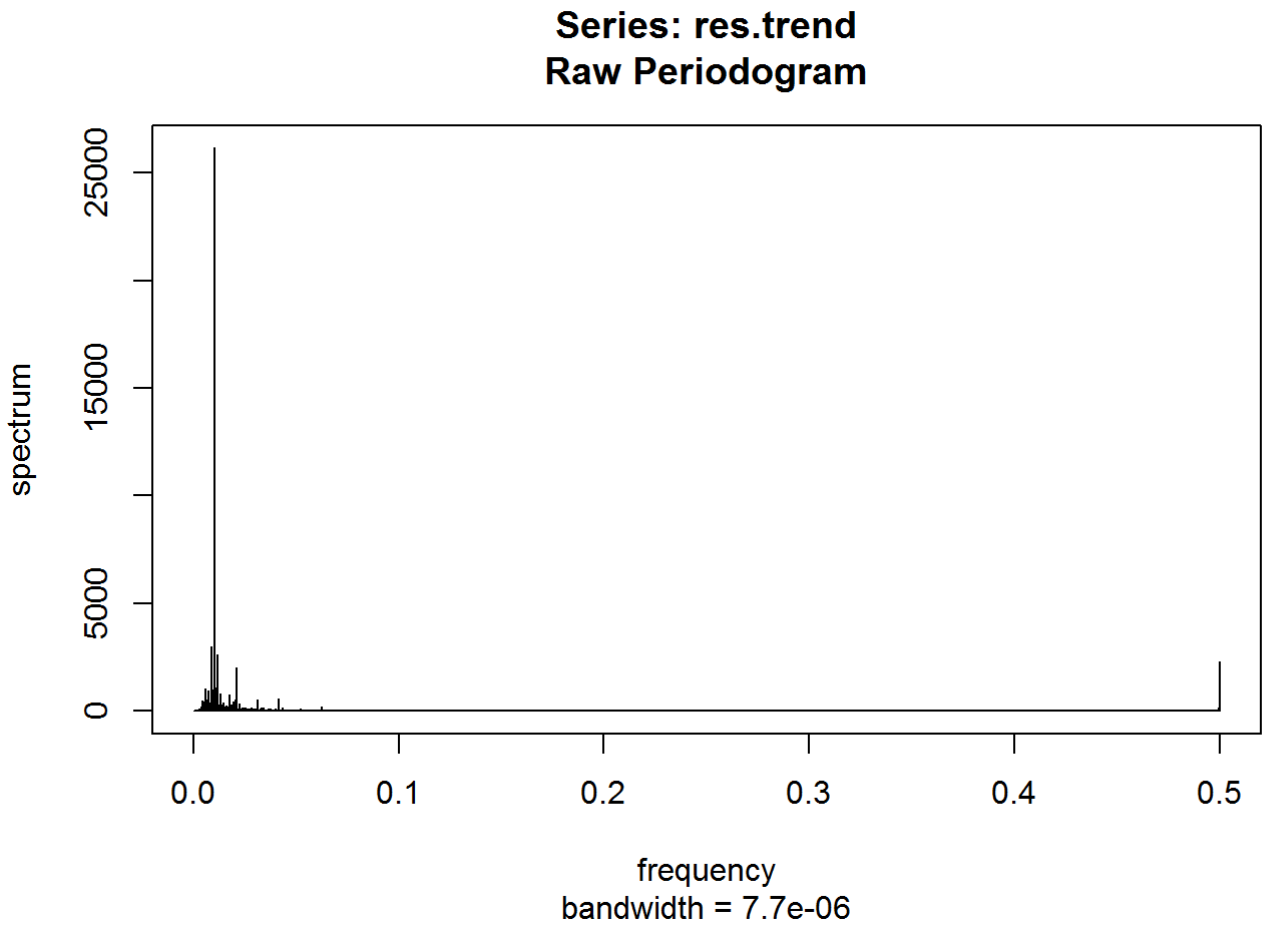

```
plot(wcor(s, groups = list(c(1),c(2,24), c(25,32), c(33,50))))
```



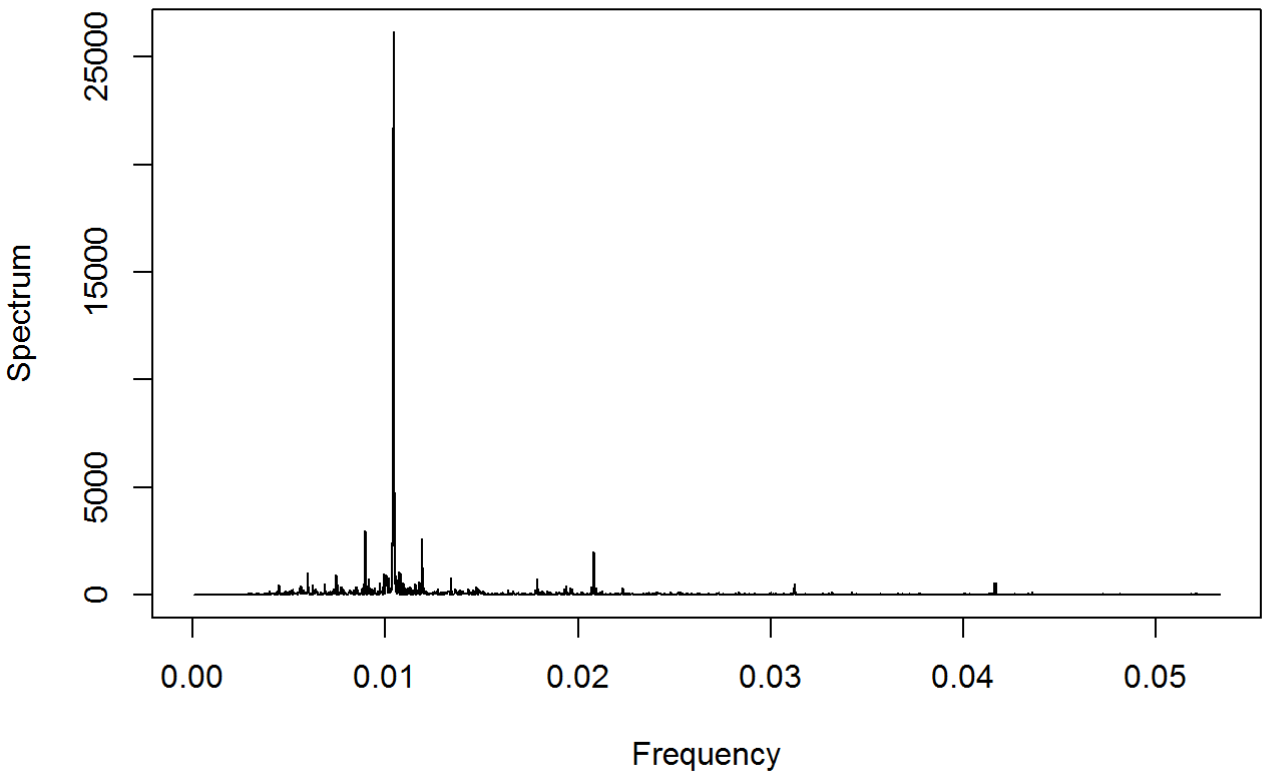

```
res1 <- reconstruct(s, groups = list(1))
trend <- res1$F1
plot(res1, add.residuals = FALSE, plot.type = "single", col = c("black", "red"), lwd = c(1, 2))
```



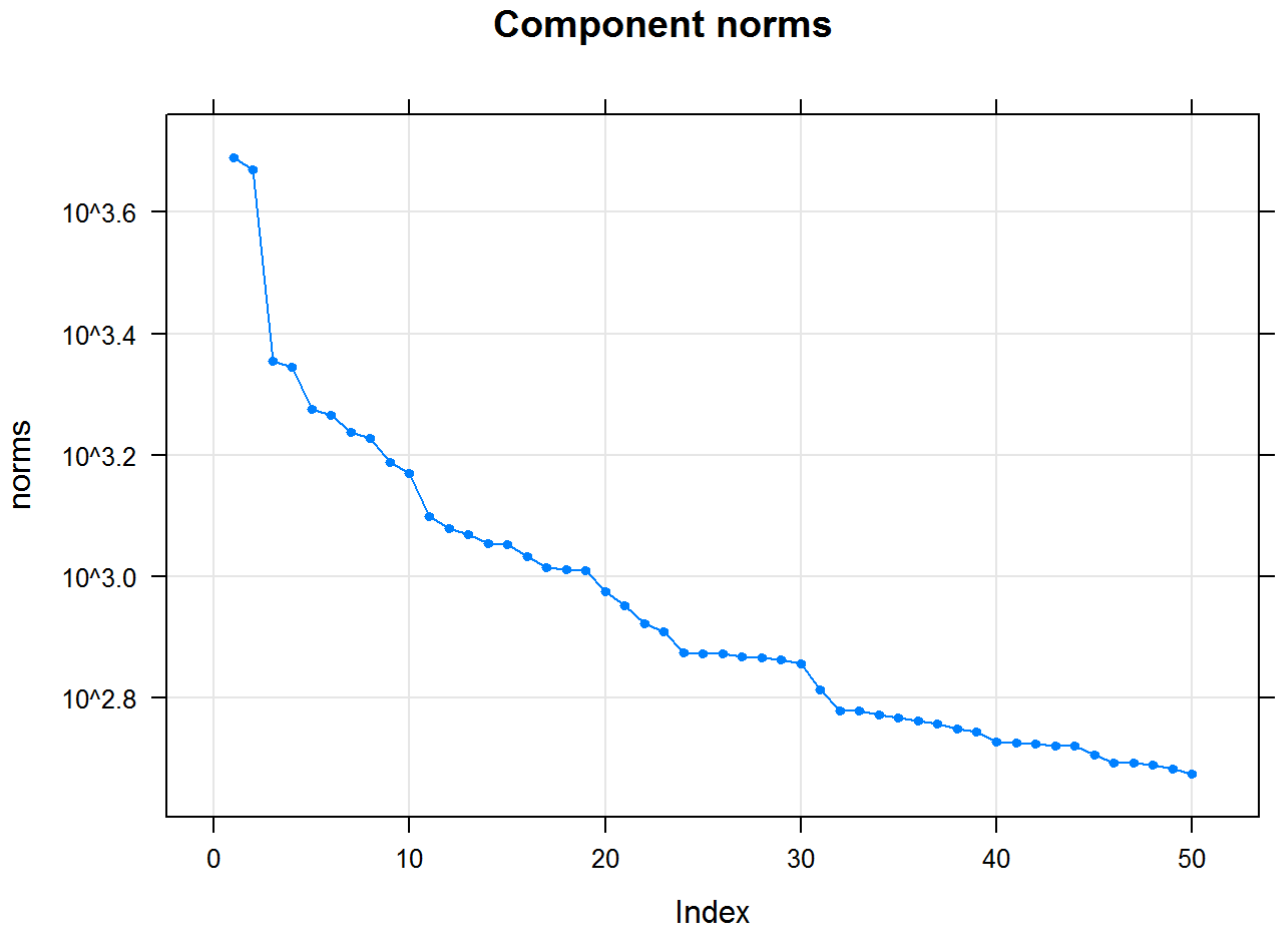

```
res.trend <- residuals(res1)

Spectogram <-spec.pgram(res.trend, detrend = FALSE, log = "no")
```



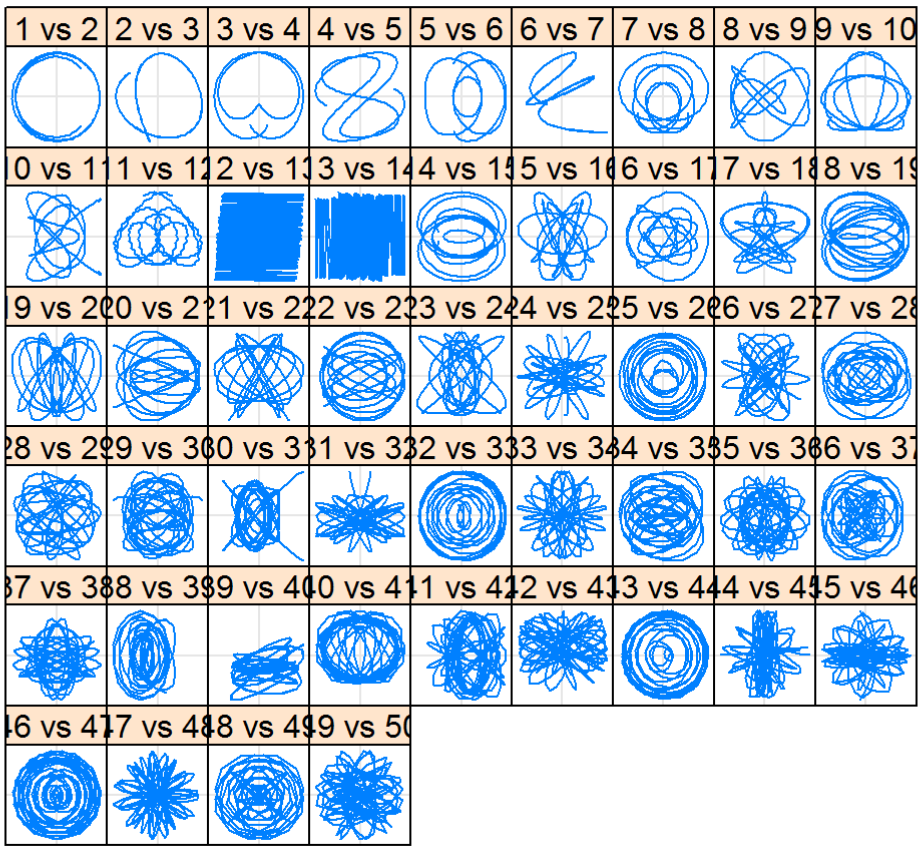

```
plot(Spectogram$freq[1:2000], Spectogram$spec[1:2000], type ="l", xlab="Frequency", ylab="Spectr
um")
```



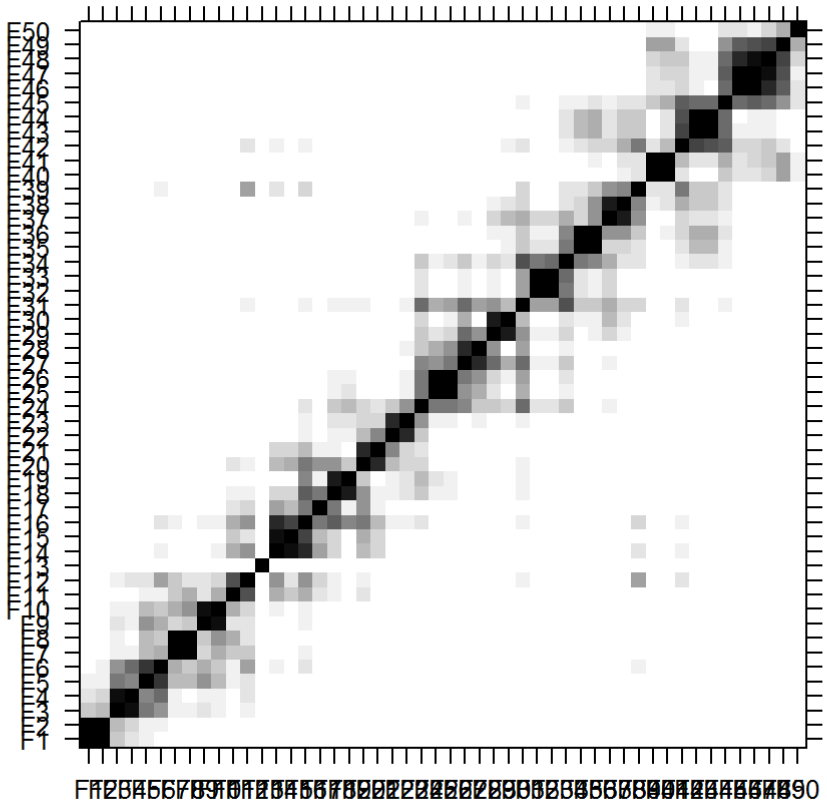

```
s2 <- ssa(res.trend, L=264)
plot(s2)
```



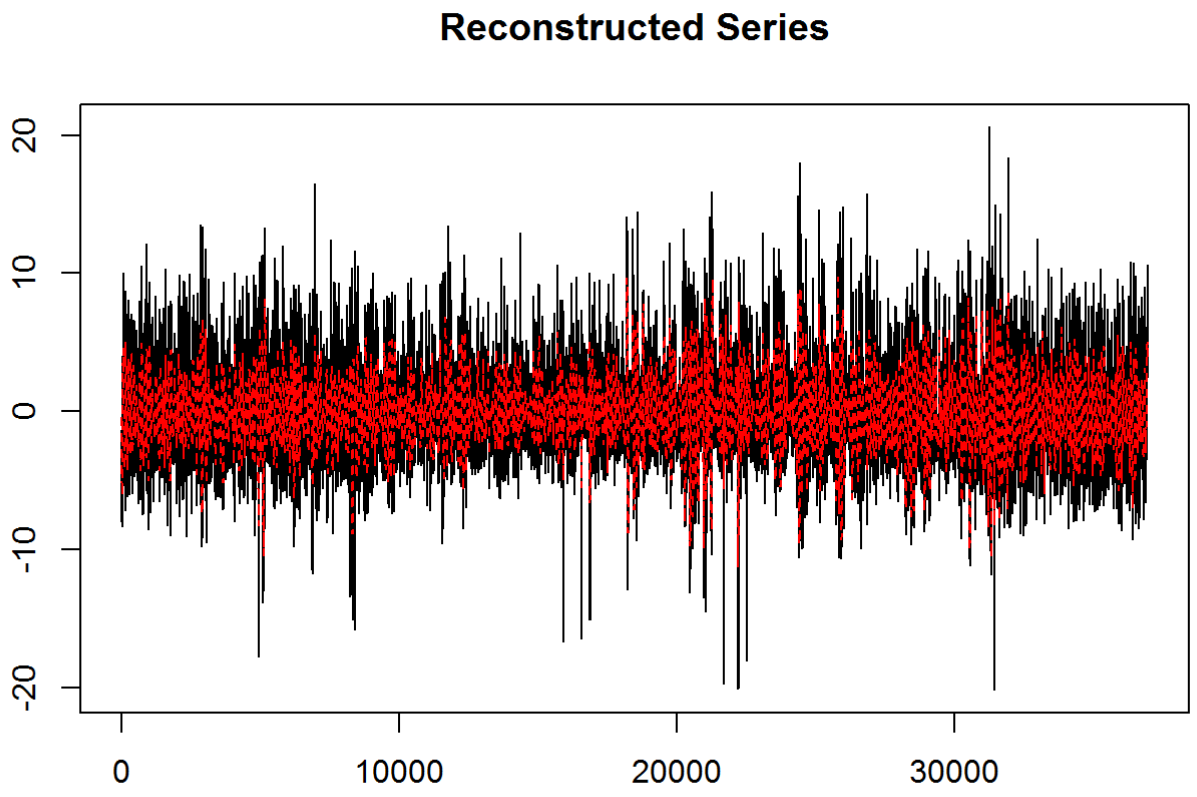

```
plot(s2, type = "paired", idx = 1:49, plot.contrib = FALSE)
```




```
# Calculate the w-correlation matrix using the first 30 components.
# Here the 'groups' argument as usual denotes the grouping used.
w <- wcor(s2, groups = as.list(1:50))
plot(w)
```




```
#parestimate(s2, list(1:12), method = "esprit-ls")$periods



res2 <- reconstruct(s2, groups=list(1:10))
seasonality <- res2$F1
res <- residuals(res2)
# Extracted seasonality
plot(res2, add.residuals = FALSE)
```




```
# Result of Sequential SSA
#plot(res2)
# Seasonally adjusted series
#plot(Data-seasonality, type="l")
```

Codes for data Cleaning and imputation

```
sdata <- fread("C:/Users/Inspiron/Desktop/EdifesPlots/3906.csv")

