M3: Class Exercise on Trend and Season

Luana Lima

Setting R code chunk options

First R code chunk is used for setting the options for all R code chunks. The choice echo=TRUE means both code and output will appear on report, include = FALSE neither code nor output is printed.

Loading packages and initializing

Second R code chunk is for loading packages. By setting message = FALSE, the code will appear but not the output.

```
library(lubridate)
library(ggplot2)
library(forecast)
library(Kendall)
library(tseries)
```

Importing data

Let's continue working with our inflow data for reservoirs in Brazil.

```
#Importing time series data from text file#
raw_inflow_data <- read.table(file="../Data/inflowtimeseries.txt",header=FALSE,skip=0)

#Trim the table to include only columns you need
nhydro <- ncol(raw_inflow_data)-2
nobs <- nrow(raw_inflow_data)

#If your file does not have header like this one you can add column names after
#creating the data frame
colnames(raw_inflow_data)=c("Month","Year", "HP1", "HP2","HP3","HP4", "HP5",
                            "HP6","HP7", "HP8","HP9","HP10", "HP11","HP12",
                            "HP13", "HP14","HP15")

#Checking data
head(raw_inflow_data)
```

```
##   Month Year  HP1  HP2  HP3  HP4  HP5  HP6 HP7  HP8 HP9 HP10 HP11 HP12 HP13
## 1   Jan 1931 4782 4076 2518 2450 2649 1462 450  968 246 2636  452 4870  452
## 2   Feb 1931 7323 7681 4188  150 2401  758 554  219  74 4158  457 4550  796
## 3   Mar 1931 8266 5921 3253 2389 3261  707 615  333 123 3847  631 6537  804
## 4   Apr 1931 6247 4600 2449 1253 2006  469 474  297 113 3291  510 7298  644
```

```
## 5   May 1931 3642 2789 1651 2374 2454 3167 378 3295 938 1956  276 4942  421
## 6   Jun 1931 2425 2062 1270 2672 2433 3236 301 2547 951 1371  201 2478  305
##    HP14  HP15
## 1 17342 31270
## 2 21530 43827
## 3 33299 49884
## 4 34674 43962
## 5 15184 35156
## 6  8611 25764
```

```
str(raw_inflow_data)
```

```
## 'data.frame':    972 obs. of  17 variables:
##  $ Month: chr  "Jan" "Feb" "Mar" "Apr" ...
##  $ Year : int  1931 1931 1931 1931 1931 1931 1931 1931 1931 1931 ...
##  $ HP1  : int  4782 7323 8266 6247 3642 2425 2158 1854 1839 1896 ...
##  $ HP2  : int  4076 7681 5921 4600 2789 2062 1644 1301 1439 1340 ...
##  $ HP3  : int  2518 4188 3253 2449 1651 1270 1204 1152 1297 1259 ...
##  $ HP4  : int  2450 150 2389 1253 2374 2672 1238 605 1016 674 ...
##  $ HP5  : int  2649 2401 3261 2006 2454 2433 1798 1160 1584 1563 ...
##  $ HP6  : int  1462 758 707 469 3167 3236 1957 844 1937 1484 ...
##  $ HP7  : int  450 554 615 474 378 301 256 244 222 355 ...
##  $ HP8  : int  968 219 333 297 3295 2547 2585 1173 3596 1140 ...
##  $ HP9  : int  246 74 123 113 938 951 883 404 378 211 ...
##  $ HP10 : int  2636 4158 3847 3291 1956 1371 1186 1049 1162 1507 ...
##  $ HP11 : int  452 457 631 510 276 201 213 196 161 208 ...
##  $ HP12 : int  4870 4550 6537 7298 4942 2478 1905 1647 1453 1358 ...
##  $ HP13 : int  452 796 804 644 421 305 261 246 250 328 ...
##  $ HP14 : int  17342 21530 33299 34674 15184 8611 5939 4259 3282 3305 ...
##  $ HP15 : int  31270 43827 49884 43962 35156 25764 18109 13320 8225 8900 ...
```

Creating the date object

Here we use the function my() from package lubridate.

```
#using package lubridate
my_date <- paste(raw_inflow_data[,1],raw_inflow_data[,2],sep="-")
my_date <- my(my_date)  #function my from package lubridate
head(my_date)
```

```
## [1] "1931-01-01" "1931-02-01" "1931-03-01" "1931-04-01" "1931-05-01"
## [6] "1931-06-01"
```

```
#add that to inflow_data and store in a new data frame
inflow_data <- cbind(my_date,raw_inflow_data[,3:(3+nhydro-1)])
head(inflow_data)
```

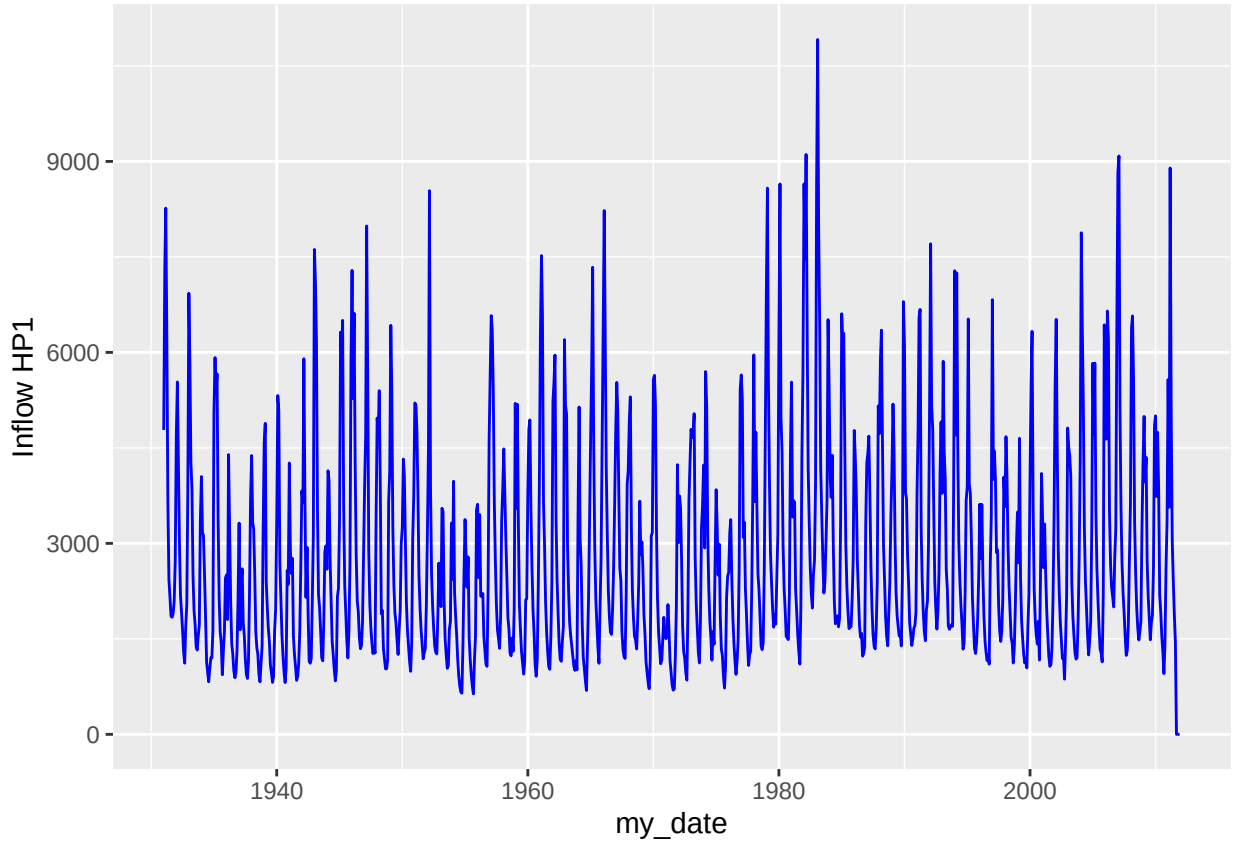
```
##      my_date  HP1  HP2  HP3  HP4  HP5  HP6 HP7  HP8 HP9 HP10 HP11 HP12 HP13
## 1 1931-01-01 4782 4076 2518 2450 2649 1462 450  968 246 2636  452 4870  452
## 2 1931-02-01 7323 7681 4188  150 2401  758 554  219  74 4158  457 4550  796
## 3 1931-03-01 8266 5921 3253 2389 3261  707 615  333 123 3847  631 6537  804
```

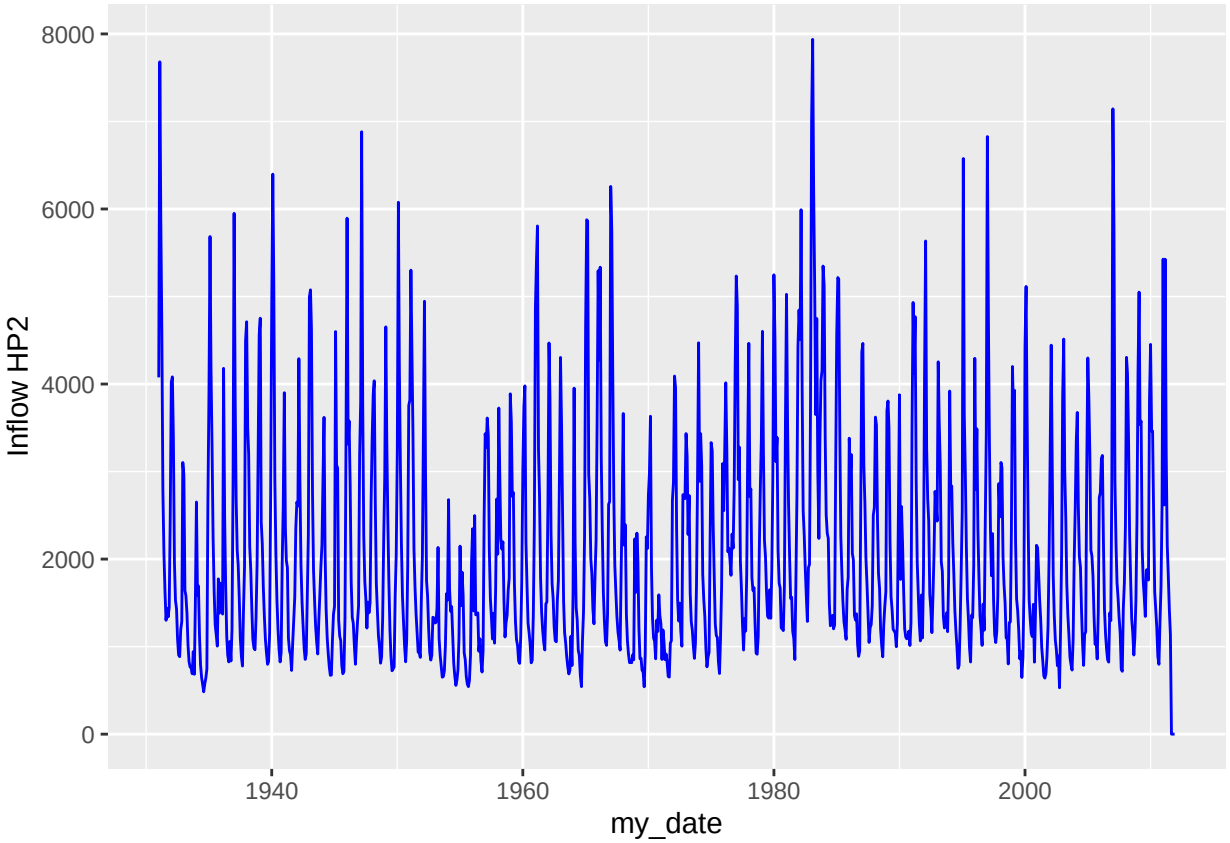
```
## 4 1931-04-01 6247 4600 2449 1253 2006  469 474  297 113 3291  510 7298  644
## 5 1931-05-01 3642 2789 1651 2374 2454 3167 378 3295 938 1956  276 4942  421
## 6 1931-06-01 2425 2062 1270 2672 2433 3236 301 2547 951 1371  201 2478  305
##    HP14  HP15
## 1 17342 31270
## 2 21530 43827
## 3 33299 49884
## 4 34674 43962
## 5 15184 35156
## 6  8611 25764
```

