q1

```
library(readr)
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(xts)
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
library(ggplot2)
library(reshape2)
library(dygraphs)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:xts':
##
##     first, last
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(fpp)
```

```
## Loading required package: fma
```

```
## Loading required package: expsmooth

## Loading required package: lmtest

## Loading required package: tseries
```

```
library(fpp3)
```

```
## -- Attaching packages -------------------------------------------- fpp3 0.4.0 --

## v tibble      3.1.0      v tsibbledata 0.4.0
## v tidyr       1.1.3      v feasts      0.2.2
## v lubridate   1.7.10     v fable       0.3.1
## v tsibble     1.1.1

## -- Conflicts ------------------------------------------------- fpp3_conflicts --
## x lubridate::date()      masks base::date()
## x dplyr::filter()        masks stats::filter()
## x dplyr::first()         masks xts::first()
## x fabletools::forecast() masks forecast::forecast()
## x tsibble::index()       masks zoo::index()
## x tsibble::intersect()   masks base::intersect()
## x tsibble::interval()    masks lubridate::interval()
## x dplyr::lag()           masks stats::lag()
## x dplyr::last()          masks xts::last()
## x tsibble::setdiff()     masks base::setdiff()
## x tsibble::union()       masks base::union()

##
## Attaching package: 'fpp3'

## The following object is masked from 'package:fpp':
##
##     insurance
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v purrr   0.3.4     v forcats 0.5.1
## v stringr 1.4.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x lubridate::as.difftime() masks base::as.difftime()
## x gridExtra::combine()     masks dplyr::combine()
## x lubridate::date()        masks base::date()
## x dplyr::filter()          masks stats::filter()
## x dplyr::first()           masks xts::first()
## x tsibble::intersect()     masks lubridate::intersect(), base::intersect()
## x tsibble::interval()      masks lubridate::interval()
## x dplyr::lag()             masks stats::lag()
## x dplyr::last()            masks xts::last()
## x tsibble::setdiff()       masks lubridate::setdiff(), base::setdiff()
## x tsibble::union()         masks lubridate::union(), base::union()
```

```
library(timetk)
library(tseries)
# Setup for the plotly charts (# FALSE returns ggplots)
interactive <- TRUE
```

#Step 1 read the csv datasets files

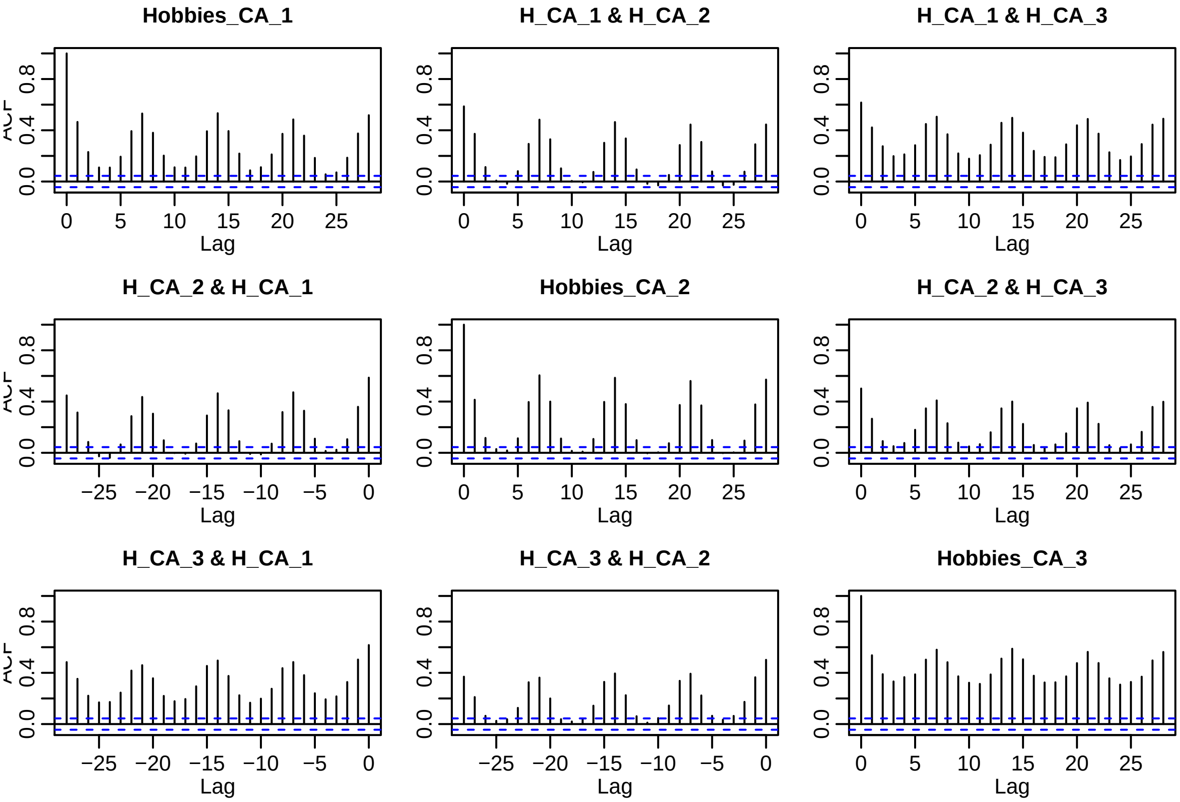
```
data = read.csv("/Users/yingding/Desktop/Forcasting M2/Project and data-20220107/github code/Projectdat
```

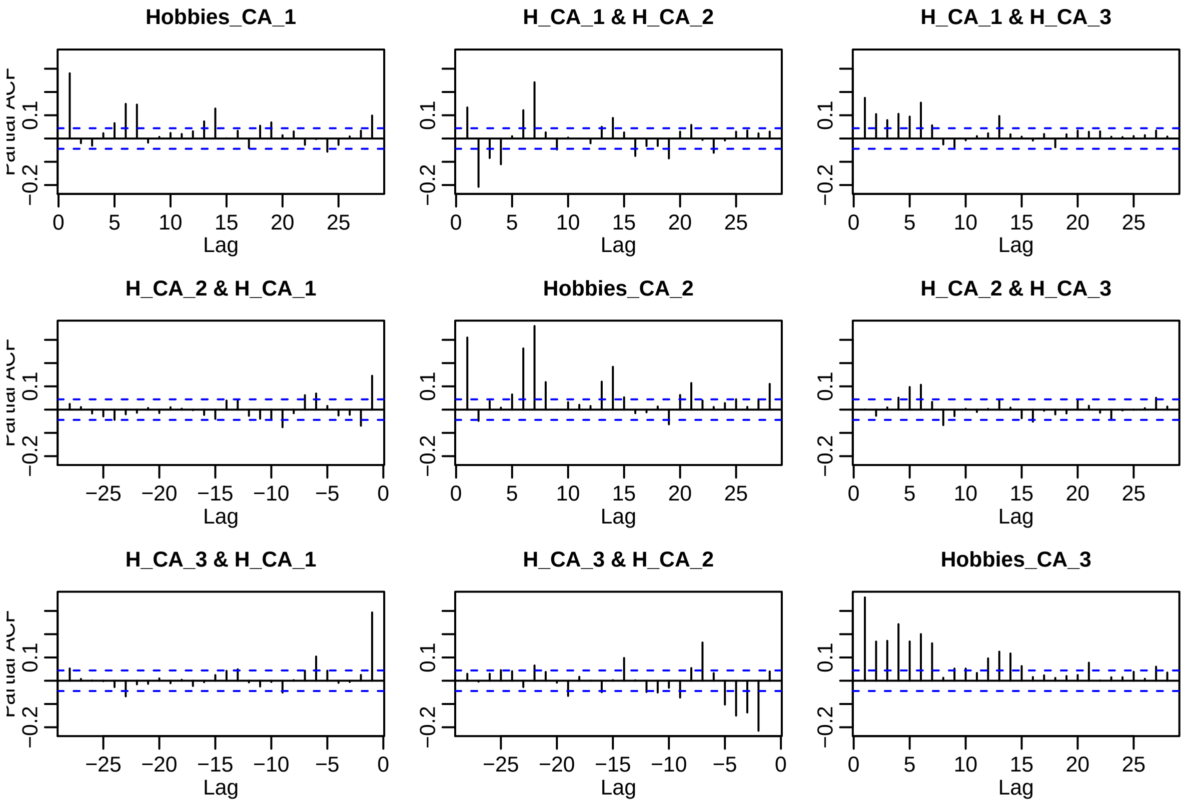
#transform to time series data

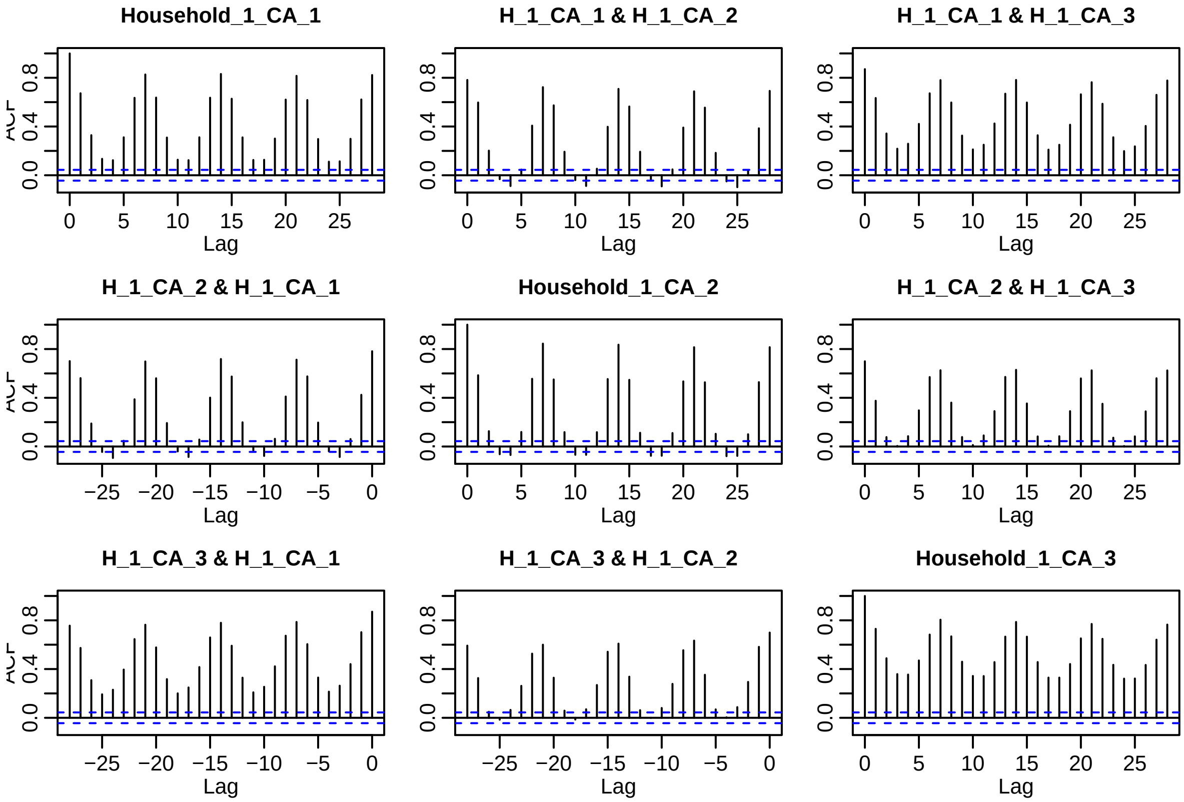
```
data$date <- as.Date(data$date)
tsdata <- xts(data[,2:19], data$date )
```