PID= unique(levels(as.factor(sdata$PointID)))
sdata=sdata[which(str_detect(sdata$PointID, PID[1])),]

sdata <- sdata[5000:length(sdata$FacilityID), ]
sdata$UnitOfMeasure <- as.factor(sdata$UnitOfMeasure)
#Change the time format to POSIXct
date <- strptime(sdata$MeterRecordedTimeStampUTC, format = '%Y-%m-%d %H:%M:%S')
sdata$MeterRecordedTimeStampUTC <- as.POSIXct(date, format = "%Y-%m-%dT%H:%M:%S")

c_time <-rev(sdata$MeterRecordedTimeStampUTC)
#finding the time interval
N <- difftime(c_time[1:(length(c_time)-1)], c_time[2:length(c_time)], units ="mins")
N <-round(N)
n <- sort(table(N),decreasing=TRUE)[1]
n <- as.integer(names(n))
rounddate <- round_date(sdata$MeterRecordedTimeStampUTC,  paste(n, "min"))
fdata <- cbind(sdata, rounddate)
fdata <-fdata[,MeterRecordedTimeStampUTC:=NULL]
fdata <- fdata[!duplicated(fdata$rounddate),]
fdata.z <- zoo(fdata, order.by = fdata$rounddate)
newtime <- seq(from =as.POSIXct(start(fdata.z)), to = as.POSIXct(end(fdata.z)), by = paste(n, "m
in"))
newtime.z <- zoo(x = newtime, order.by = newtime)


fulldata.z = merge(fdata.z, newtime.z, all = TRUE)
zooToDf <- function(z) {
  df <- as.data.frame(z)
  df$Date <- time(z) #create a Date column
  rownames(df) <- NULL #so row names not filled with dates
  df <- df[,c(ncol(df), 1:(ncol(df)-1))] #reorder columns so Date first
  return(df)
}

fulldata <- zooToDf(fulldata.z)

fulldata <- fulldata[c(122:37082),]
ggplot(fulldata)+geom_line(aes(x = Date, y = as.numeric(as.character(fulldata$PresentRawValu