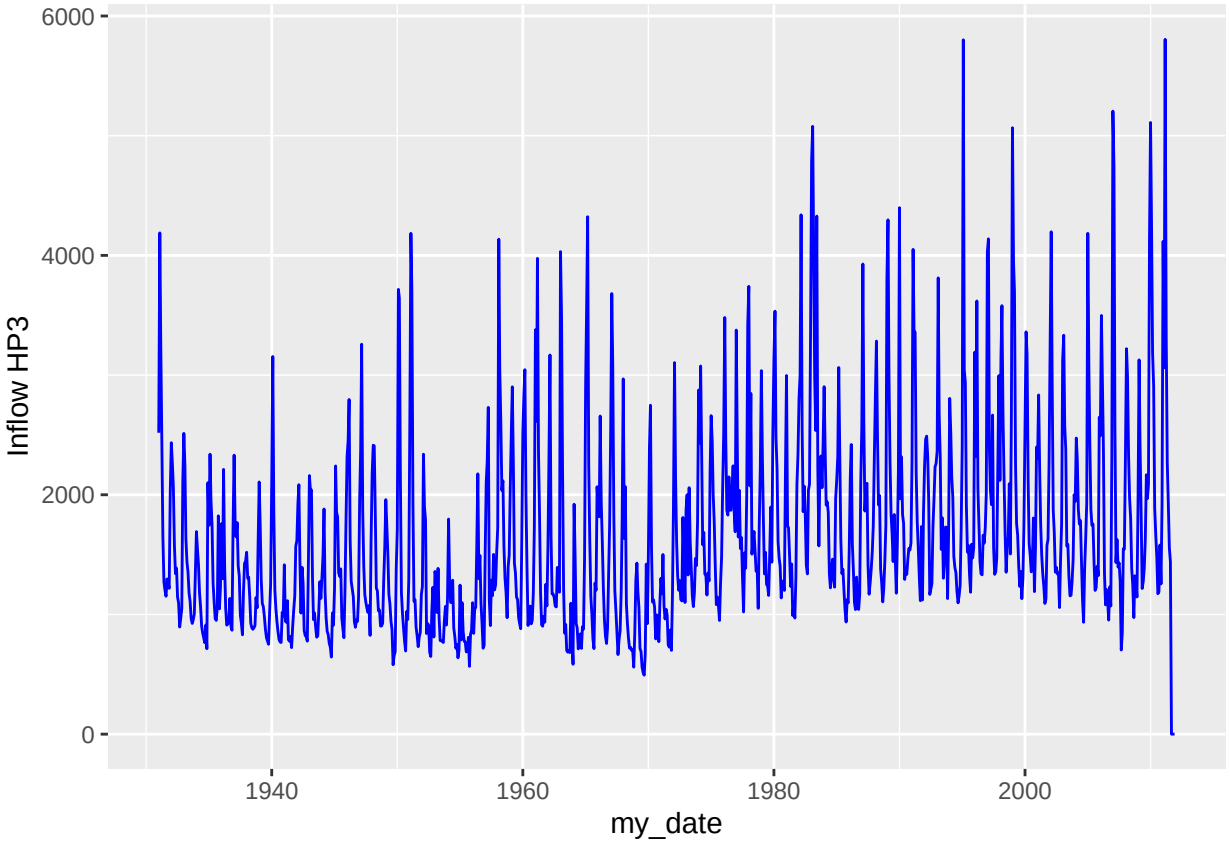
Initial Plots

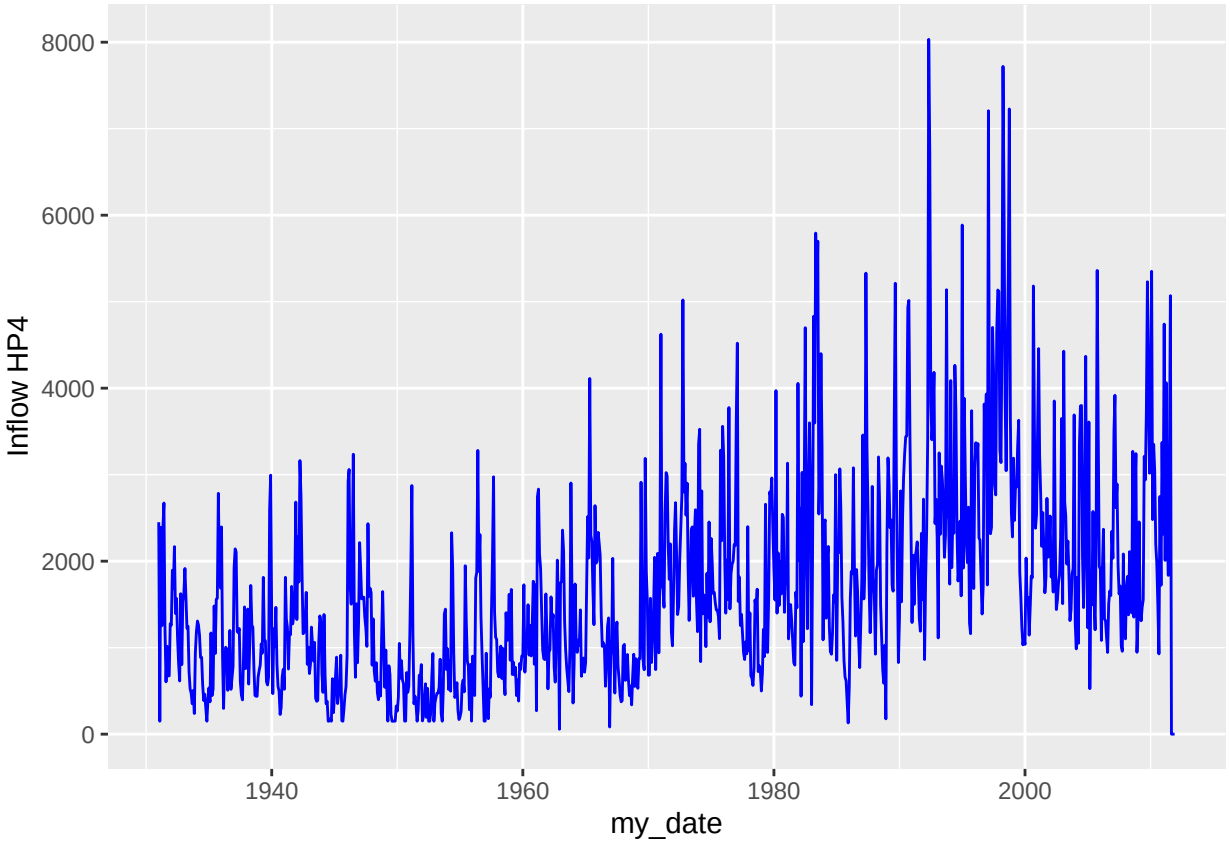
Initial time series plot.

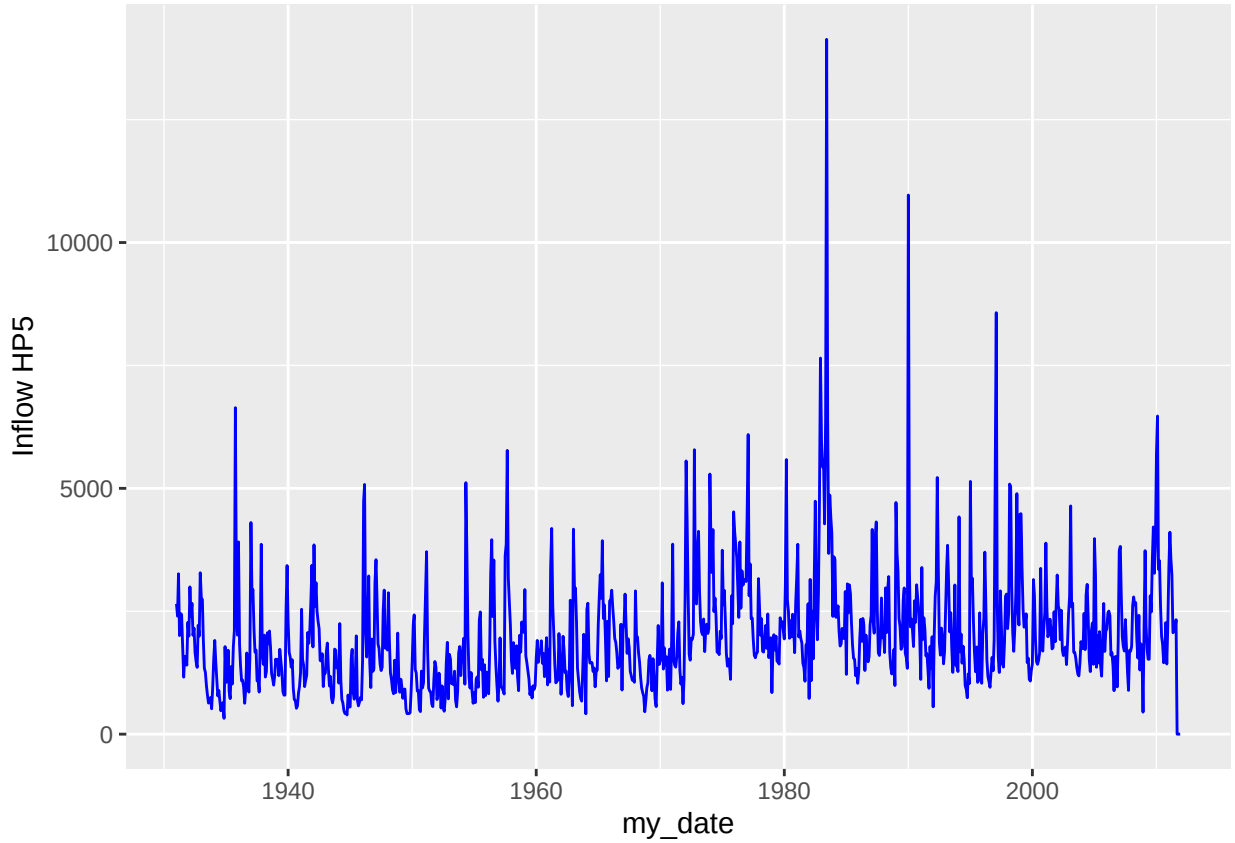
```
#using package ggplot2
for(i in 1:nhydro){
  print(ggplot(inflow_data, aes(x=my_date, y=inflow_data[,(1+i)])) +
            geom_line(color="blue") +
            ylab(paste0("Inflow ",colnames(inflow_data)[(1+i)],sep=""))
        )
}
```