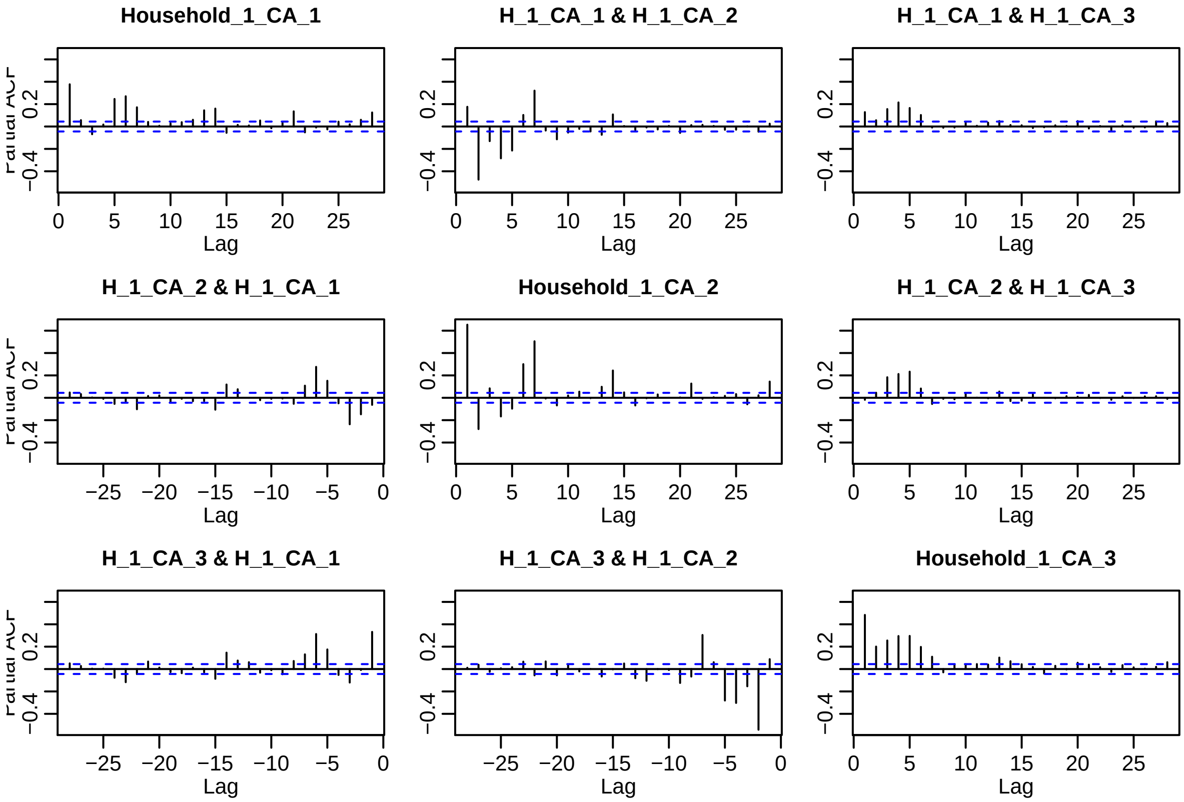
#Step 2 plot the data with acf and pacf

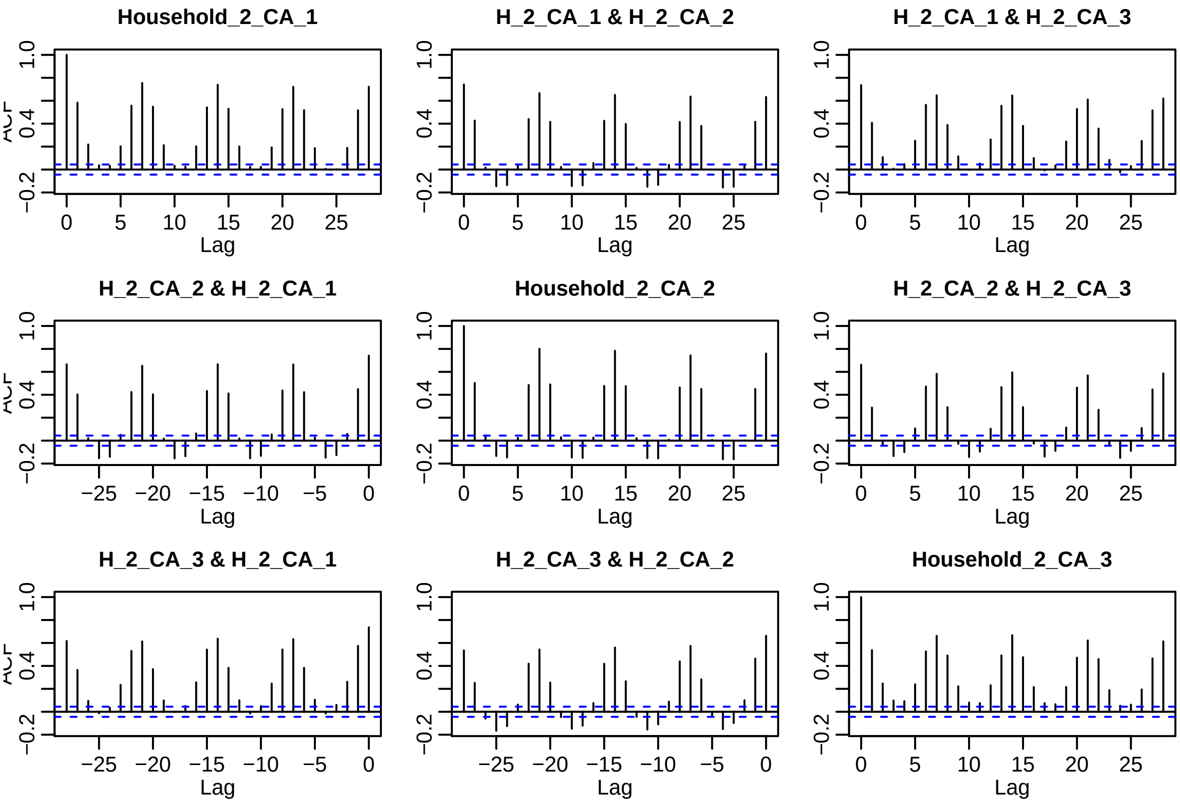
```
for (i in 1:6){
 pardata <- tsdata[,c(i,i+6,i+12)]
 p <- dygraph(pardata)
 acf(pardata)
 pacf(pardata)
 print(p)
 }
```

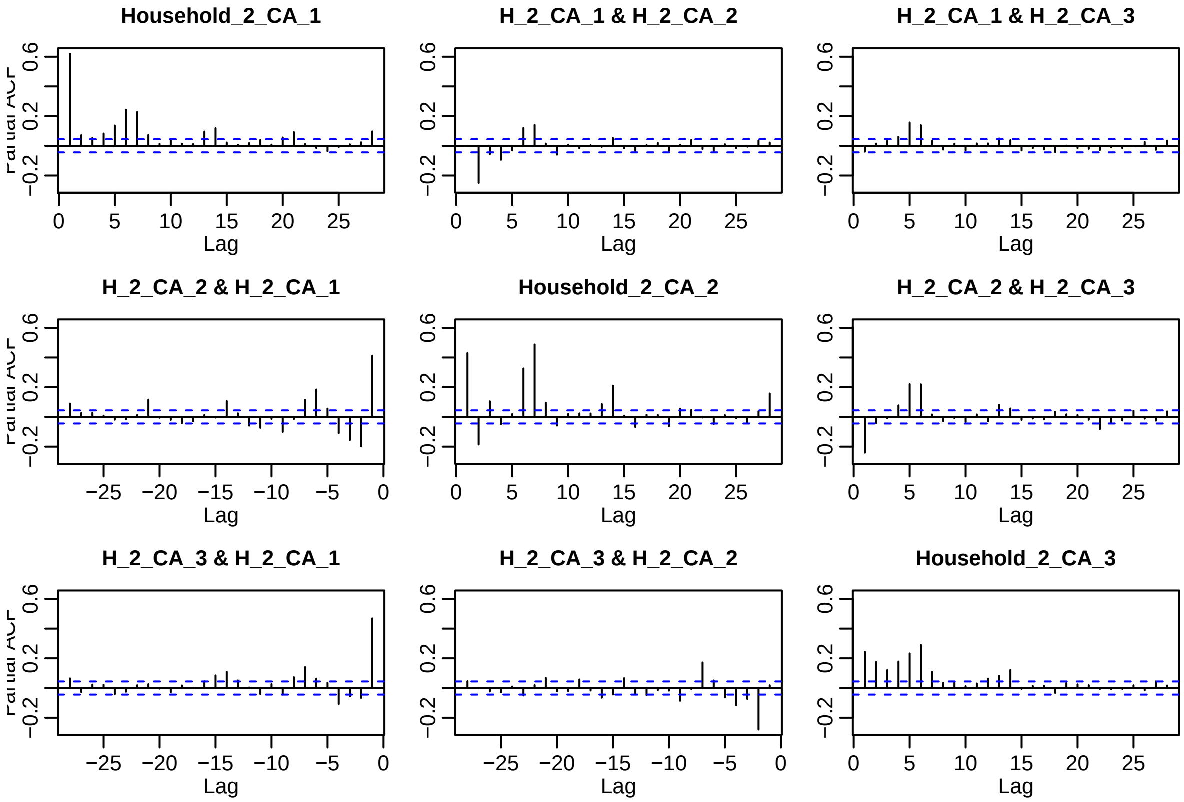

Hobbies_CA_1
H_CA_1 & H_CA_2
H_CA_1 & H_CA_3
H_CA_2 & H_CA_1
Hobbies_CA_2
H_CA_2 & H_CA_3
H_CA_3 & H_CA_1
H_CA_3 & H_CA_2
Hobbies_CA_3
ACF
Lag
0.0
0.4
0.8
0
5
10
15
20
25
−25
−20
−15
−10
−5

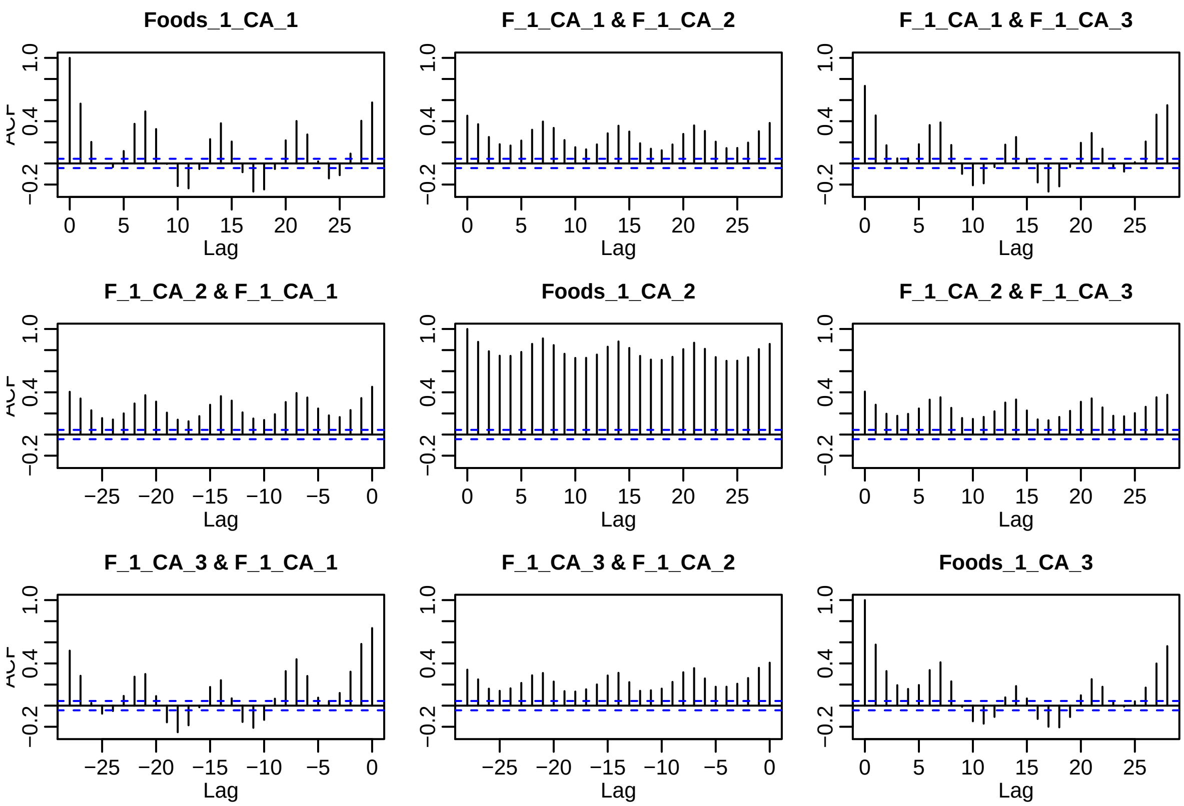

Hobbies_CA_1
H_CA_1 & H_CA_2
H_CA_1 & H_CA_3
H_CA_2 & H_CA_1
Hobbies_CA_2
H_CA_2 & H_CA_3
H_CA_3 & H_CA_1
H_CA_3 & H_CA_2
Hobbies_CA_3
Partial ACF
Lag

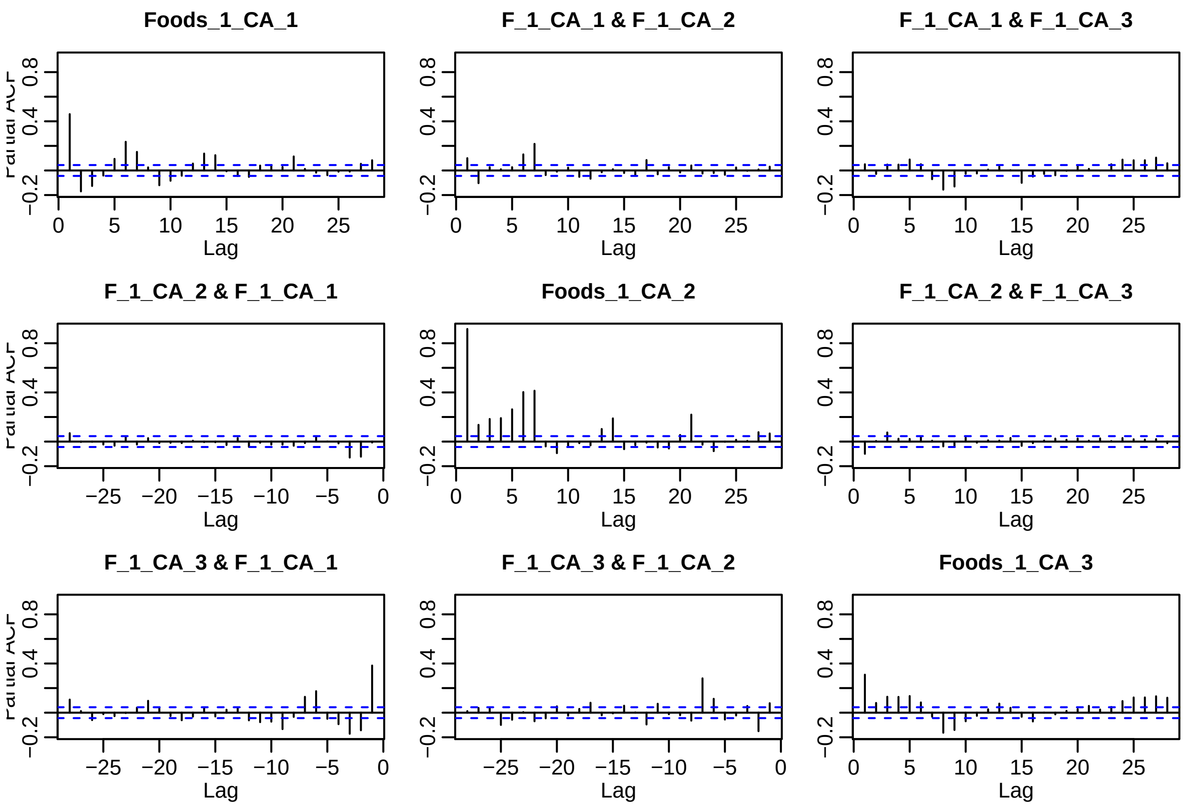

Household_1_CA_1
H_1_CA_1 & H_1_CA_2
H_1_CA_1 & H_1_CA_3
H_1_CA_2 & H_1_CA_1
Household_1_CA_2
H_1_CA_2 & H_1_CA_3
H_1_CA_3 & H_1_CA_1
H_1_CA_3 & H_1_CA_2
Household_1_CA_3
ACF
Lag