e))))+ labs(title = paste("Facility:",fulldata$FacilityID,"Point:",fulldata$PointID))+ labs(y= p
aste("Consumption ",fulldata$UnitOfMeasure))
reduceddata <- fulldata[, !names(fulldata) %in% c("PresentRawValue","rounddate", "newtime.z")]

replace_na_with_last<-function(x,a=!is.na(x)){
  x[which(a)[c(1,1:sum(a))][cumsum(a)+1]]
}

for (i in 1:ncol(reduceddata)){
  reduceddata[,i] <- replace_na_with_last(reduceddata[,i])
}

fulldata <- cbind(reduceddata, fulldata$PresentRawValue)
```

```
fulldata <- fulldata[,c(2:12,13,1)]

fulldata$RawValueDifference<-as.numeric(as.character(fulldata$RawValueDifference))
daycount <-0
MSE.itr <-c()

for(n.month in 9:12){
for(n.day in seq(1,30,4)){
  daycount <-daycount+1
for(n.int in 1:23){

tsdata <- fulldata[,c(2,12,13,8)]
tsdata$`fulldata$PresentRawValue` <- as.integer(as.character(tsdata$`fulldata$PresentRawValue`))
names(tsdata)[2]<-paste("Energy")

Flag <- data.table(Flag = rep(0,length(tsdata$Energy)))
tsdata <- cbind(tsdata,Flag)
tsdata[which(is.na(tsdata$Energy)),]$Flag <-1

#Day and month

tsdata$Day<- weekdays(tsdata$Date)
tsdata$Month <-months(tsdata$Date)
##Outlier Detect
tsdata.noNA<-tsdata[which(!is.na(tsdata$Energy)),]
L <- getOutliersI(tsdata.noNA$Energy,rho=c(1,1),
                  FLim=c(0.01,0.99),distribution="normal")

tsdata.noNA[which(tsdata.noNA$Energy<L$limit[1]),]$Energy <-NA
#tsdata.noNA[which(tsdata.noNA$Energy>L$limit[2]),]$Energy <-NA
tsdata[which(!is.na(tsdata$Energy)),]$Energy <-tsdata.noNA$Energy