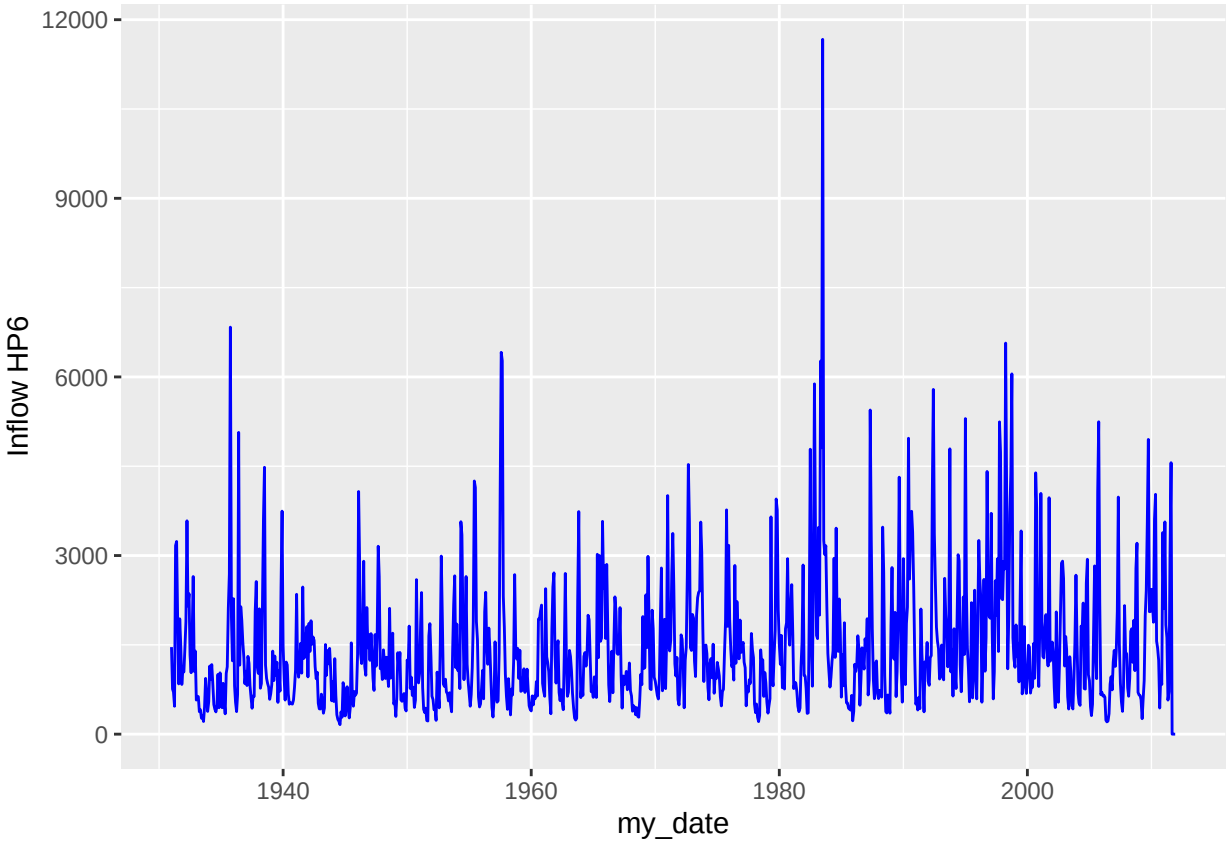


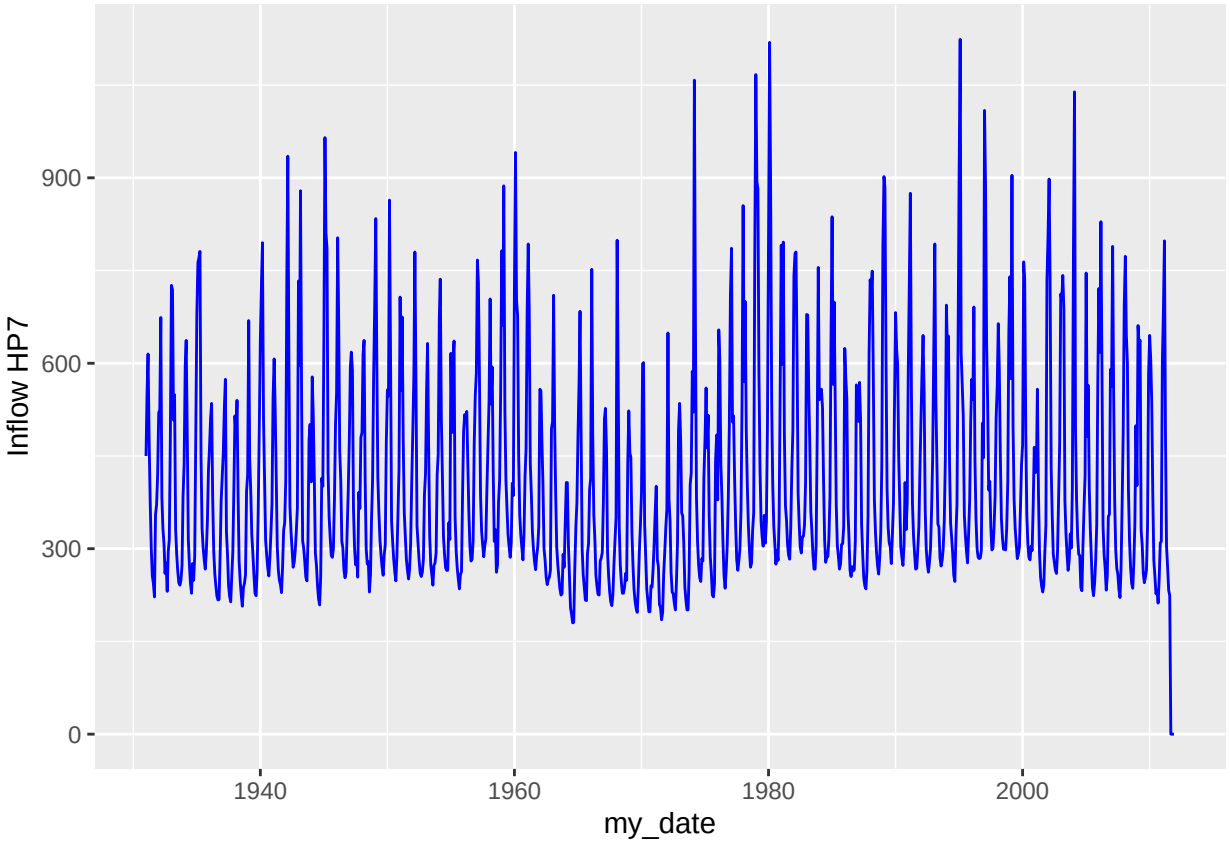

8000
6000
4000
2000
0
Inflow HP2
1940
1960
1980
2000
my_date

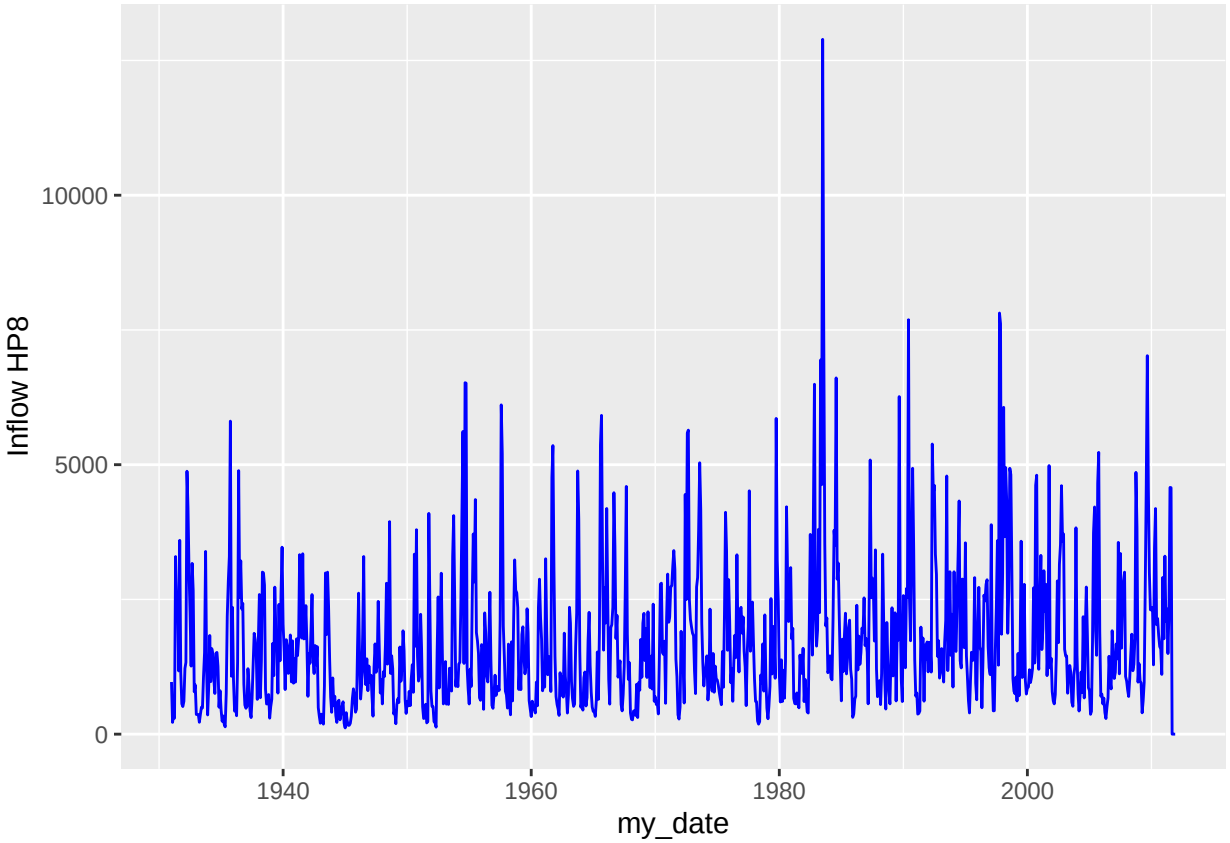

6000
4000
2000
0
Inflow HP3
1940
1960
1980
2000
my_date

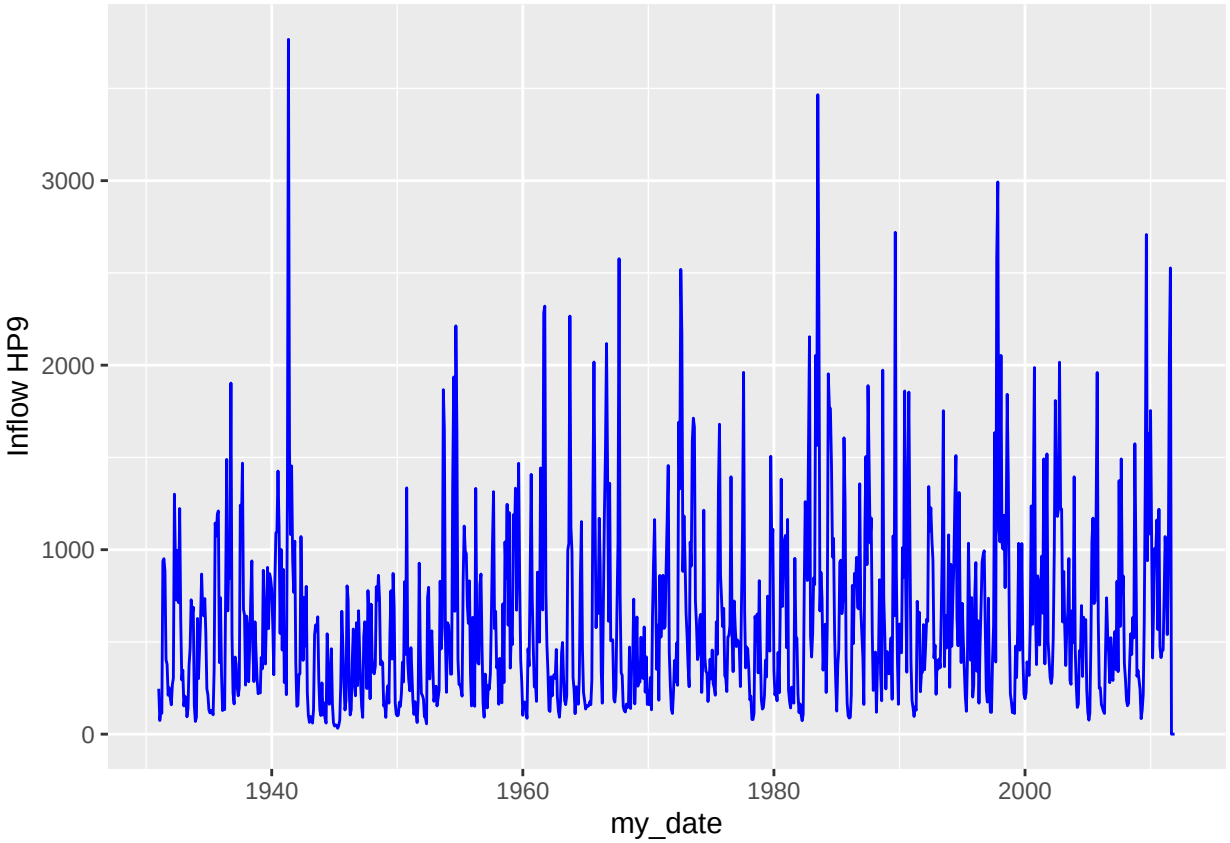

8000
6000
4000
2000
0
Inflow HP4
1940
1960
1980
2000
my_date

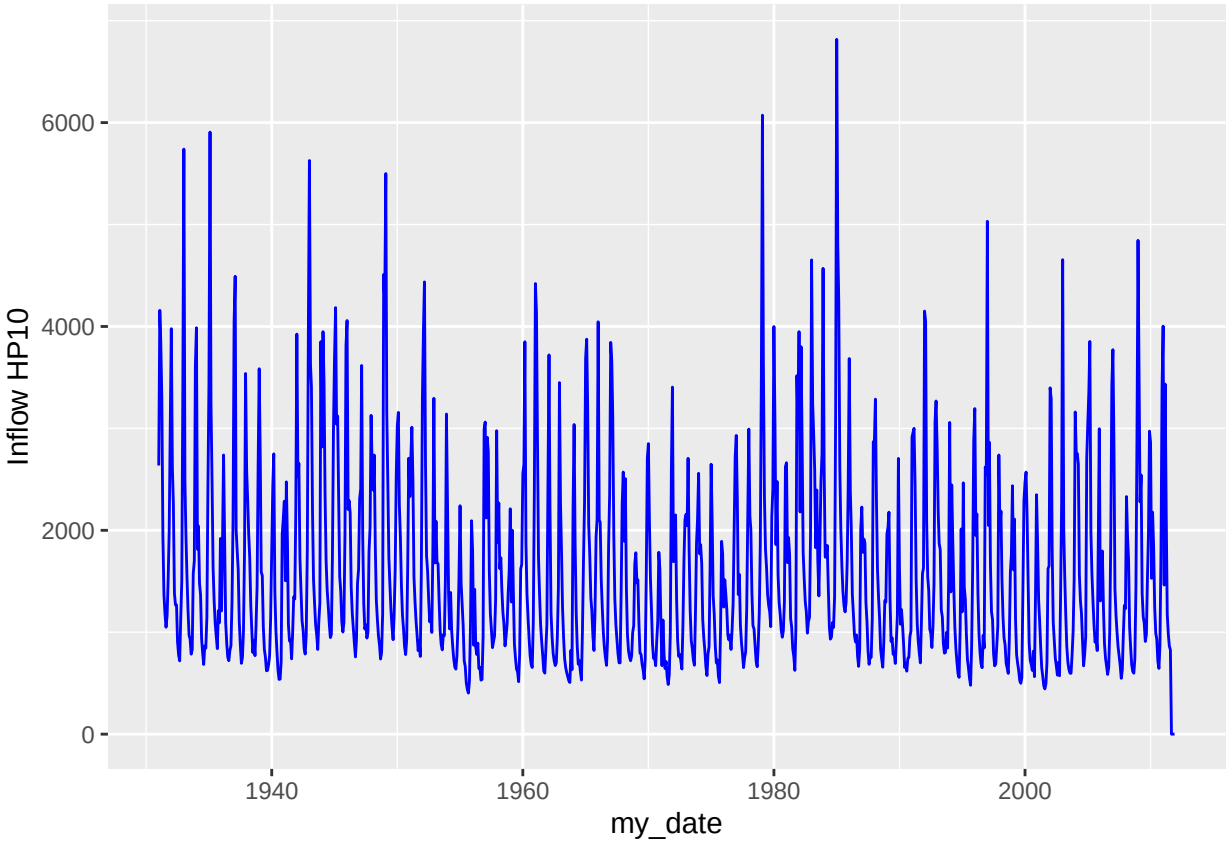

Inflow HP5
10000
5000
0
1940
1960
1980
2000
my_date

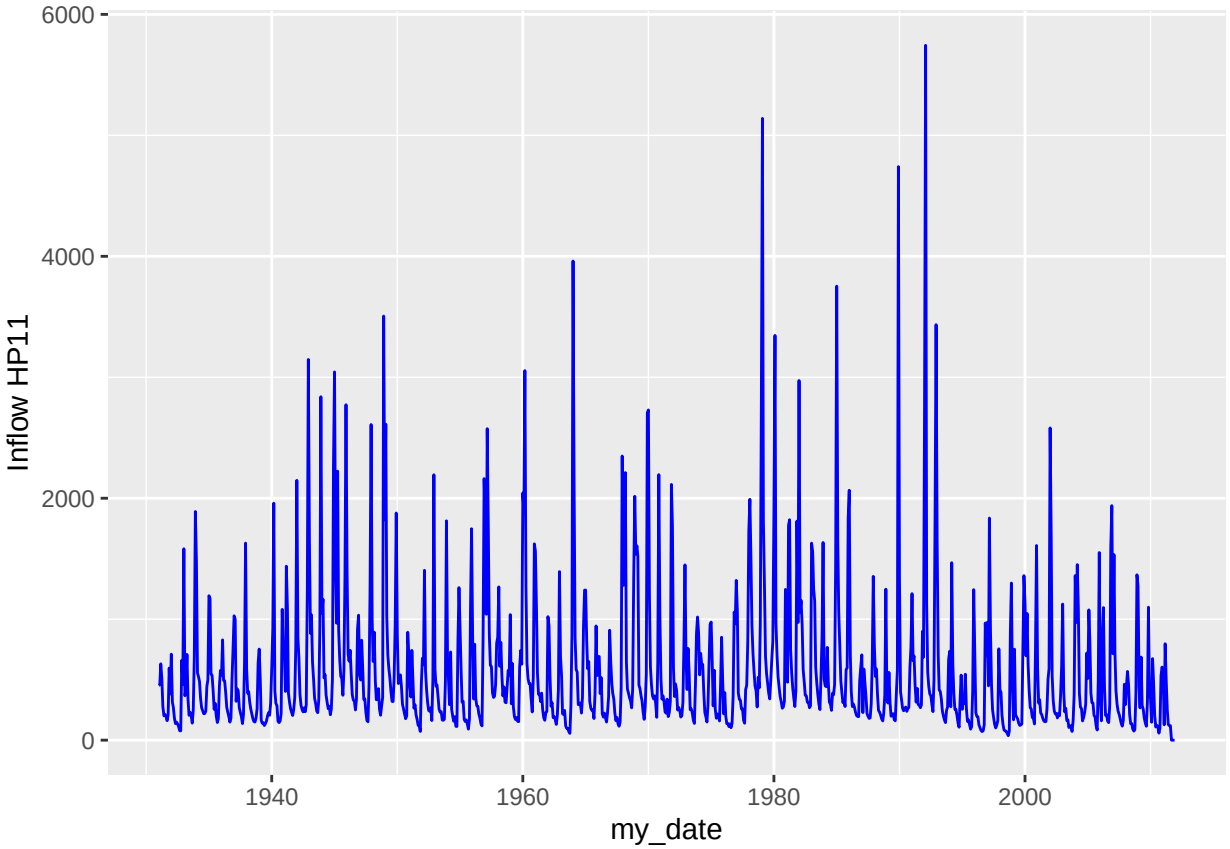

12000
9000
6000
3000
0
Inflow HP6
1940
1960
1980
2000
my_date

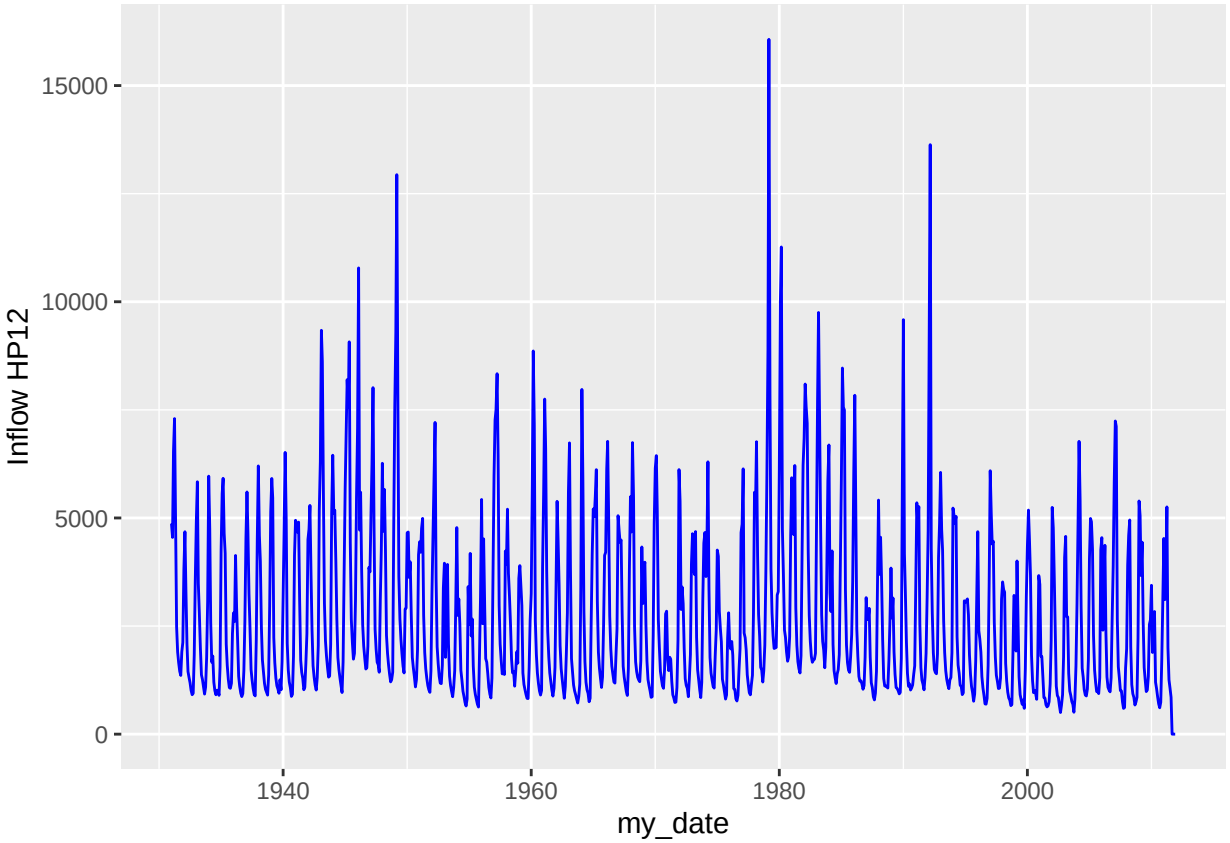

Inflow HP7
900
600
300
0
1940
1960
1980
2000
my_date

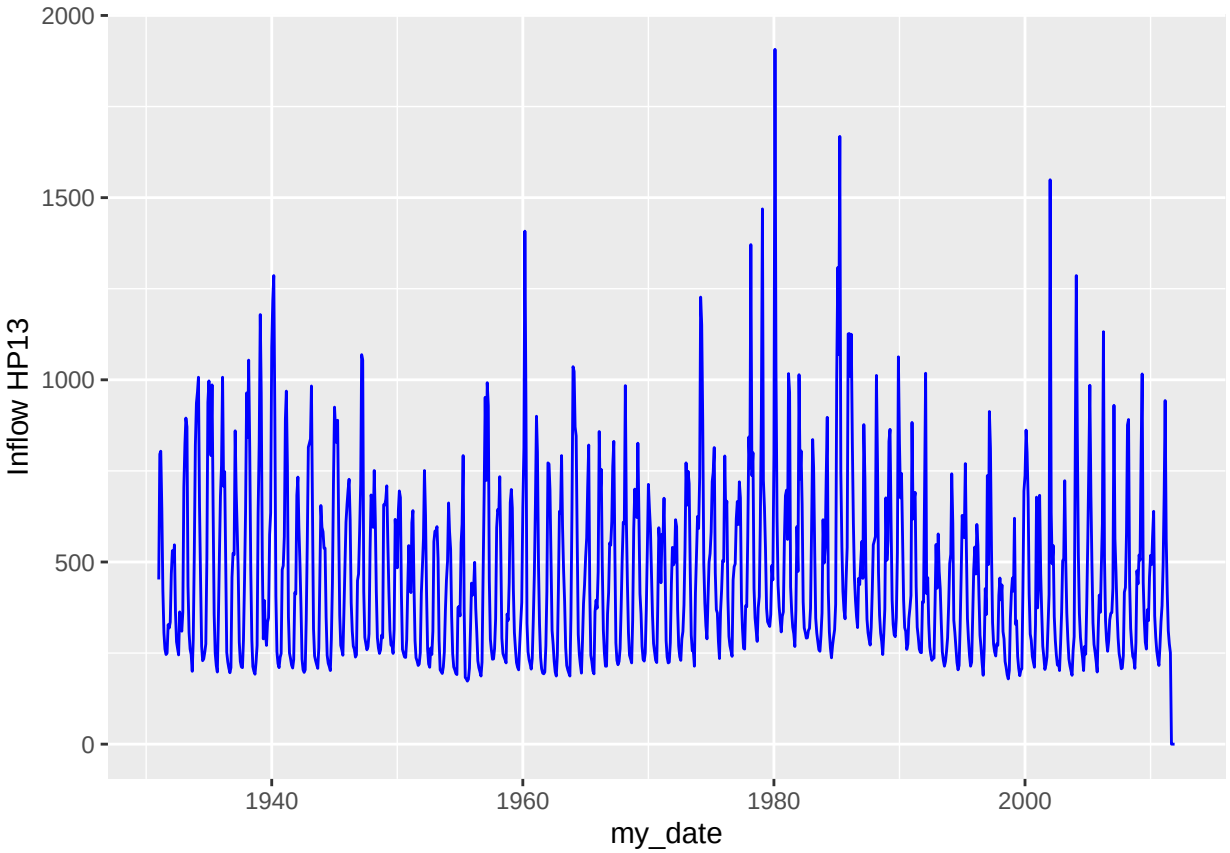

Inflow HP8
10000
5000
0
1940
1960
1980
2000
my_date

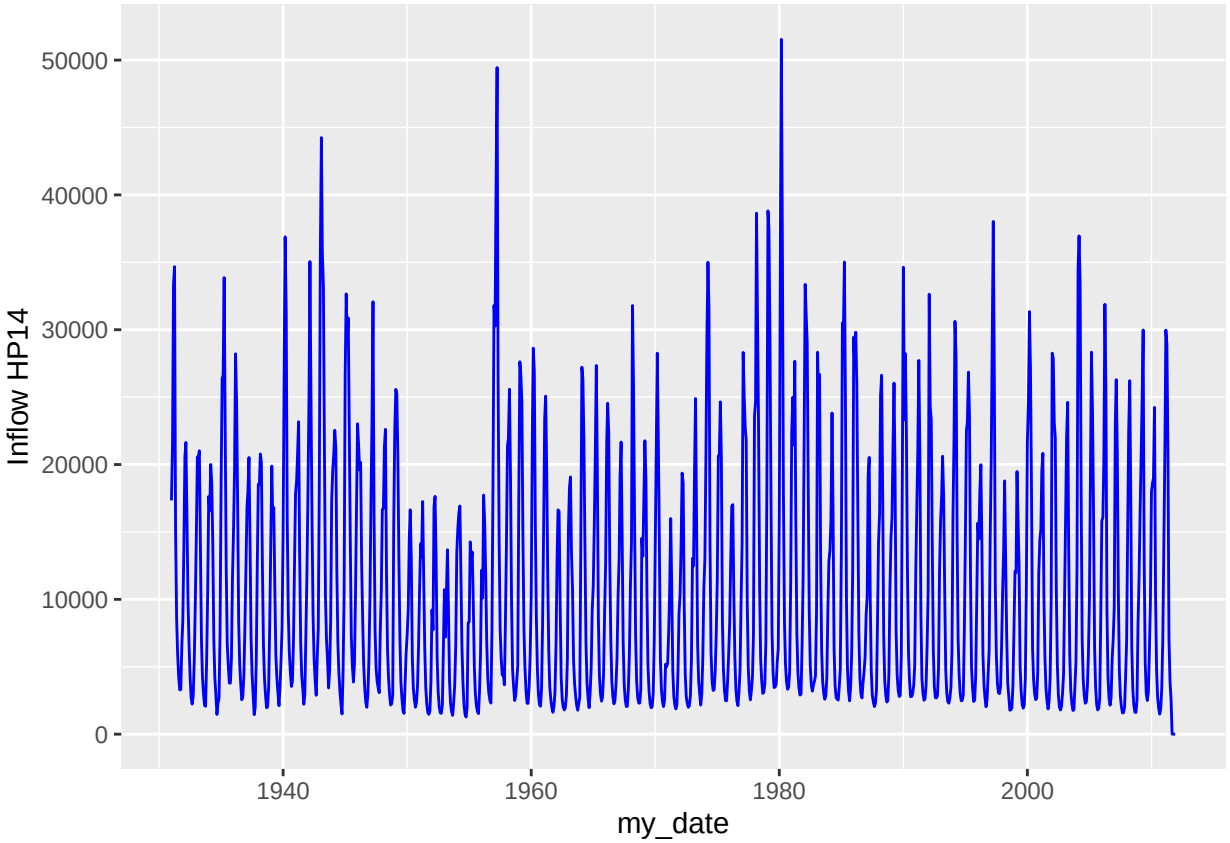

Inflow HP9
3000
2000
1000
0
1940
1960
1980
2000
my_date

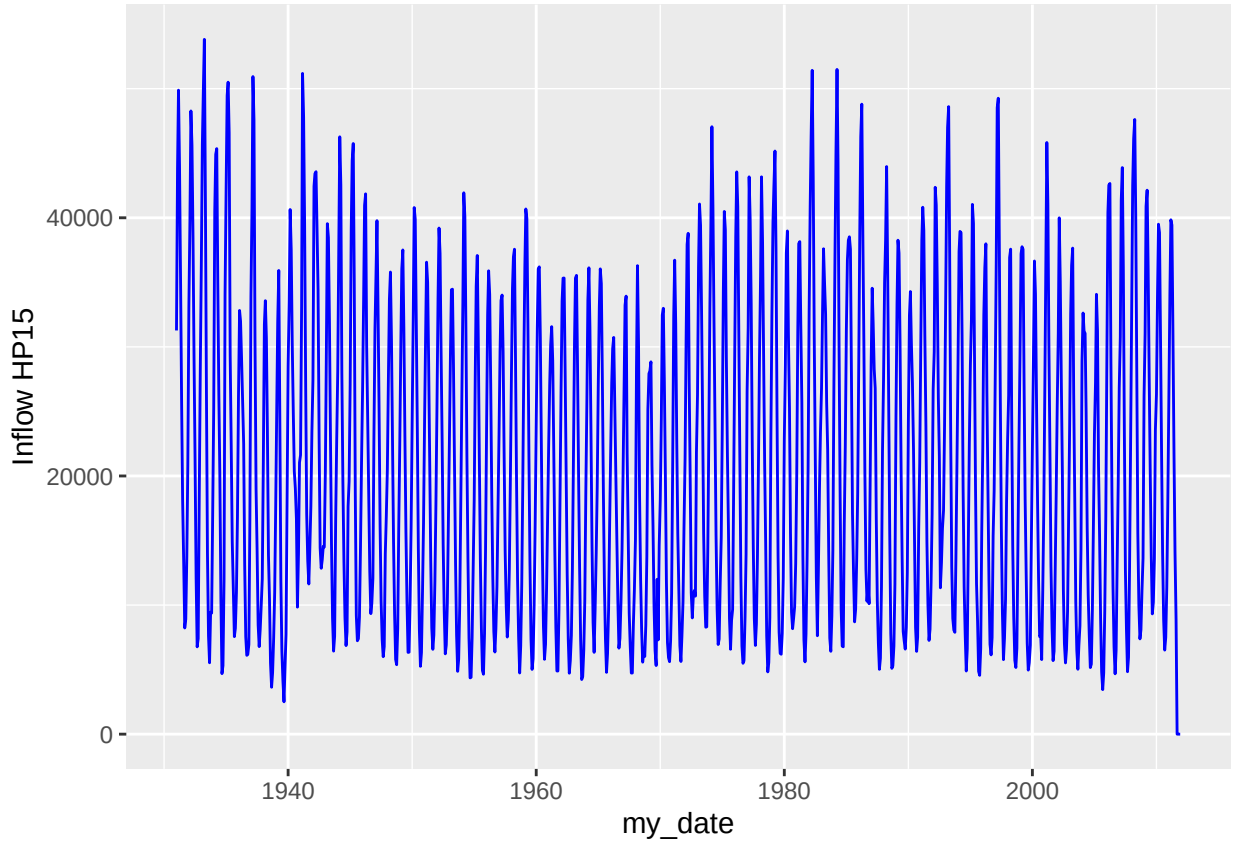

Inflow HP10
6000
4000
2000
0
1940
1960
1980
2000
my_date


6000
4000
2000
0
Inflow HP11
1940
1960
1980
2000
my_date


15000
10000
5000
0
Inflow HP12
1940
1960
1980
2000
my_date


2000
1500
1000
500
0
Inflow HP13
1940
1960
1980
2000
my_date


Inflow HP14
50000
40000
30000
20000
10000
0
1940
1960
1980
2000
my_date



Zeros in the end on data