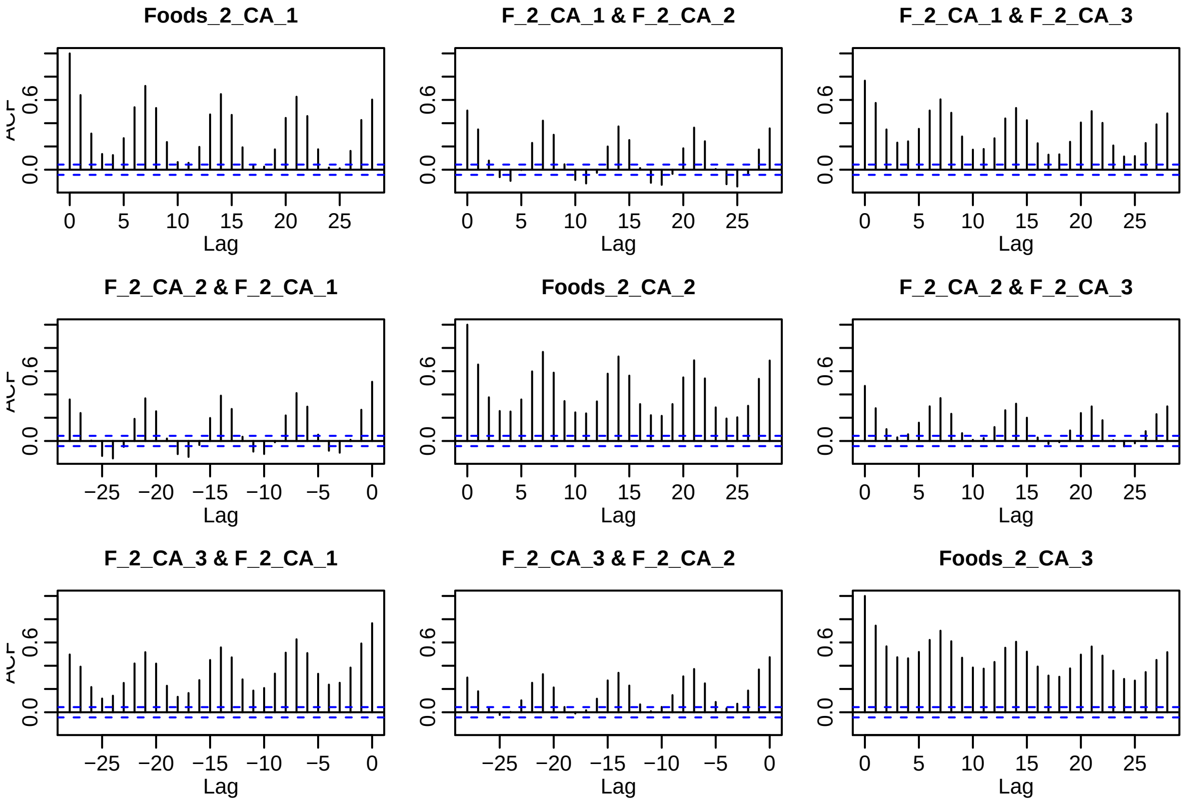

Household_1_CA_1
H_1_CA_1 & H_1_CA_2
H_1_CA_1 & H_1_CA_3
H_1_CA_2 & H_1_CA_1
Household_1_CA_2
H_1_CA_2 & H_1_CA_3
H_1_CA_3 & H_1_CA_1
H_1_CA_3 & H_1_CA_2
Household_1_CA_3
Partial ACF
Lag
0.2
−0.4
0
5
10
15
20
25
−25
−20
−15
−10
−5

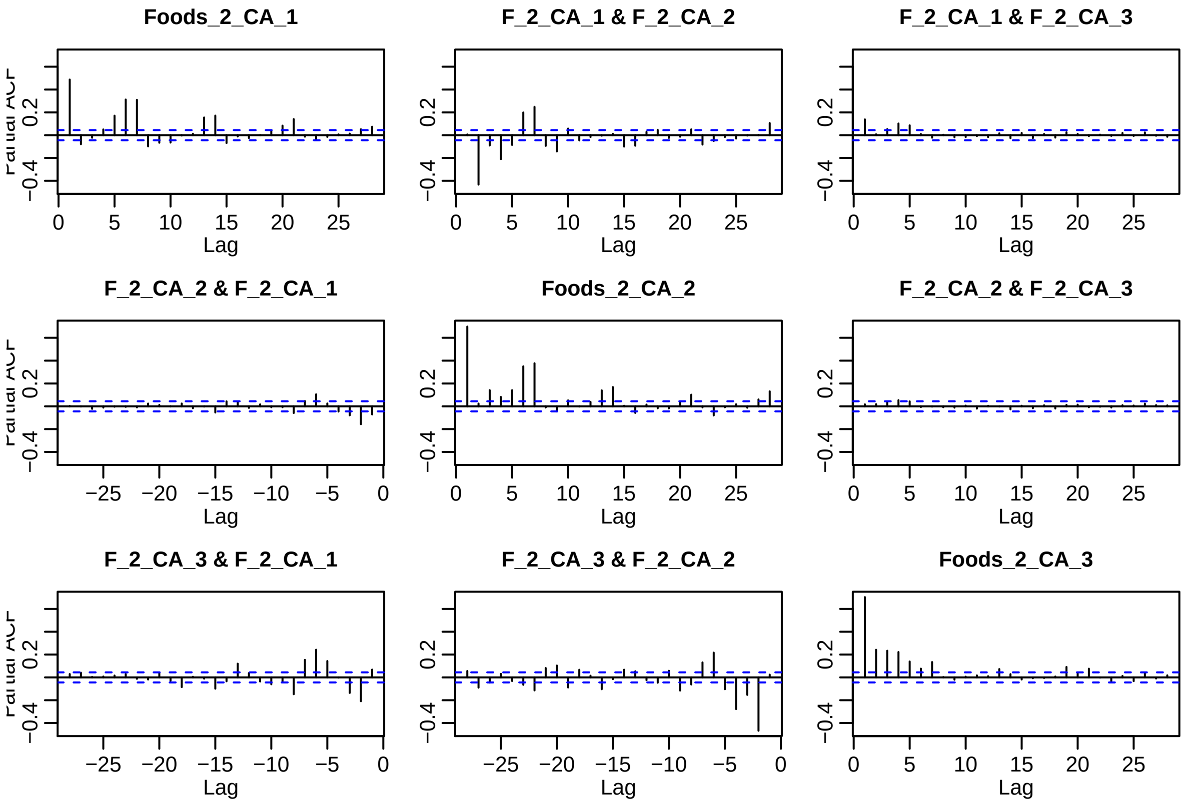

Household_2_CA_1
H_2_CA_1 & H_2_CA_2
H_2_CA_1 & H_2_CA_3
H_2_CA_2 & H_2_CA_1
Household_2_CA_2
H_2_CA_2 & H_2_CA_3
H_2_CA_3 & H_2_CA_1
H_2_CA_3 & H_2_CA_2
Household_2_CA_3
ACF
Lag
−0.2
0.4
1.0
0
5
10
15
20
25
−25
−20
−15
−10
−5

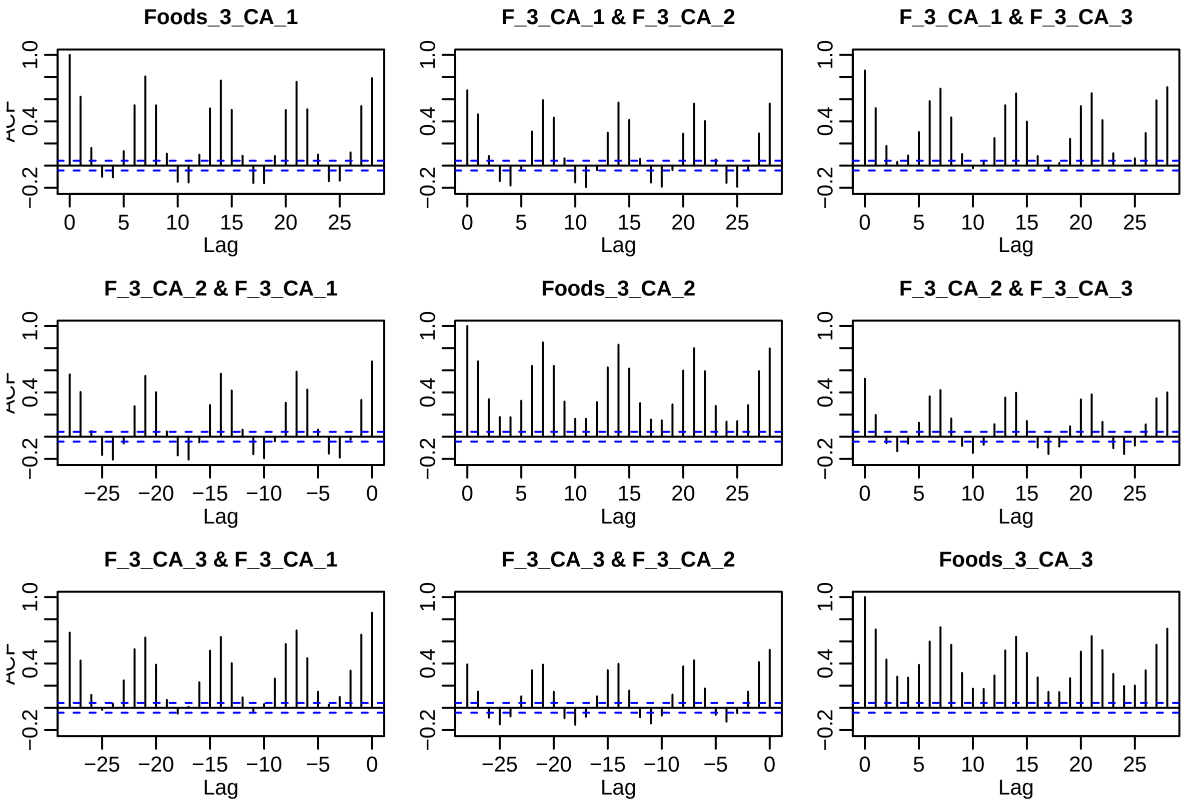

Household_2_CA_1
H_2_CA_1 & H_2_CA_2
H_2_CA_1 & H_2_CA_3
H_2_CA_2 & H_2_CA_1
Household_2_CA_2
H_2_CA_2 & H_2_CA_3
H_2_CA_3 & H_2_CA_1
H_2_CA_3 & H_2_CA_2
Household_2_CA_3
Partial ACF
Lag

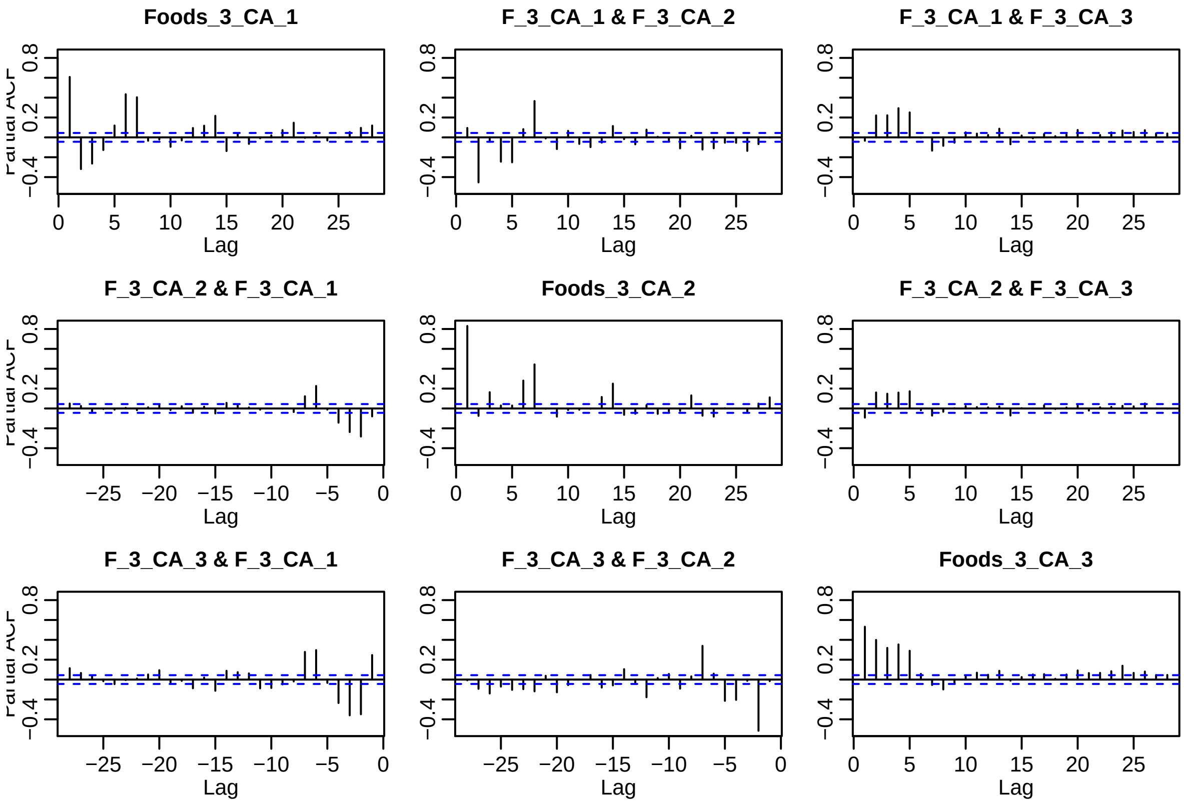

Foods_1_CA_1
F_1_CA_1 & F_1_CA_2
F_1_CA_1 & F_1_CA_3
F_1_CA_2 & F_1_CA_1
Foods_1_CA_2
F_1_CA_2 & F_1_CA_3
F_1_CA_3 & F_1_CA_1
F_1_CA_3 & F_1_CA_2
Foods_1_CA_3
ACF
Lag