tsdata$Energy <- na.interpolation(tsdata$Energy)

meta <-subset(tsdata,Date>=as.POSIXct(paste("2015-",n.month,"-",n.day ," 00:00:00 EDT", sep
="")))
meta <-subset(meta,Date<=as.POSIXct(paste("2015-",n.month,"-",n.day," ", n.int,":00:00 EDT", sep
="")))

tsdata.Sep.day <-meta
tsdata[which(tsdata$Date>=as.POSIXct(paste("2015-",n.month,"-",n.day ," 00:00:00 EDT", sep=""))&
tsdata$Date<=as.POSIXct(paste("2015-",n.month,"-",n.day," ", n.int,":00:00 EDT", sep=""))),]$Ene
rgy <- NA
MM<-tsdata
Miss.Date <- tsdata[which(is.na(tsdata$Energy)),]
Miss.Energy <-tsdata.Sep.day$Energy
First.week <- tsdata[which(is.na(tsdata$Energy))+(96*7),]
Second.week <- tsdata[which(is.na(tsdata$Energy))+(2*96*7),]
Third.week <-tsdata[which(is.na(tsdata$Energy))-(96*7),]
Fourth.week <-tsdata[which(is.na(tsdata$Energy))-(2*96*7),]
Fifth.week <- tsdata[which(is.na(tsdata$Energy))+(3*96*7),]
Sixth.week <- tsdata[which(is.na(tsdata$Energy))+(4*96*7),]

Gather.week <- cbind(First.week$Energy,Second.week$Energy,
```

```
                    Third.week$Energy,Fourth.week$Energy)
Gather.week<-as.data.frame(Gather.week)
Gather.week$Mean <-rowSums(Gather.week)/4

Miss.see <-tsdata[which(is.na(tsdata$Energy)),]
Miss.see$Energy<-Miss.Energy
Miss.see$Mean<-Gather.week$Mean
Miss.see$Diff <-Miss.see$RawValueDifference
A <-diff(Miss.see$Mean)
B <-c()
B[1]<-Miss.see$RawValueDifference[1]
B[2:length(Miss.see$Energy)]=A
Miss.see$Diff <-B
Miss.see$sign<-0
Miss.see[which(Miss.see$Diff>0),]$sign<-1