The initial plots showed that we have zeros in the end of the data set. It could be missing observation or observation that haven't been observed yet. Use the tail() to find out how many zeros you have and how many lines you will need to remove.

```
#check the final obs on data
tail(inflow_data)
```

```
##        my_date  HP1  HP2  HP3  HP4  HP5  HP6 HP7  HP8  HP9 HP10 HP11 HP12 HP13
## 967 2011-07-01 1883 1426 1560 2930 2105 2988 233 4578 2045  864  119 1068  275
## 968 2011-08-01 1444 1139 1441 5069 2328 4559 224 4573 2527  827  120  854  251
## 969 2011-09-01    0    0    0    0    0    0   0    0    0    0    0    0    0
## 970 2011-10-01    0    0    0    0    0    0   0    0    0    0    0    0    0
## 971 2011-11-01    0    0    0    0    0    0   0    0    0    0    0    0    0
## 972 2011-12-01    0    0    0    0    0    0   0    0    0    0    0    0    0
##     HP14  HP15
## 967 3910 14162
## 968 2561  8896
## 969    0     0
## 970    0     0
## 971    0     0
## 972    0     0
```

Note our last observation is from August 2011 but the data file was filled with zeros. Let's remove the last four rows of our data set.

```
#Remove last year by replacing current data frame
inflow_data <- inflow_data[1:(nobs-12),]