Foods_1_CA_1
F_1_CA_1 & F_1_CA_2
F_1_CA_1 & F_1_CA_3
F_1_CA_2 & F_1_CA_1
Foods_1_CA_2
F_1_CA_2 & F_1_CA_3
F_1_CA_3 & F_1_CA_1
F_1_CA_3 & F_1_CA_2
Foods_1_CA_3
Partial ACF
Lag


Foods_2_CA_1
F_2_CA_1 & F_2_CA_2
F_2_CA_1 & F_2_CA_3
F_2_CA_2 & F_2_CA_1
Foods_2_CA_2
F_2_CA_2 & F_2_CA_3
F_2_CA_3 & F_2_CA_1
F_2_CA_3 & F_2_CA_2
Foods_2_CA_3
ACF
0.0
0.6
Lag
0
5
10
15
20
25
−25
−20
−15
−10
−5

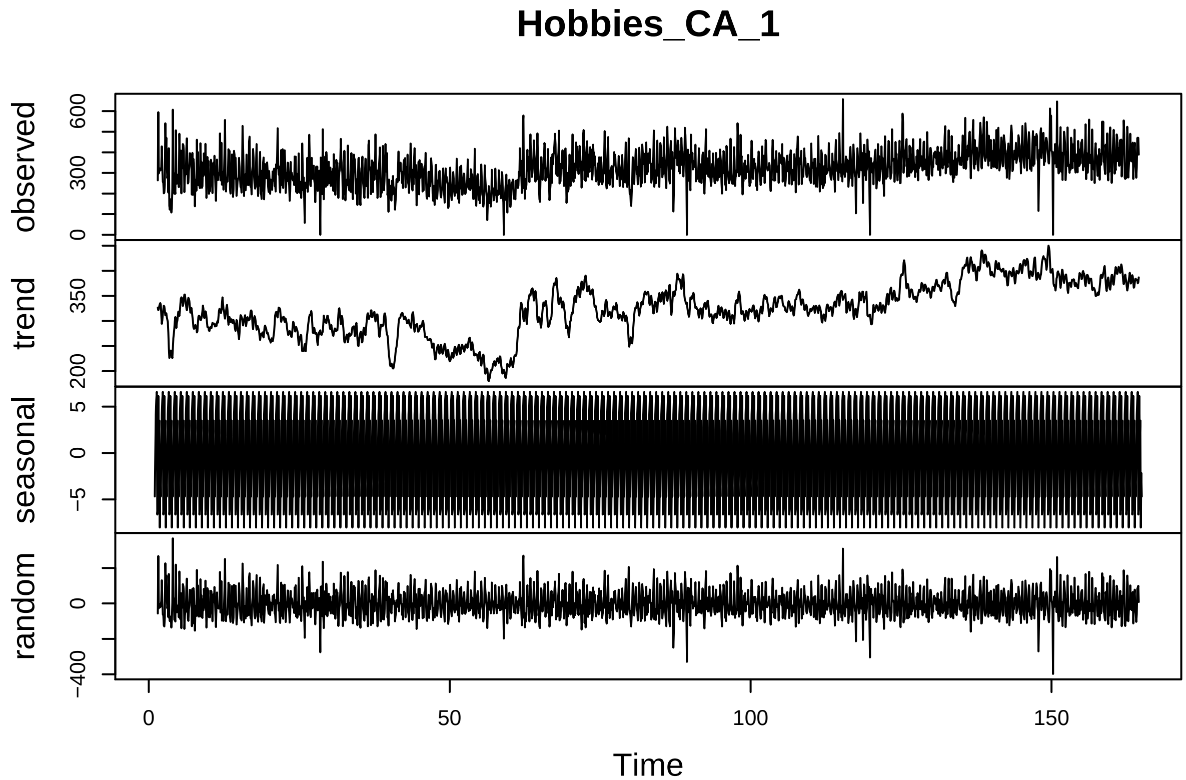

Foods_2_CA_1
F_2_CA_1 & F_2_CA_2
F_2_CA_1 & F_2_CA_3
F_2_CA_2 & F_2_CA_1
Foods_2_CA_2
F_2_CA_2 & F_2_CA_3
F_2_CA_3 & F_2_CA_1
F_2_CA_3 & F_2_CA_2
Foods_2_CA_3
Partial ACF
Lag

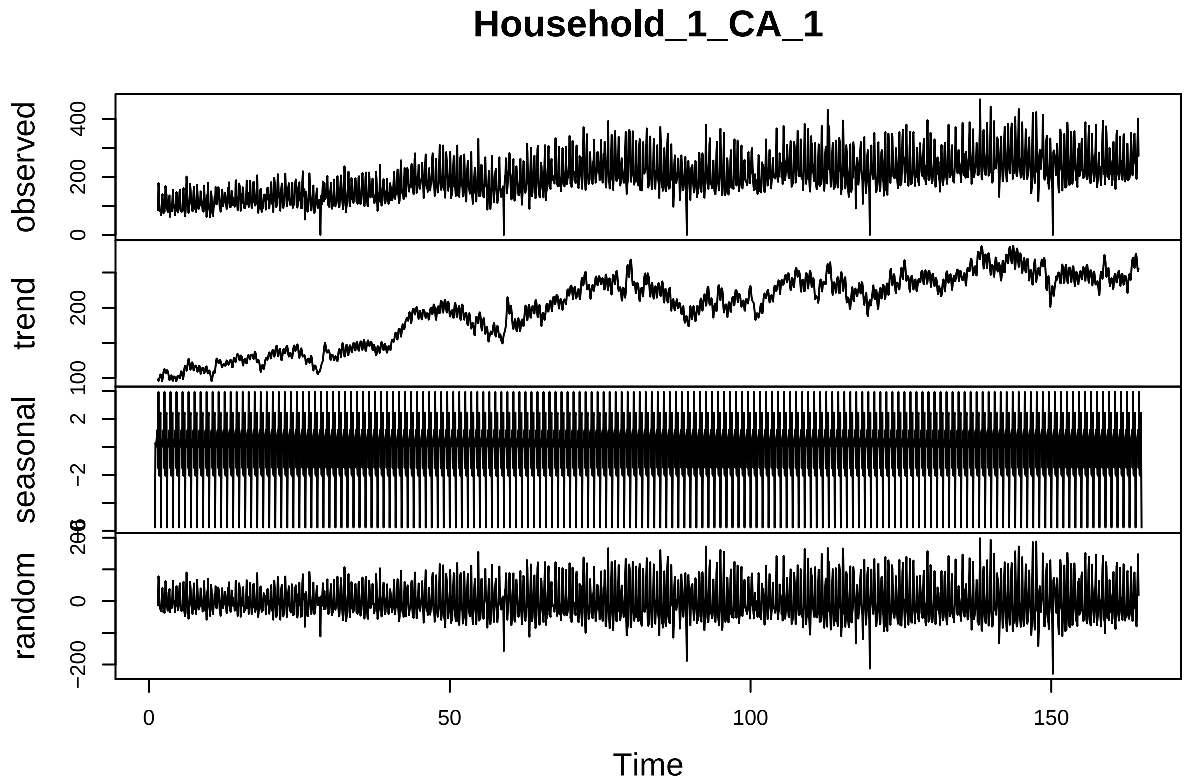

Foods_3_CA_1
F_3_CA_1 & F_3_CA_2
F_3_CA_1 & F_3_CA_3
F_3_CA_2 & F_3_CA_1
Foods_3_CA_2
F_3_CA_2 & F_3_CA_3
F_3_CA_3 & F_3_CA_1
F_3_CA_3 & F_3_CA_2
Foods_3_CA_3
ACF
Lag
−0.2
0.4
1.0
0
5
10
15
20
25
−25
−20
−15
−10
−5



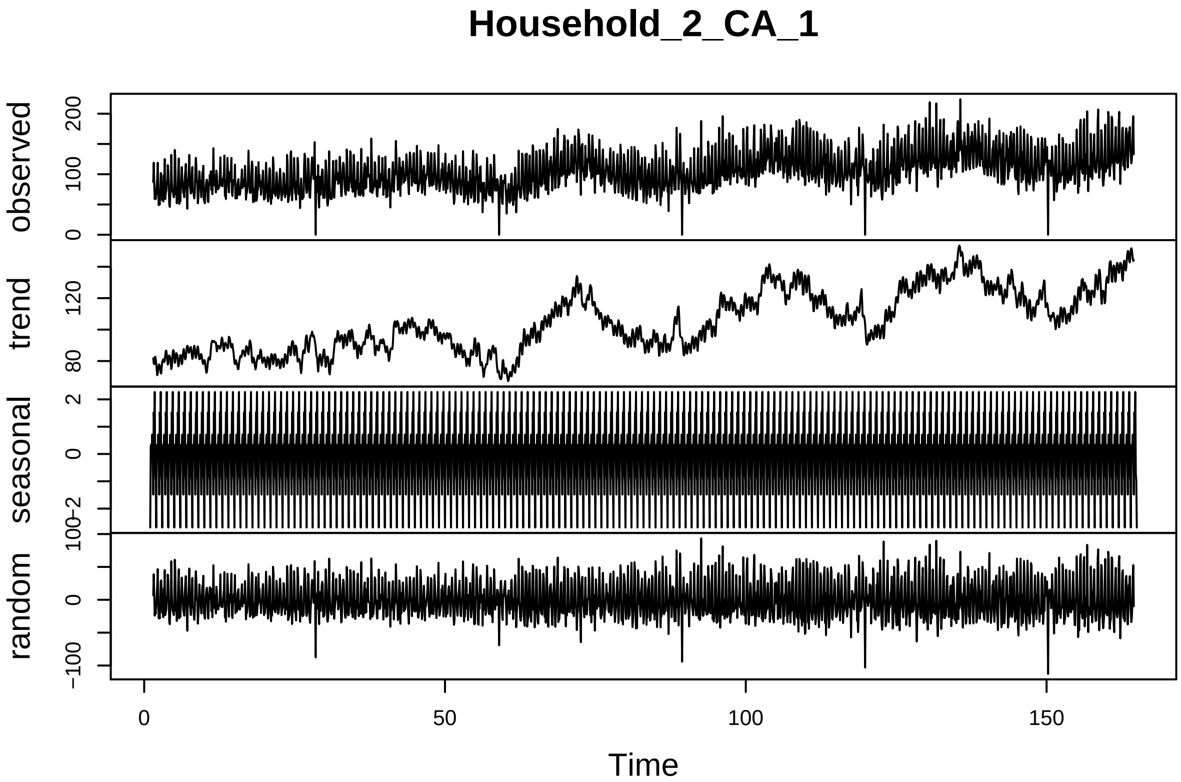
#Step 3 check seasonality

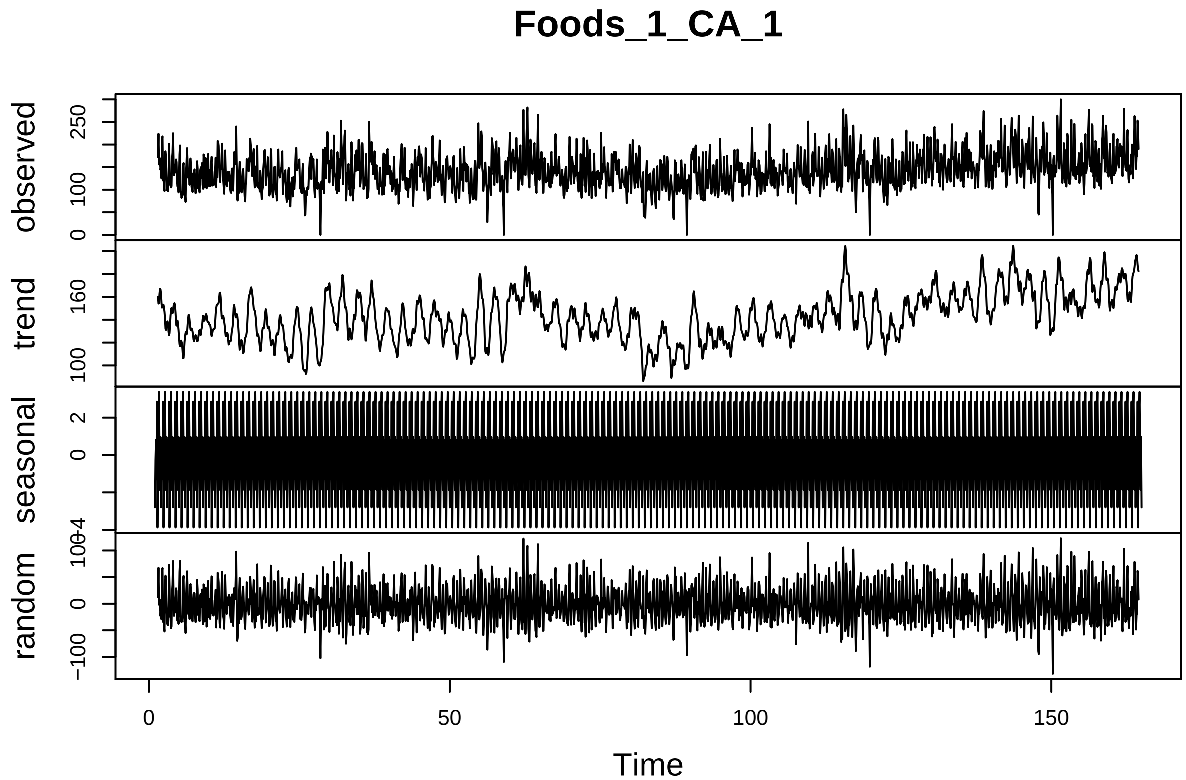
```
#function to plot hte decomposition
decomp.plot <- function(x, main = NULL, ...)
{
    if(is.null(main))
    main <- paste("Decomposition of", x$type, "time series")
    plot(cbind(observed = x$random + if (x$type == "additive")
        x$trend + x$seasonal
    else x$trend * x$seasonal, trend = x$trend, seasonal = x$seasonal,
        random = x$random), main = main, ...)
}

for (i in 1:18)
{
 monthdata <- ts(tsdata, frequency=12)
 seasondata<- decompose(monthdata[,i])
 decomp.plot(seasondata, main = colnames(monthdata)[i])
}
```