if(Miss.see$Diff<0){
Miss.see[which(Miss.see$Diff<0),]$sign<-(-1)
}
Miss.see$actualsign<-0
Miss.see[which(Miss.see$RawValueDifference>0),]$actualsign<-1
if(Miss.see$RawValueDifference<0){
Miss.see[which(Miss.see$RawValueDifference<0),]$actualsign<-(-1)
}
Miss.see$Switch<-Miss.see$actualsign-Miss.see$sign
Miss.see$switchdetect <- 0
Miss.see[which(Miss.see$Switch==0),]$switchdetect<-1
Percent.w <-sum(Miss.see$switchdetect)/length(Miss.see$Energy)

Miss.see.w<-Miss.see
Rate.w<-sum(abs(Miss.see$sign))/(sum(abs(Miss.see$actualsign)))


A<-(Miss.see$Energy-Miss.see$Mean)^2
B<-sum(A)
MSE.w <- (1/length(Miss.see$Energy))*B

tsdata<-MM
First.week <- tsdata[which(is.na(tsdata$Energy))+(96*1),]
Second.week <- tsdata[which(is.na(tsdata$Energy))+(2*96*1),]
Third.week <-tsdata[which(is.na(tsdata$Energy))-(96*1),]
Fourth.week <-tsdata[which(is.na(tsdata$Energy))-(2*96*1),]
Fifth.week <- tsdata[which(is.na(tsdata$Energy))+(3*96*1),]
Sixth.week <- tsdata[which(is.na(tsdata$Energy))+(4*96*1),]

Gather.week <- cbind(First.week$Energy,Second.week$Energy,
                    Third.week$Energy,Fourth.week$Energy)
Gather.week<-as.data.frame(Gather.week)
Gather.week$Mean <-rowSums(Gather.week)/4

Miss.see <-tsdata[which(is.na(tsdata$Energy)),]
Miss.see$Energy<-Miss.Energy
Miss.see$Mean<-Gather.week$Mean
A<-(Miss.see$Energy-Miss.see$Mean)^2
B<-sum(A)
```

```
MSE <- (1/length(Miss.see$Energy))*B

A <-diff(Miss.see$Mean)
B[1]<-Miss.see$RawValueDifference[1]