#update object with number of observations
nobs <- nobs-12

#Tail again to check if the rows were correctly removed
tail(inflow_data)
```

```
##        my_date  HP1  HP2  HP3  HP4  HP5  HP6 HP7  HP8  HP9 HP10 HP11 HP12 HP13
## 955 2010-07-01 1539 1214 1481 1978 1828 1449 227 2146 1161  932  124  867  255
## 956 2010-08-01 1289  886 1173 1490 1452 1238 233 1834  567  715  108  702  233
## 957 2010-09-01  953  798 1189  928 1564  439 212 1626 1219  645   58  610  216
## 958 2010-10-01 1411 1265 1580 2748 2268  971 251 1581  476  871  100  738  268
## 959 2010-11-01 2608 1681 1255 1721 1427  835 309 1109  415 1807  534 1726  336
## 960 2010-12-01 3338 2608 1921 3373 2203 3386 312 2908  453 3402  604 3064  380
##     HP14  HP15
## 955 2746 14043
## 956 1931  8815
## 957 1485  6512
## 958 1900  7492
## 959 3470 11387
## 960 7027 17839
```

Fixed!

Transforming data into time series object

Many of the functions we will use require a time series object. You can transform your data in a time series using the function *ts()*.

```
ts_inflow_data <- ts(inflow_data[,2:(2+nhydro-1)],start=c(1931,1),frequency=12)
#note that we are only transforming columns with inflow data, not the date columns  #start=my_date[1],e