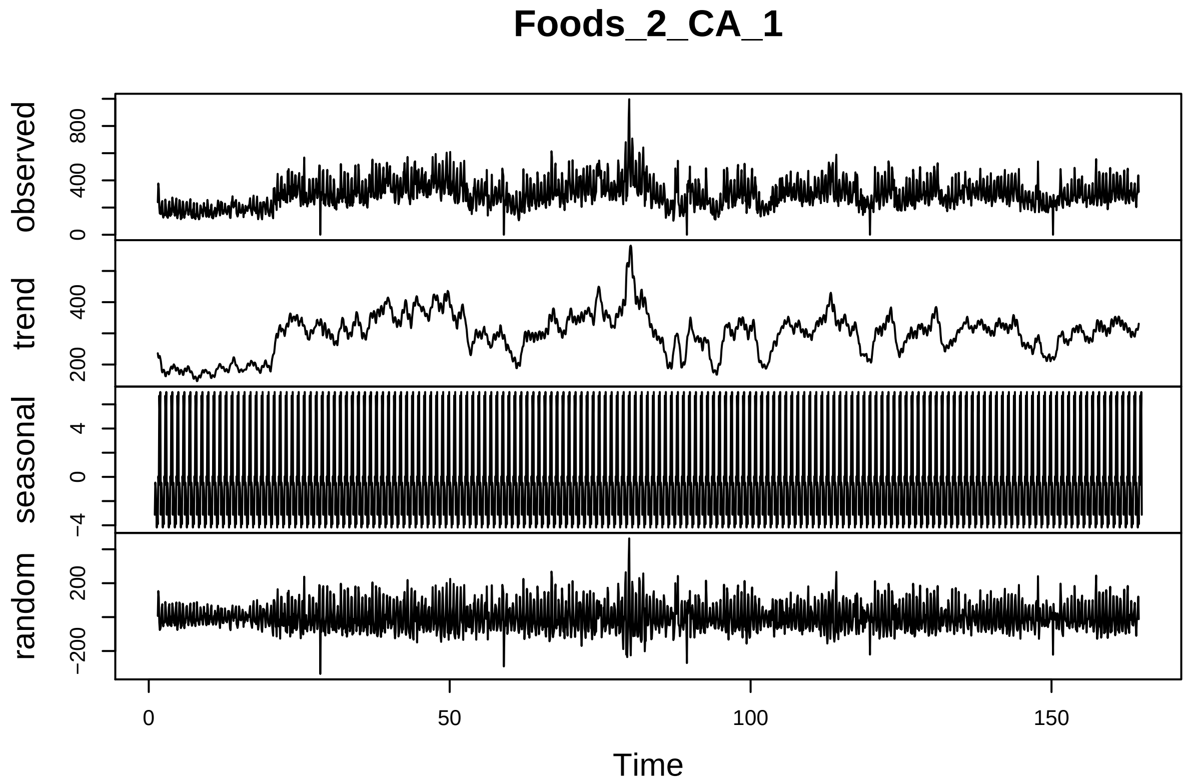

Hobbies_CA_1
observed
trend
seasonal
random
Time
0
300
600
200
350
−5
0
5
−400
0
50
100
150



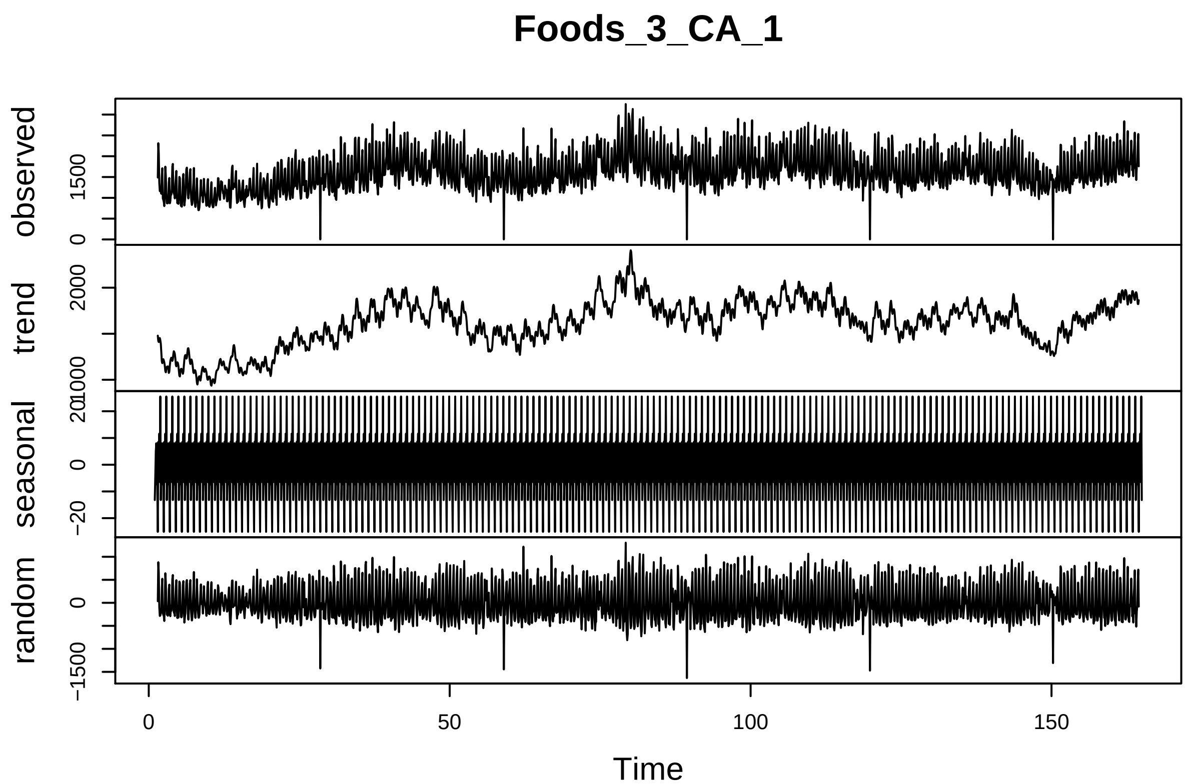
Household_1_CA_1
observed
trend
seasonal
random
0
200
400
100
200
2
−2
−6
−200
0
200
0
50
100
150
Time

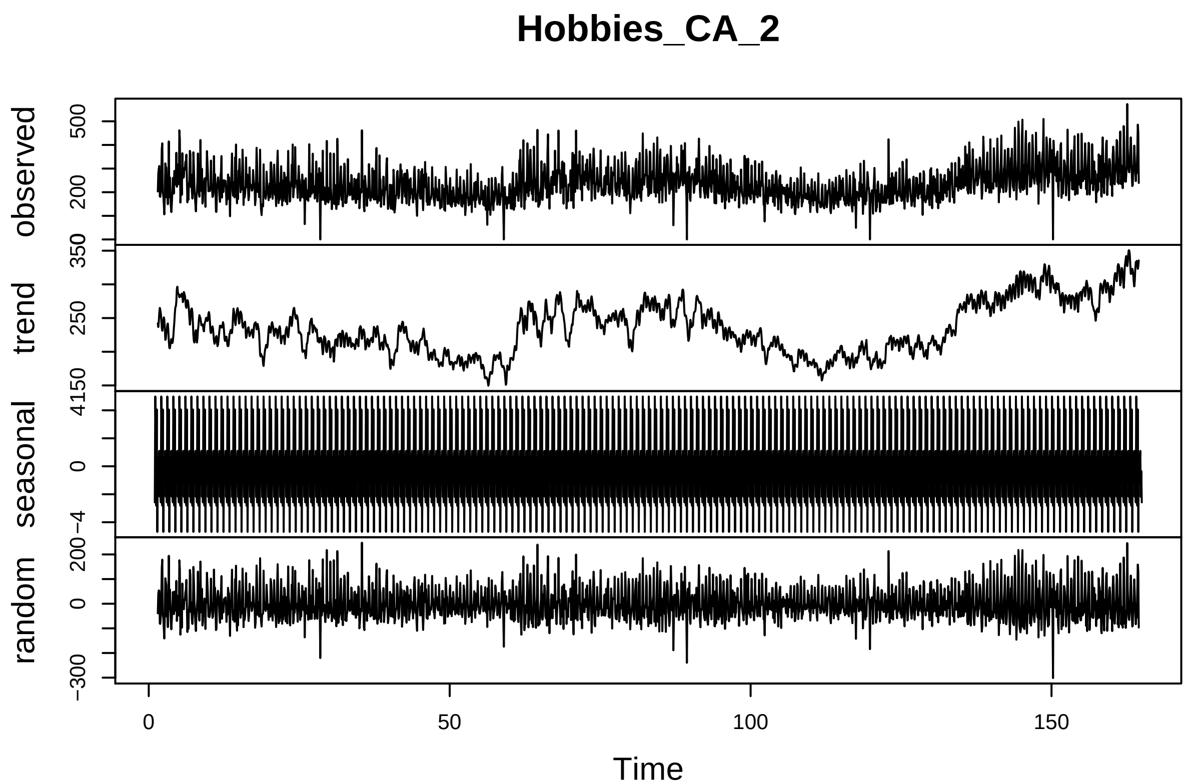

Household_2_CA_1
observed
trend
seasonal
random
0
100
200
80
120
2
0
−2
−100
0
100
0
50
100
150
Time

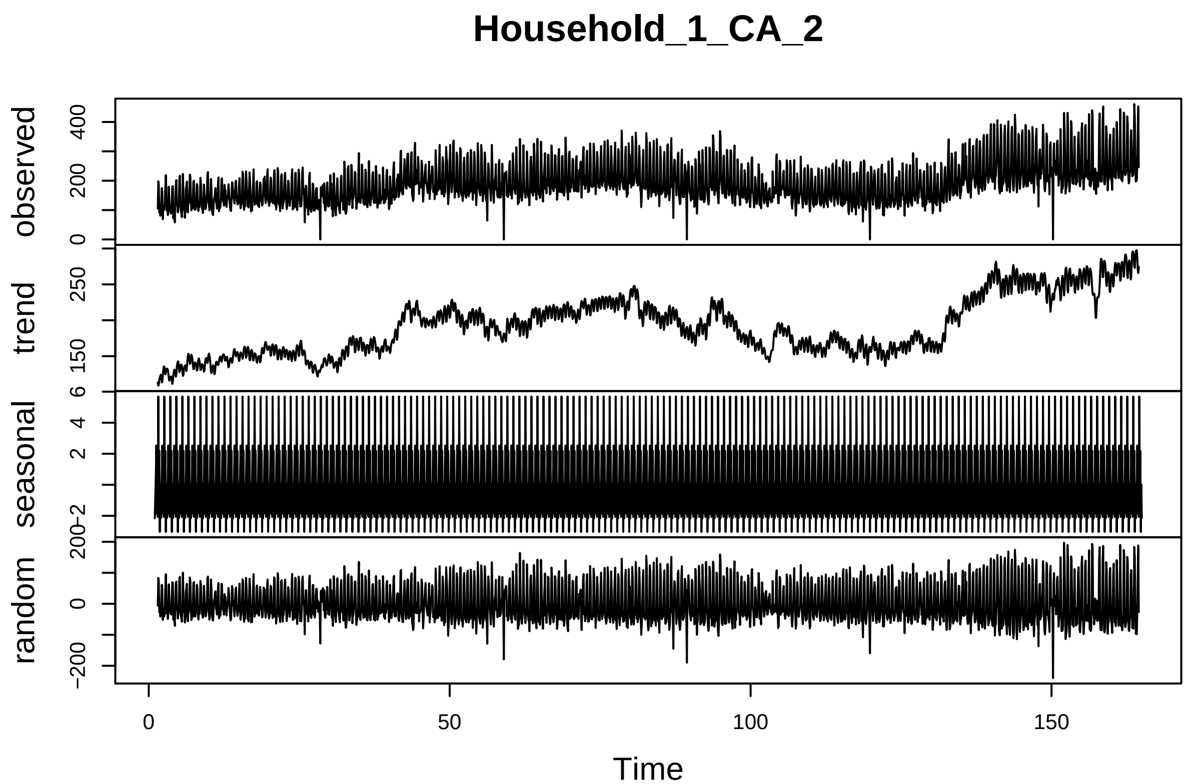

Foods_1_CA_1
observed
trend
seasonal
random
0
100
250
100
160
2
0
-4
100
0
-100
0
50
100
150
Time