B[2:length(Miss.see$Energy)]=A
Miss.see$Diff <-B
Miss.see$sign<-0
Miss.see[which(Miss.see$Diff>0),]$sign<-1
if(Miss.see$Diff<0){
Miss.see[which(Miss.see$Diff<0),]$sign<-(-1)
}
Miss.see$actualsign<-0
Miss.see[which(Miss.see$RawValueDifference>0),]$actualsign<-1
if(Miss.see$RawValueDifference<0){
Miss.see[which(Miss.see$RawValueDifference<0),]$actualsign<-(-1)
}
Miss.see$Switch<-Miss.see$actualsign-Miss.see$sign
Miss.see$switchdetect <- 0
Miss.see[which(Miss.see$Switch==0),]$switchdetect<-1
Percent <-sum(Miss.see$switchdetect)/length(Miss.see$Energy)

Rate<-sum(abs(Miss.see$sign))/(sum(abs(Miss.see$actualsign)))
###
tsdata<-MM
First.week <- tsdata[which(is.na(tsdata$Energy))+(96*1),]
Second.week <- tsdata[which(is.na(tsdata$Energy))+(2*96*1),]
Third.week <-tsdata[which(is.na(tsdata$Energy))-(96*1),]
Fourth.week <-tsdata[which(is.na(tsdata$Energy))-(2*96*1),]
Fifth.week <- tsdata[which(is.na(tsdata$Energy))+(3*96*1),]
Sixth.week <- tsdata[which(is.na(tsdata$Energy))+(4*96*1),]

Gather.week <- cbind(First.week$Energy,
                     Third.week$Energy)
Gather.week<-as.data.frame(Gather.week)
Gather.week$Mean <-rowSums(Gather.week)/2

Miss.see <-tsdata[which(is.na(tsdata$Energy)),]
Miss.see$Energy<-Miss.Energy
Miss.see$Mean<-Gather.week$Mean
A<-(Miss.see$Energy-Miss.see$Mean)^2
B<-sum(A)
MSE2 <- (1/length(Miss.see$Energy))*B

A <-diff(Miss.see$Mean)
B[1]<-Miss.see$RawValueDifference[1]
B[2:length(Miss.see$Energy)]=A
Miss.see$Diff <-B
Miss.see$sign<-0
if(Miss.see$Diff>0){