head(ts_inflow_data,15)
```

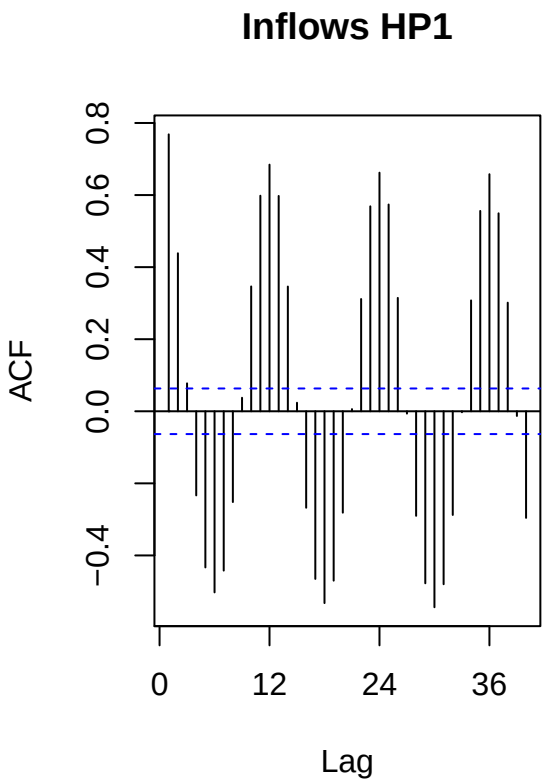
```
##           HP1  HP2  HP3  HP4  HP5  HP6 HP7  HP8 HP9 HP10 HP11 HP12 HP13  HP14
## Jan 1931 4782 4076 2518 2450 2649 1462 450  968 246 2636  452 4870  452 17342
## Feb 1931 7323 7681 4188  150 2401  758 554  219  74 4158  457 4550  796 21530
## Mar 1931 8266 5921 3253 2389 3261  707 615  333 123 3847  631 6537  804 33299
## Apr 1931 6247 4600 2449 1253 2006  469 474  297 113 3291  510 7298  644 34674
## May 1931 3642 2789 1651 2374 2454 3167 378 3295 938 1956  276 4942  421 15184
## Jun 1931 2425 2062 1270 2672 2433 3236 301 2547 951 1371  201 2478  305  8611
## Jul 1931 2158 1644 1204 1238 1798 1957 256 2585 883 1186  213 1905  261  5939
## Aug 1931 1854 1301 1152  605 1160  844 244 1173 404 1049  196 1647  246  4259
## Sep 1931 1839 1439 1297 1016 1584 1937 222 3596 378 1162  161 1453  250  3282
## Oct 1931 1896 1340 1259  674 1563 1484 355 1140 211 1507  208 1358  328  3305
## Nov 1931 2095 1447 1218  674 1404  835 371  563 252 1996  596 1905  319  6500
## Dec 1931 2725 2479 2013 1278 2272 1073 419  512 197 3015  381 2121  335  8461
## Jan 1932 4679 4021 2435 1259 1995 1044 520  609 159 3978  711 3811  467 14002
```