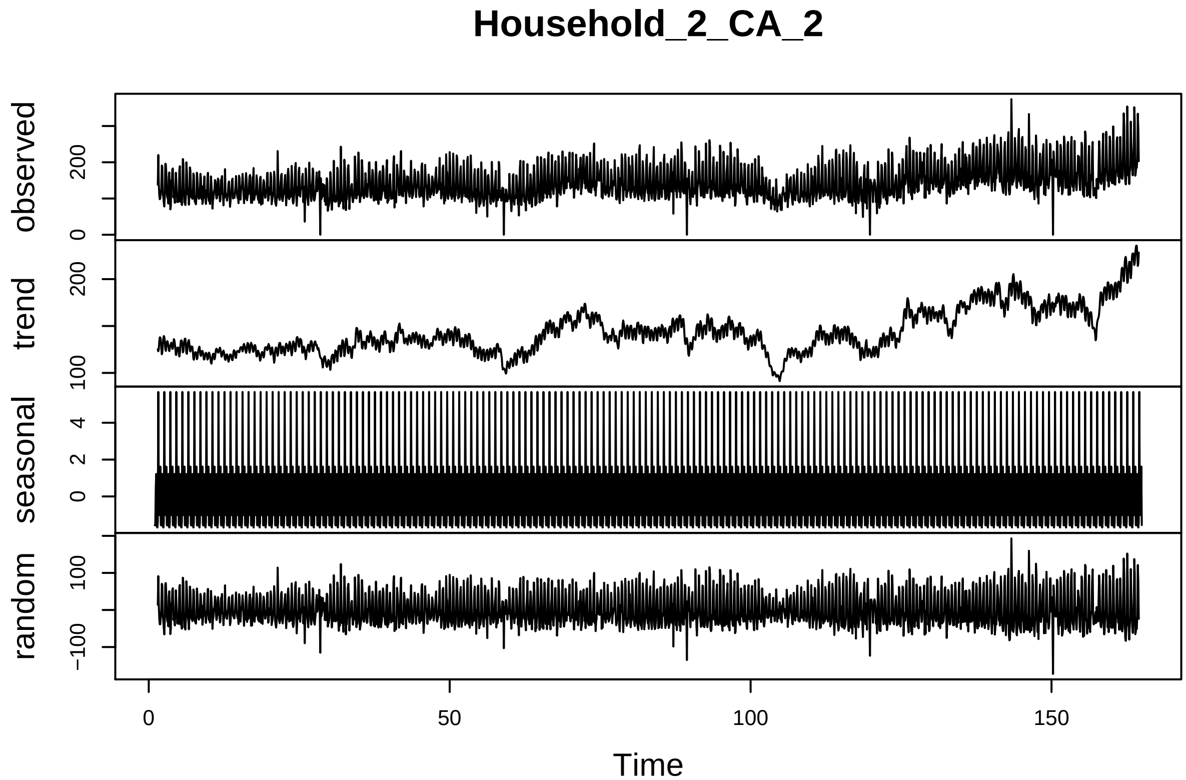
Foods_2_CA_1
observed
trend
seasonal
random
Time
0
400
800
200
4
−4
−200
50
100
150

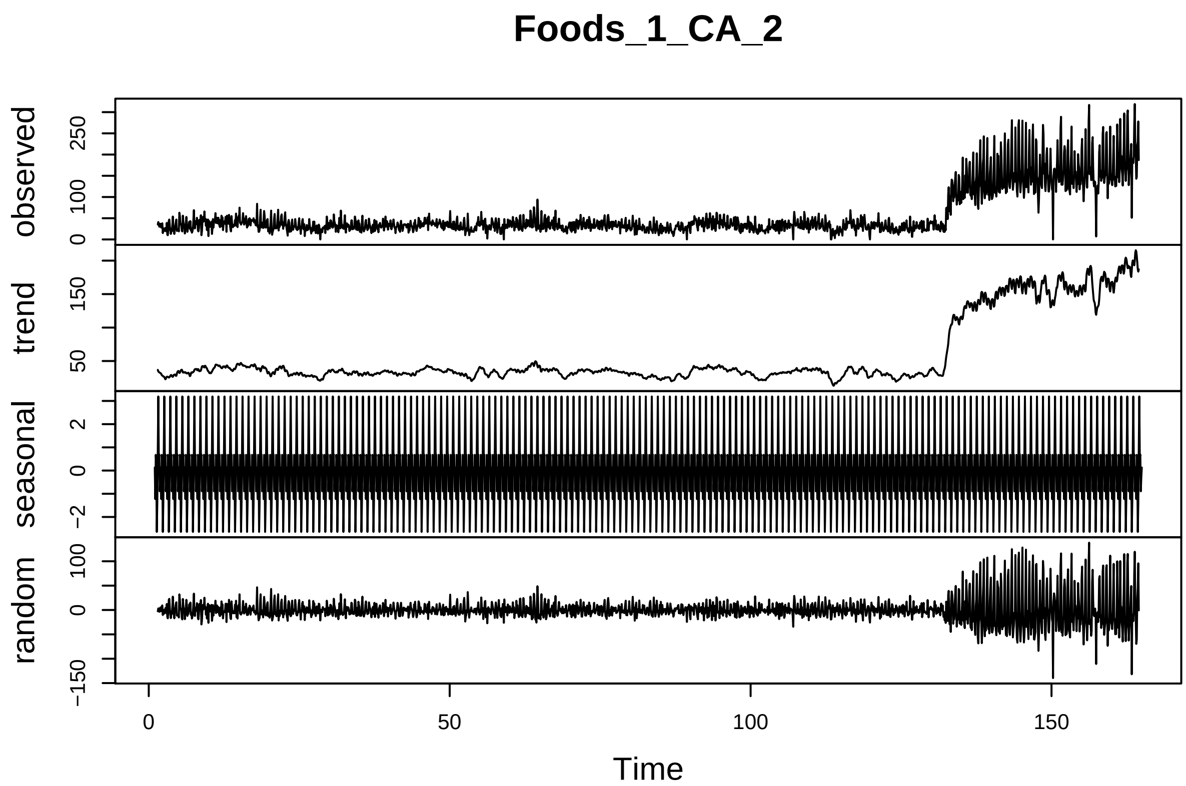

Foods_3_CA_1
observed
trend
seasonal
random
0
1500
2000
1000
20
0
−20
0
−1500
0
50
100
150
Time

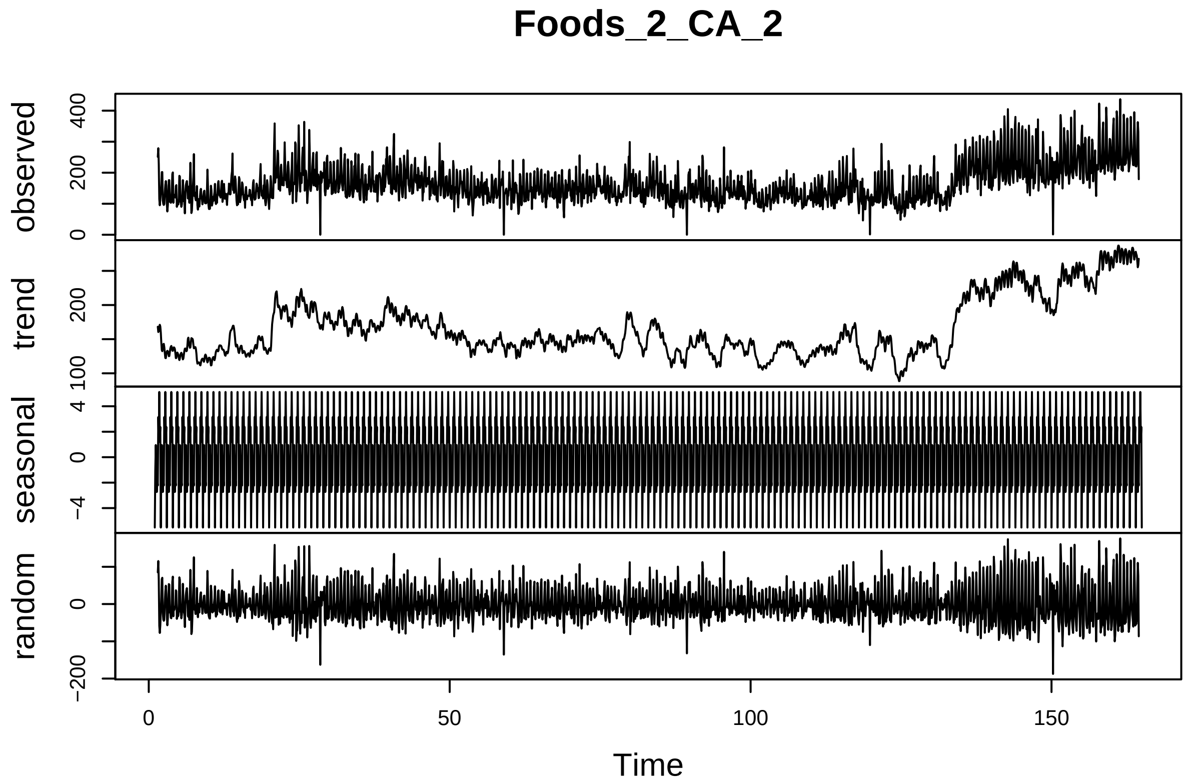

Hobbies_CA_2
observed
trend
seasonal
random
Time
500
200
350
250
150
4
0
−4
200
0
−300
0
50
100
150

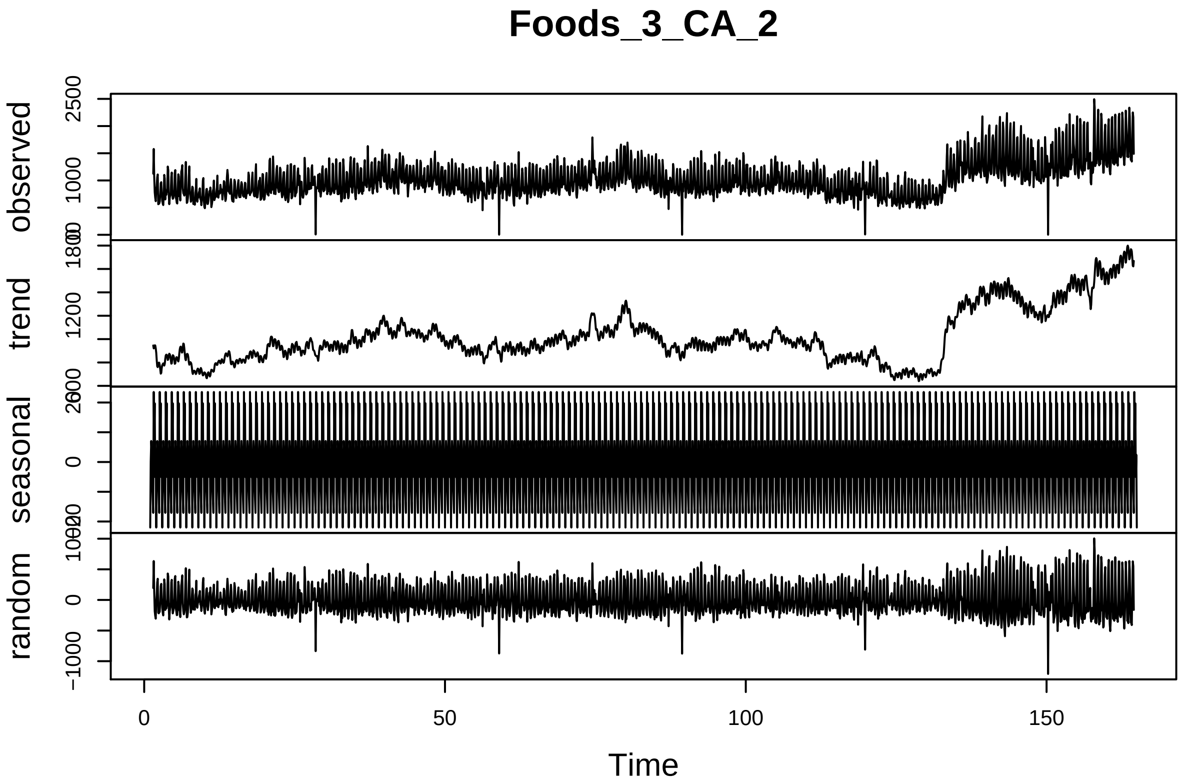


Household_1_CA_2
observed
trend
seasonal
random
Time
0
200
400
150
250
-2
2
4
6
-200
0
200
0
50
100
150