Miss.see[which(Miss.see$Diff>0),]$sign<-1
}
if(Miss.see$Diff<0){
Miss.see[which(Miss.see$Diff<0),]$sign<-(-1)
}
```

```
Miss.see$actualsign<-0
Miss.see[which(Miss.see$RawValueDifference>0),]$actualsign<-1
if(Miss.see$RawValueDifference<0){
Miss.see[which(Miss.see$RawValueDifference<0),]$actualsign<-(-1)
}
Miss.see$Switch<-Miss.see$actualsign-Miss.see$sign
Miss.see$switchdetect <- 0
Miss.see[which(Miss.see$Switch==0),]$switchdetect<-1
Percent2 <-sum(Miss.see$switchdetect)/length(Miss.see$Energy)
###
Rate2<-sum(abs(Miss.see$sign))/(sum(abs(Miss.see$actualsign)))



D.Time <- cbind(daycount,n.int,MSE.w,MSE,MSE2,
                Percent.w,Percent,Percent2,Rate.w,Rate,Rate2)

MSE.itr <-rbind(MSE.itr,D.Time)
}
}
}


MSE.itr <-as.data.frame(MSE.itr)
head(MSE.itr)
MSE.day<-aggregate(MSE.itr$MSE,by=list(category=MSE.itr$n.int),FUN=mean, na.rm=TRUE)
MSE.two.day<-aggregate(MSE.itr$MSE2,by=list(category=MSE.itr$n.int),FUN=mean, na.rm=TRUE)

MSE.week<-aggregate(MSE.itr$MSE.w,by=list(category=MSE.itr$n.int),FUN=mean, na.rm=TRUE)
Percent.day<-aggregate(MSE.itr$Percent,by=list(category=MSE.itr$n.int),FUN=mean, na.rm=TRUE)
Percent.two.day<-aggregate(MSE.itr$Percent2,by=list(category=MSE.itr$n.int),FUN=mean, na.rm=TRU
E)

Percent.week<-aggregate(MSE.itr$Percent.w,by=list(category=MSE.itr$n.int),FUN=mean, na.rm=TRUE)
Rate.day<-aggregate(MSE.itr$Rate,by=list(category=MSE.itr$n.int),FUN=mean, na.rm=TRUE)
Rate.two.day<-aggregate(MSE.itr$Rate2,by=list(category=MSE.itr$n.int),FUN=mean, na.rm=TRUE)