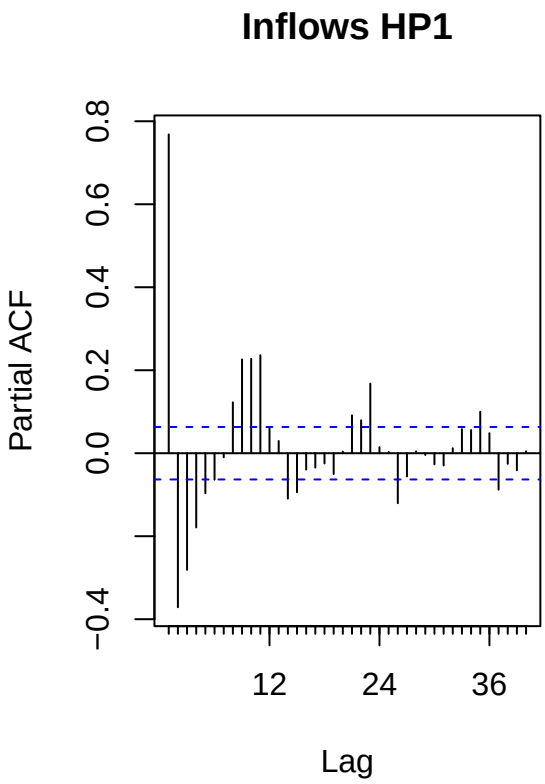
```
## Feb 1932 5535 4082 2262 1895 2996 1454 525 1219 268 2615  316 4681  531 20596
## Mar 1932 4310 3398 2065 1686 2392 1888 674 1332 304 2269  271 3329  501 21638
##           HP15
## Jan 1931 31270
## Feb 1931 43827
## Mar 1931 49884
## Apr 1931 43962
## May 1931 35156
## Jun 1931 25764
## Jul 1931 18109
## Aug 1931 13320
## Sep 1931  8225
## Oct 1931  8900
## Nov 1931 13766
## Dec 1931 20880
## Jan 1932 33160
## Feb 1932 39791
## Mar 1932 48274
```

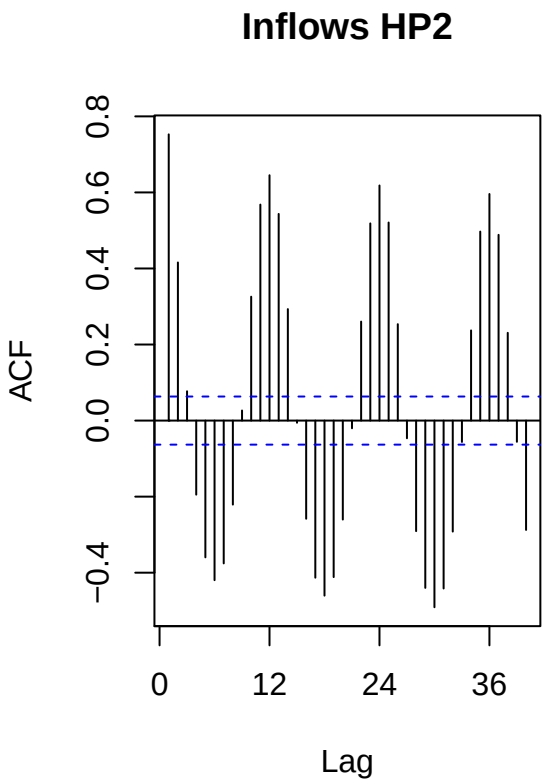
Plotting ACF and PACF

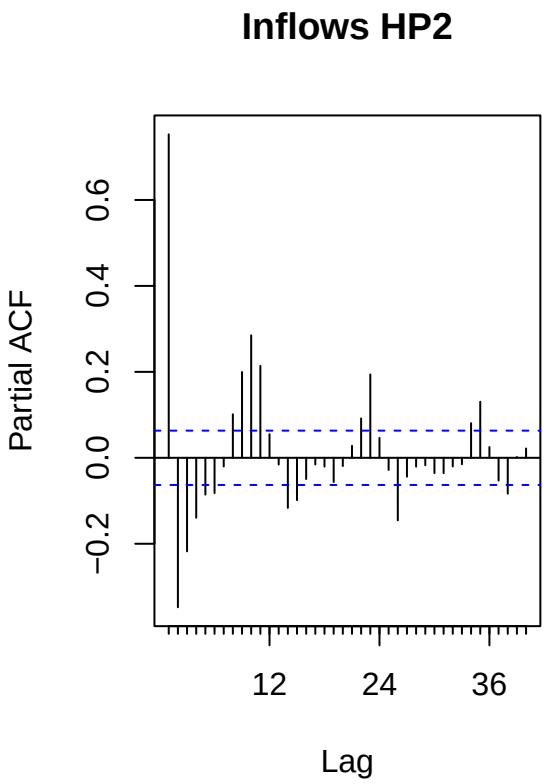
Let's use functions Acf() and Pacf() from package "forecast".

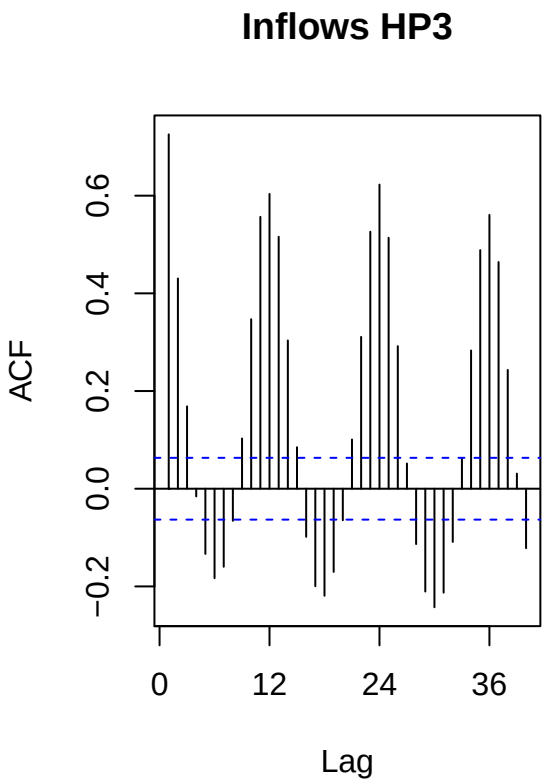
```
#Acf and Pacf for HP1
for(i in 1:nhydro){
  par(mfrow=c(1,2))  #place plot side by side
  Acf(ts_inflow_data[,i],lag.max=40,main=paste("Inflows HP",i,sep=""))
  # because I am not storing Acf() into any object, I don't need to specify plot=TRUE
  Pacf(ts_inflow_data[,i],lag.max=40,main=paste("Inflows HP",i,sep=""))
}
```