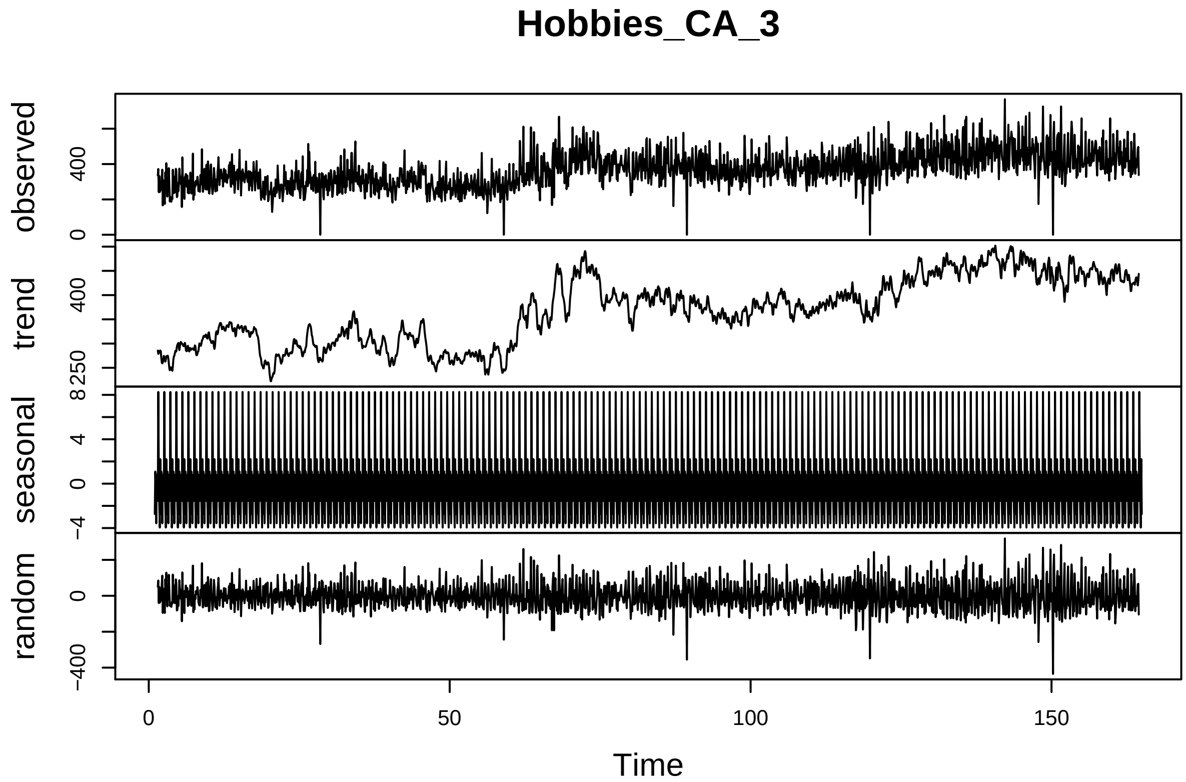

Household_2_CA_2
observed
0
200
trend
100
200
seasonal
0
2
4
random
−100
100
0
50
100
150
Time

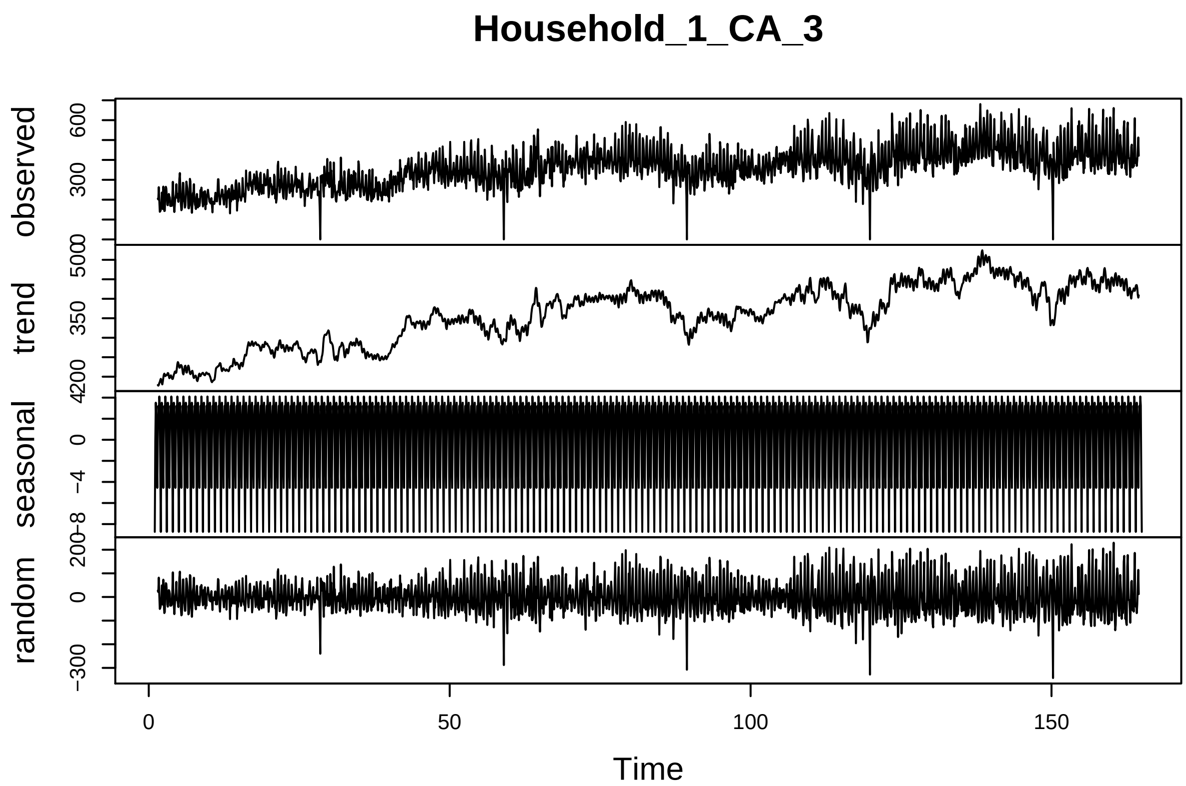

Foods_1_CA_2
observed
trend
seasonal
random
Time
0
50
100
150
250
100
0
150
50
2
0
−2
100
0
−150

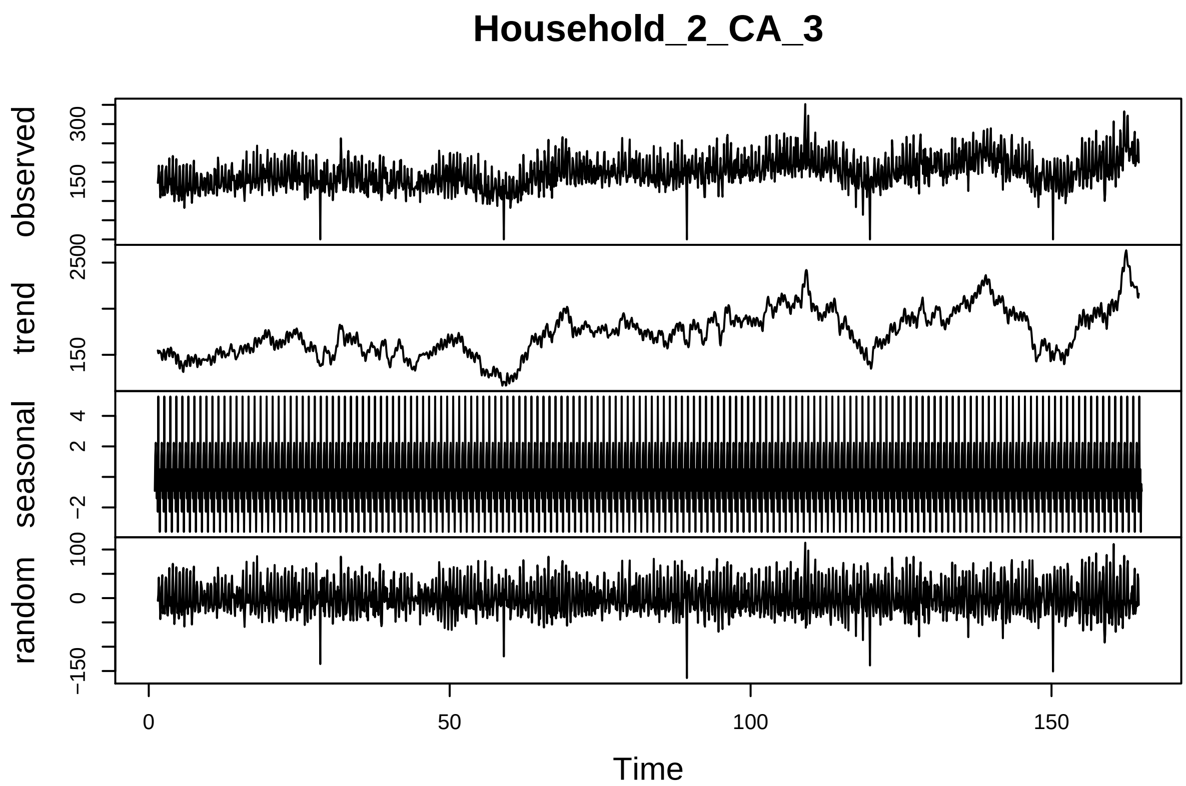

Foods_2_CA_2
observed
trend
seasonal
random
Time
0
200
400
100
200
4
0
−4
0
−200
0
50
100
150

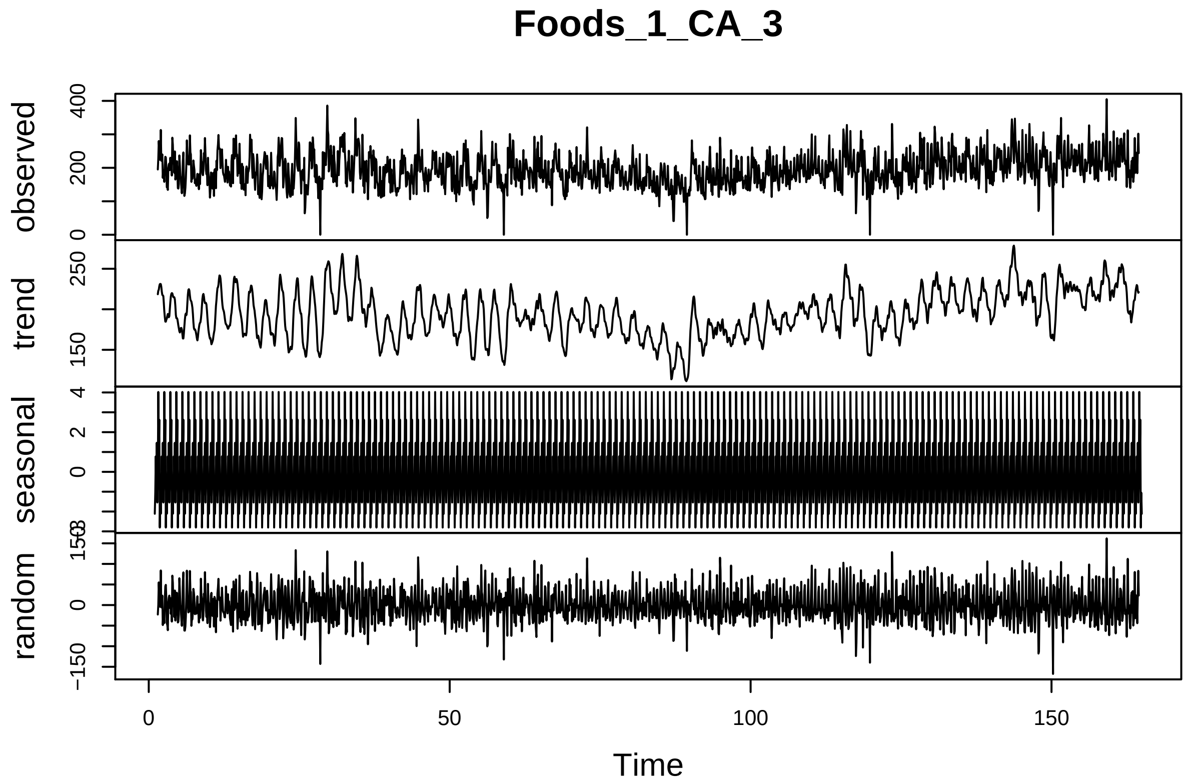


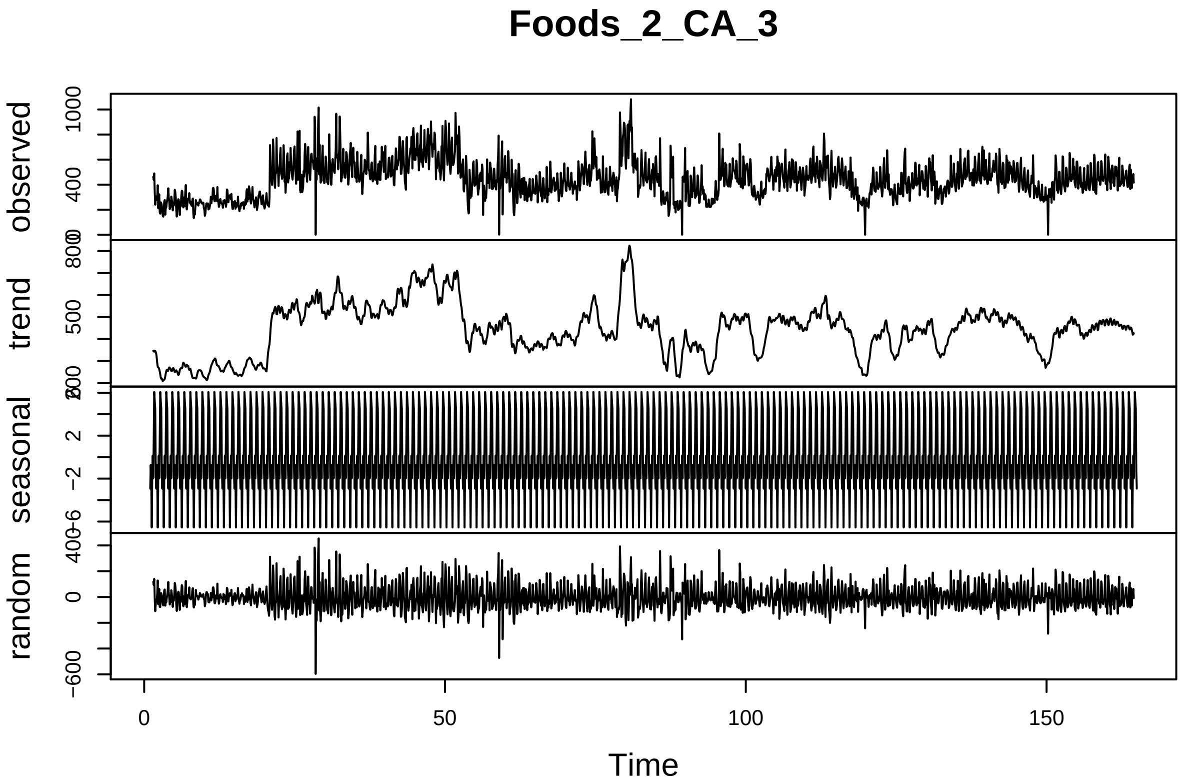
Foods_3_CA_2
observed
trend
seasonal
random
Time
0
50
100
150
2500
1000
1200
0
-1000