Rate.week<-aggregate(MSE.itr$Rate.w,by=list(category=MSE.itr$n.int),FUN=mean, na.rm=TRUE)

Results <-cbind(MSE.day[1]+1,MSE.day[2],MSE.two.day[2],MSE.week[2],Percent.day[2],Percent.two.da
y[2],Percent.week[2],Rate.day[2],Rate.two.day[2],Rate.week[2])
Results<- as.data.frame(Results)
colnames(Results)<-c("MissingInterval.Hour","MSE.4day.Mean","MSE.2day.Mean","MSE.4week.Mean",
                     "MatchPercent.4day.Mean","MatchPercent.2day.Mean","MatchPercent.4week.Mea
n",
                     "MatchRate.4day.Mean","MatchRate.2day.Mean","MatchRate.4week.Mean")

# RMSE

A <-mean(tsdata$Energy, na.rm =TRUE)
MSE.itr$RMSE <- sqrt(MSE.itr$MSE)*100/A
MSE.itr$RMSE2 <- sqrt(MSE.itr$MSE2)*100/A
MSE.itr$RMSE.w <- sqrt(MSE.itr$MSE.w)*100/A

pl<-ggplot(data=MSE.itr) + geom_line(aes(x=n.int,y=RMSE, colour="4day.Mean"), size=1)+
```

```
  geom_line(aes(x=n.int,y=RMSE2, colour="2day.Mean"), size=1) +
  geom_line(aes(x=n.int,y=RMSE.w,colour="4Week.Mean"), size=1)+facet_wrap(~daycount)
pl+labs(title="Imputation Accuracy")+xlab("Missing Interval (Hour)")+ylab(" RMSE/Mean (%)")+
  theme(axis.title=element_text(size = 15, face = "bold")) +
  theme(axis.title=element_text(size = 15, face = "bold")) +
  theme(axis.text=element_text(size = 12, face = "bold")) +
  theme(plot.title=element_text(size = 15, face = "bold")) +
  theme(strip.text = element_text(size = 12, face = "bold")) +
  theme(legend.text = element_text(size = 12, face = "bold"))
```