Hobbies_CA_3
observed
trend
seasonal
random
Time
0
400
250
400
−4
0
4
8
−400
0
0
50
100
150

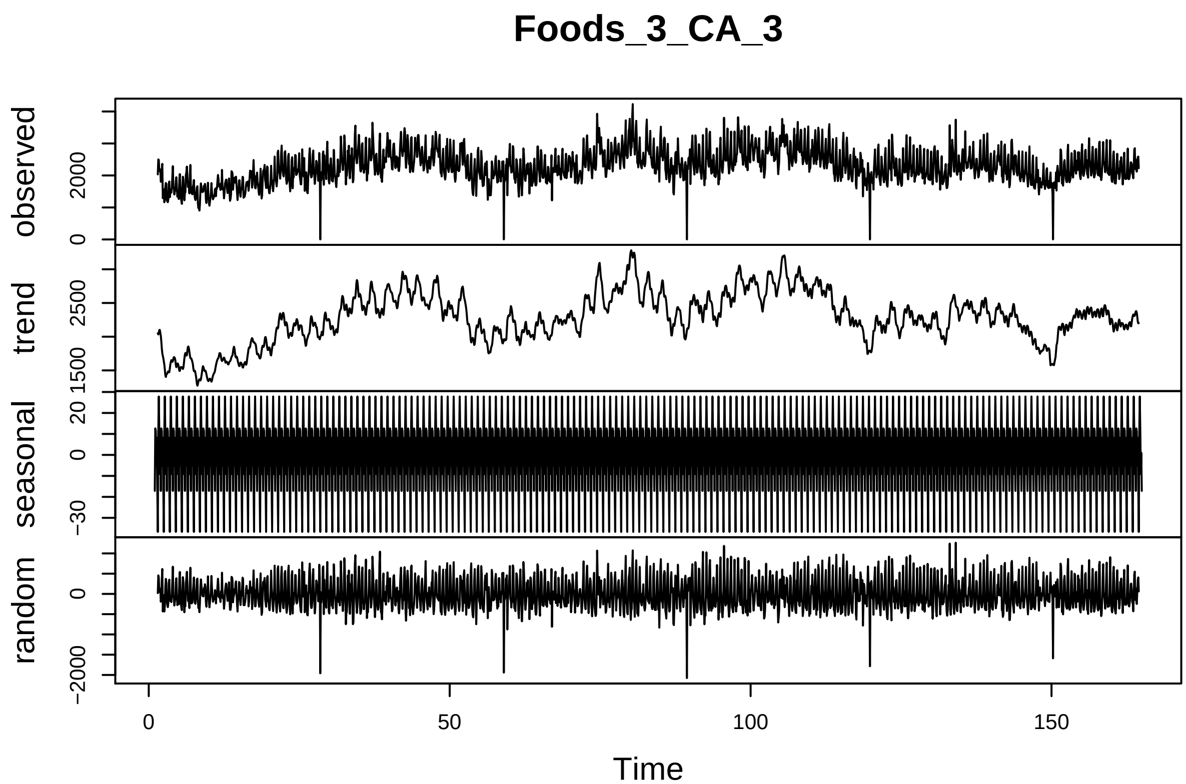

Household_1_CA_3
observed
trend
seasonal
random
Time
600
300
0
500
350
200
4
0
−4
−8
200
0
−300
0
50
100
150


Household_2_CA_3
observed
trend
seasonal
random
Time
300
150
0
2500
150
4
2
−2
100
0
−150
0
50
100
150


Foods_1_CA_3
observed
trend
seasonal
random
Time
0
50
100
150
400
200
0
250
150
4
2
0
150
0
−150



Foods_2_CA_3
observed
trend
seasonal
random
Time
0
50
100
150



#Step 3 check seasonality: ADF test

```
for (i in 2:18){
a <- adf.test(data[,i])
print(a)
}
```

```
## Warning in adf.test(data[, i]): p-value smaller than printed p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  data[, i]
## Dickey-Fuller = -7.2691, Lag order = 12, p-value = 0.01
## alternative hypothesis: stationary

## Warning in adf.test(data[, i]): p-value smaller than printed p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  data[, i]
## Dickey-Fuller = -5.9547, Lag order = 12, p-value = 0.01
## alternative hypothesis: stationary

## Warning in adf.test(data[, i]): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data[, i]
## Dickey-Fuller = -5.6564, Lag order = 12, p-value = 0.01
## alternative hypothesis: stationary

## Warning in adf.test(data[, i]): p-value smaller than printed p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  data[, i]
## Dickey-Fuller = -10.85, Lag order = 12, p-value = 0.01
## alternative hypothesis: stationary

## Warning in adf.test(data[, i]): p-value smaller than printed p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  data[, i]
## Dickey-Fuller = -5.1882, Lag order = 12, p-value = 0.01
## alternative hypothesis: stationary

## Warning in adf.test(data[, i]): p-value smaller than printed p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  data[, i]
## Dickey-Fuller = -5.2553, Lag order = 12, p-value = 0.01
## alternative hypothesis: stationary

## Warning in adf.test(data[, i]): p-value smaller than printed p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  data[, i]
## Dickey-Fuller = -4.8925, Lag order = 12, p-value = 0.01
## alternative hypothesis: stationary
##
##
##  Augmented Dickey-Fuller Test
##
## data:  data[, i]
## Dickey-Fuller = -3.8192, Lag order = 12, p-value = 0.01803
## alternative hypothesis: stationary

## Warning in adf.test(data[, i]): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data[, i]
## Dickey-Fuller = -4.5383, Lag order = 12, p-value = 0.01
## alternative hypothesis: stationary
##
##
##  Augmented Dickey-Fuller Test
##
## data:  data[, i]
## Dickey-Fuller = -2.1596, Lag order = 12, p-value = 0.5107
## alternative hypothesis: stationary
##
##
##  Augmented Dickey-Fuller Test
##
## data:  data[, i]
## Dickey-Fuller = -3.4659, Lag order = 12, p-value = 0.04557
## alternative hypothesis: stationary
##
##
##  Augmented Dickey-Fuller Test
##
## data:  data[, i]
## Dickey-Fuller = -2.8543, Lag order = 12, p-value = 0.2167
## alternative hypothesis: stationary

## Warning in adf.test(data[, i]): p-value smaller than printed p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  data[, i]
## Dickey-Fuller = -6.872, Lag order = 12, p-value = 0.01
## alternative hypothesis: stationary

## Warning in adf.test(data[, i]): p-value smaller than printed p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  data[, i]
## Dickey-Fuller = -5.6101, Lag order = 12, p-value = 0.01
## alternative hypothesis: stationary

## Warning in adf.test(data[, i]): p-value smaller than printed p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  data[, i]
## Dickey-Fuller = -5.5204, Lag order = 12, p-value = 0.01
## alternative hypothesis: stationary
```

```
## Warning in adf.test(data[, i]): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data[, i]
## Dickey-Fuller = -11.661, Lag order = 12, p-value = 0.01
## alternative hypothesis: stationary
```

```
## Warning in adf.test(data[, i]): p-value smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  data[, i]
## Dickey-Fuller = -4.4117, Lag order = 12, p-value = 0.01
## alternative hypothesis: stationary
```

#Stationarity test #KPSS test #Null hypo in KPSS is that data is stationary #If p<0.05, null hypothesis is rejected

```
#Hobbies
kpss.test(tsdata[,"Hobbies_CA_1"])
```

```
## Warning in kpss.test(tsdata[, "Hobbies_CA_1"]): p-value smaller than printed p-
## value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  tsdata[, "Hobbies_CA_1"]
## KPSS Level = 11.697, Truncation lag parameter = 8, p-value = 0.01
```

```
kpss.test(tsdata[,"Hobbies_CA_2"])
```

```
## Warning in kpss.test(tsdata[, "Hobbies_CA_2"]): p-value smaller than printed p-
## value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  tsdata[, "Hobbies_CA_2"]
## KPSS Level = 3.7587, Truncation lag parameter = 8, p-value = 0.01
```

```
kpss.test(tsdata[,"Hobbies_CA_3"])
```

```
## Warning in kpss.test(tsdata[, "Hobbies_CA_3"]): p-value smaller than printed p-
## value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  tsdata[, "Hobbies_CA_3"]
## KPSS Level = 15.171, Truncation lag parameter = 8, p-value = 0.01
```

Hobbies show nonstationarity-reject null hypothesis

Household

```
kpss.test(tsdata[,"Household_1_CA_1"])
```

```
## Warning in kpss.test(tsdata[, "Household_1_CA_1"]): p-value smaller than printed
## p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  tsdata[, "Household_1_CA_1"]
## KPSS Level = 17.395, Truncation lag parameter = 8, p-value = 0.01
```

```
kpss.test(tsdata[,"Household_1_CA_2"])
```

```
## Warning in kpss.test(tsdata[, "Household_1_CA_2"]): p-value smaller than printed
## p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  tsdata[, "Household_1_CA_2"]
## KPSS Level = 8.3618, Truncation lag parameter = 8, p-value = 0.01
```

```
kpss.test(tsdata[,"Household_1_CA_3"])
```

```
## Warning in kpss.test(tsdata[, "Household_1_CA_3"]): p-value smaller than printed
## p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  tsdata[, "Household_1_CA_3"]
## KPSS Level = 16.322, Truncation lag parameter = 8, p-value = 0.01
```

```
kpss.test(tsdata[,"Household_2_CA_1"])
```

```
## Warning in kpss.test(tsdata[, "Household_2_CA_1"]): p-value smaller than printed
## p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  tsdata[, "Household_2_CA_1"]
## KPSS Level = 13.727, Truncation lag parameter = 8, p-value = 0.01
```

```r
kpss.test(tsdata[,"Household_2_CA_2"])
```

```
## Warning in kpss.test(tsdata[, "Household_2_CA_2"]): p-value smaller than printed
## p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  tsdata[, "Household_2_CA_2"]
## KPSS Level = 9.8201, Truncation lag parameter = 8, p-value = 0.01
```

```r
kpss.test(tsdata[,"Household_2_CA_3"])
```

```
## Warning in kpss.test(tsdata[, "Household_2_CA_3"]): p-value smaller than printed
## p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  tsdata[, "Household_2_CA_3"]
## KPSS Level = 9.0445, Truncation lag parameter = 8, p-value = 0.01
```

Household_1 and Household_2 parameters show nonstationarity

```r
#Foods
kpss.test(tsdata[,"Foods_1_CA_1"])
```

```
## Warning in kpss.test(tsdata[, "Foods_1_CA_1"]): p-value smaller than printed p-
## value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  tsdata[, "Foods_1_CA_1"]
## KPSS Level = 4.0412, Truncation lag parameter = 8, p-value = 0.01
```

```r
kpss.test(tsdata[,"Foods_1_CA_2"])
```

```
## Warning in kpss.test(tsdata[, "Foods_1_CA_2"]): p-value smaller than printed p-
## value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  tsdata[, "Foods_1_CA_2"]
## KPSS Level = 10.897, Truncation lag parameter = 8, p-value = 0.01
```

```r
kpss.test(tsdata[,"Foods_1_CA_3"])
```

```
## Warning in kpss.test(tsdata[, "Foods_1_CA_3"]): p-value smaller than printed p-
## value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  tsdata[, "Foods_1_CA_3"]
## KPSS Level = 2.1387, Truncation lag parameter = 8, p-value = 0.01
```

```r
kpss.test(tsdata[,"Foods_2_CA_1"])
```

```
## Warning in kpss.test(tsdata[, "Foods_2_CA_1"]): p-value smaller than printed p-
## value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  tsdata[, "Foods_2_CA_1"]
## KPSS Level = 1.6452, Truncation lag parameter = 8, p-value = 0.01
```

```r
kpss.test(tsdata[,"Foods_2_CA_2"])
```

```
## Warning in kpss.test(tsdata[, "Foods_2_CA_2"]): p-value smaller than printed p-
## value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  tsdata[, "Foods_2_CA_2"]
## KPSS Level = 5.1003, Truncation lag parameter = 8, p-value = 0.01
```

```r
kpss.test(tsdata[,"Foods_2_CA_3"])
```

```
## Warning in kpss.test(tsdata[, "Foods_2_CA_3"]): p-value smaller than printed p-
## value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  tsdata[, "Foods_2_CA_3"]
## KPSS Level = 0.9268, Truncation lag parameter = 8, p-value = 0.01
```

```r
kpss.test(tsdata[,"Foods_3_CA_1"])
```

```
## Warning in kpss.test(tsdata[, "Foods_3_CA_1"]): p-value smaller than printed p-
## value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  tsdata[, "Foods_3_CA_1"]
## KPSS Level = 5.6949, Truncation lag parameter = 8, p-value = 0.01
```

```
kpss.test(tsdata[,"Foods_3_CA_2"])
```

```
## Warning in kpss.test(tsdata[, "Foods_3_CA_2"]): p-value smaller than printed p-
## value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  tsdata[, "Foods_3_CA_2"]
## KPSS Level = 6.6207, Truncation lag parameter = 8, p-value = 0.01
```

```
kpss.test(tsdata[,"Foods_3_CA_3"])
```

```
## Warning in kpss.test(tsdata[, "Foods_3_CA_3"]): p-value smaller than printed p-
## value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  tsdata[, "Foods_3_CA_3"]
## KPSS Level = 3.3164, Truncation lag parameter = 8, p-value = 0.01
```

Foods_1,2 and 3 parameters show nonstationarity

Summary of analysis: by checking both the graphs and stationary test, all the time series show nonstationary and they have both the seasonal and trend part.