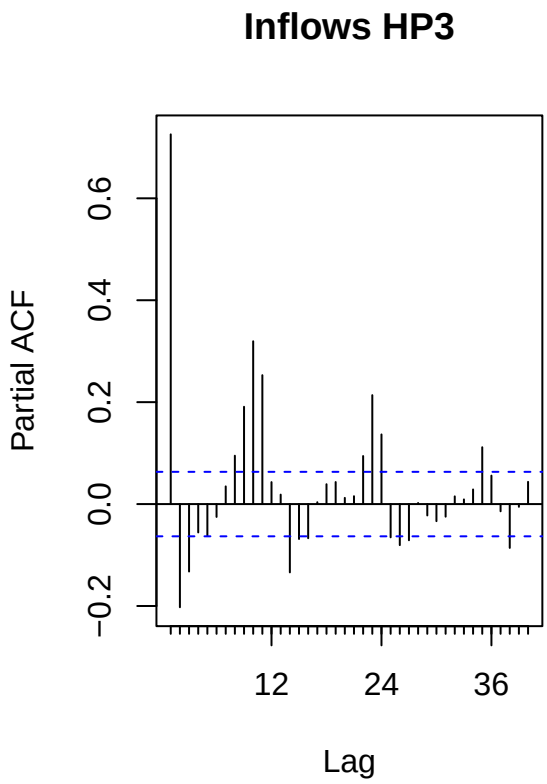

Inflows HP1
ACF
0.8
0.6
0.4
0.2
0.0
−0.4
0
12
24
36
Lag

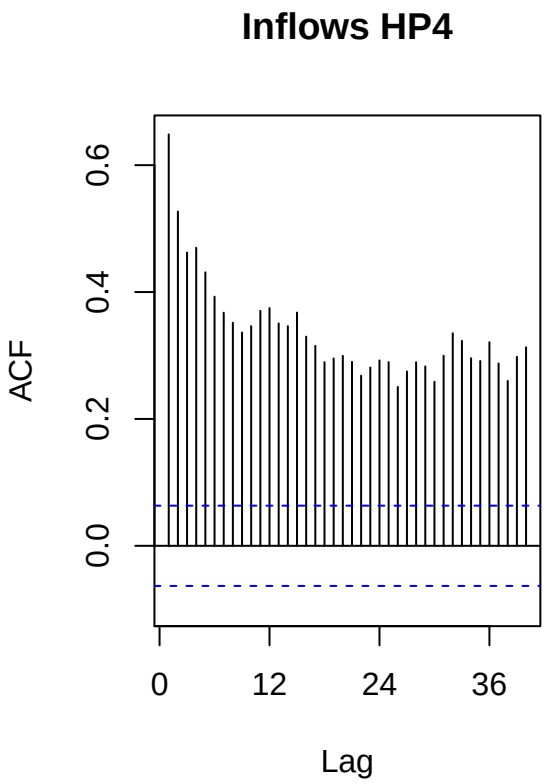

Inflows HP1
Partial ACF
0.8
0.6
0.4
0.2
0.0
−0.4
12
24
36
Lag

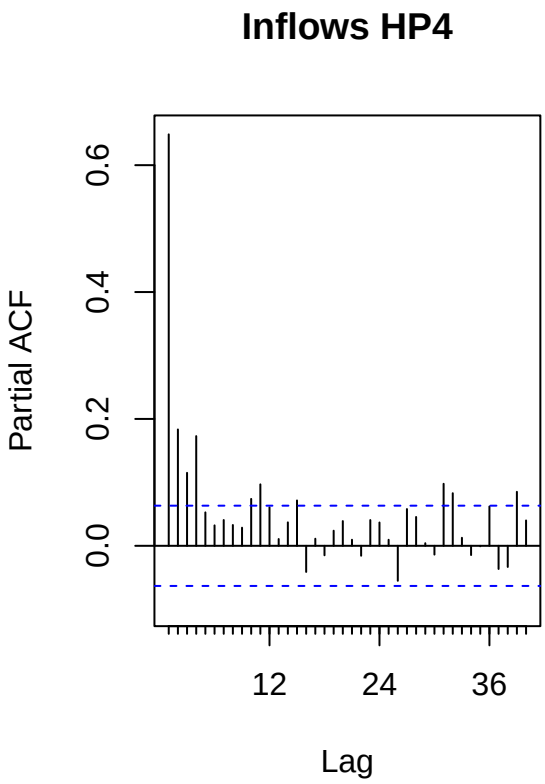

Inflows HP2
ACF
0.8
0.6
0.4
0.2
0.0
−0.4
0
12
24
36
Lag

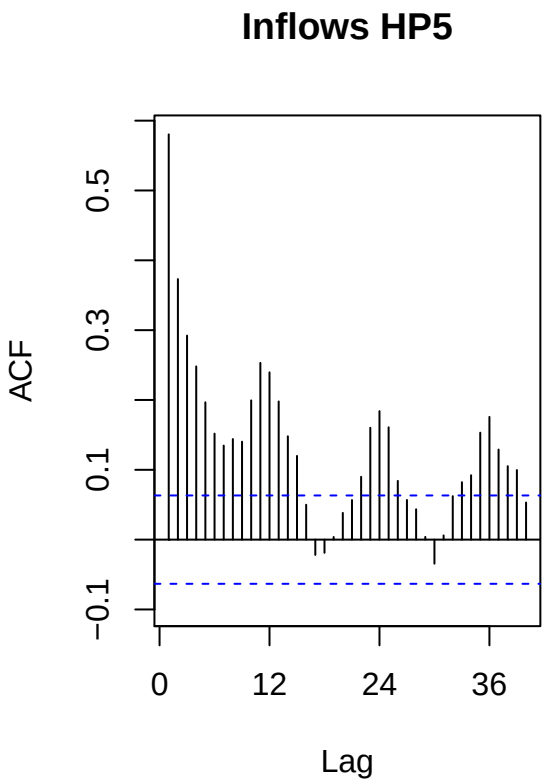

Inflows HP2
Partial ACF
0.6
0.4
0.2
0.0
−0.2
12
24
36
Lag

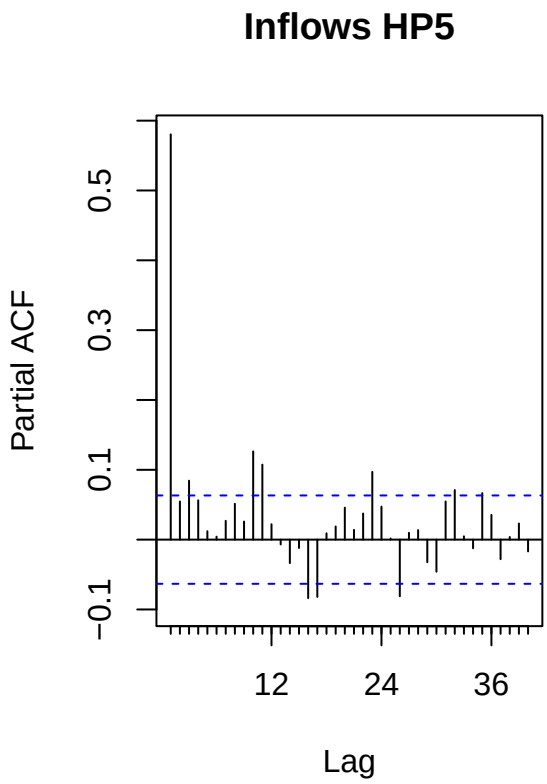

Inflows HP3
ACF
0.6
0.4
0.2
0.0
−0.2
0
12
24
36
Lag

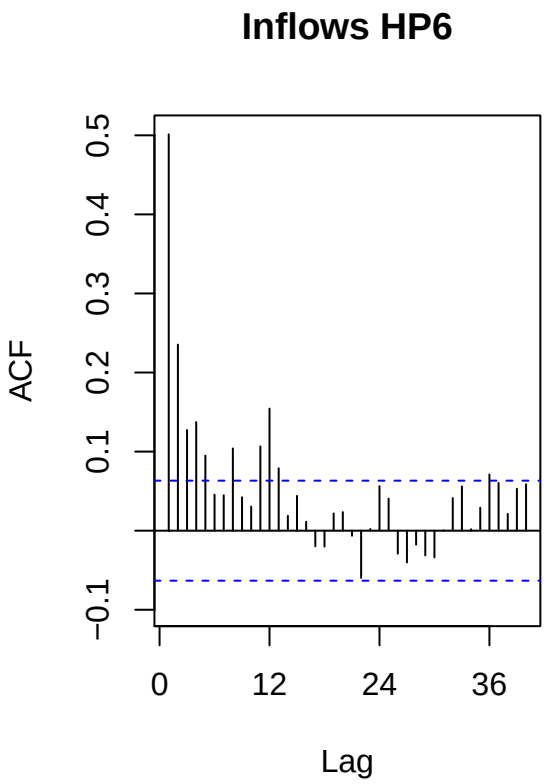

Inflows HP3
Partial ACF
0.6
0.4
0.2
0.0
−0.2
12
24
36
Lag

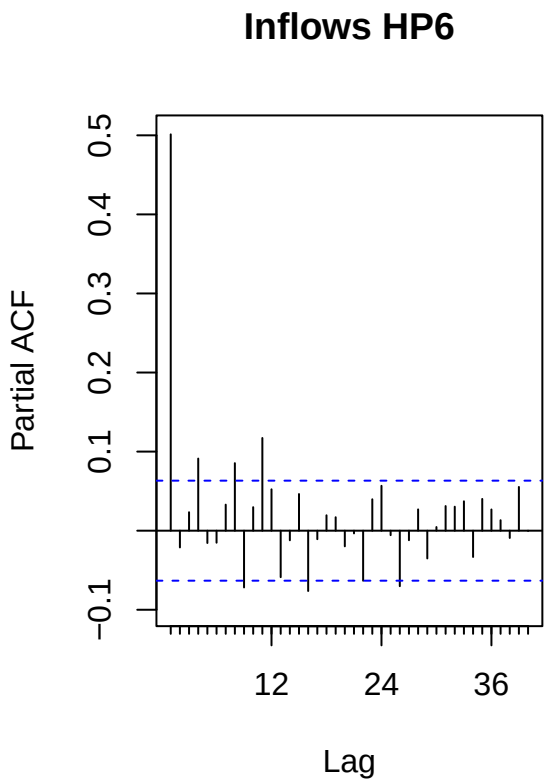

Inflows HP4
ACF
0.0
0.2
0.4
0.6
0
12
24
36
Lag

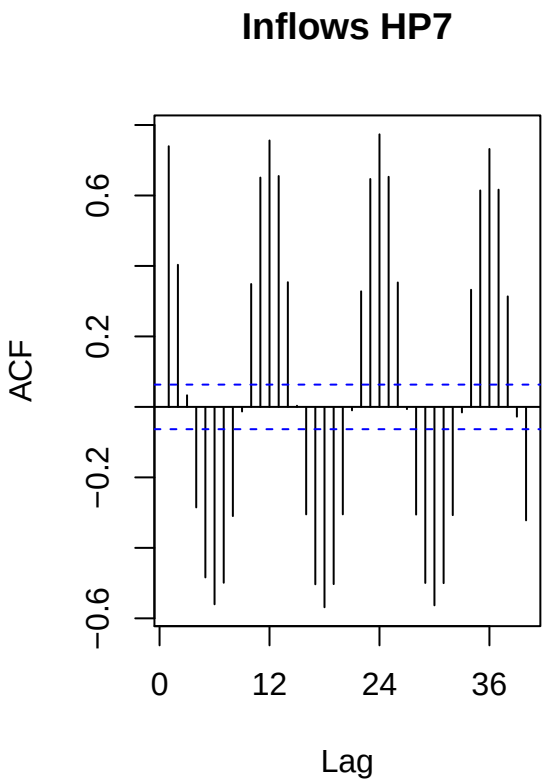

Inflows HP4
Partial ACF
0.0
0.2
0.4
0.6
12
24
36
Lag

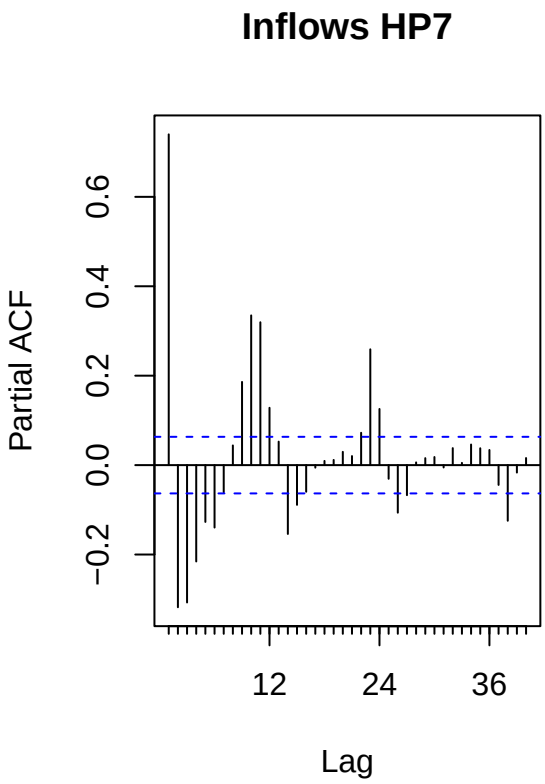

Inflows HP5
ACF
0.5
0.3
0.1
−0.1
0
12
24
36
Lag

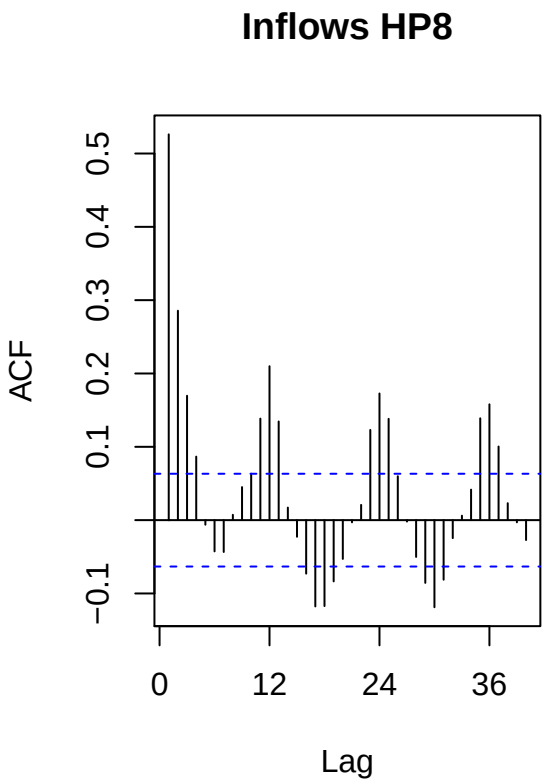

Inflows HP5
Partial ACF
0.5
0.3
0.1
−0.1
12
24
36
Lag

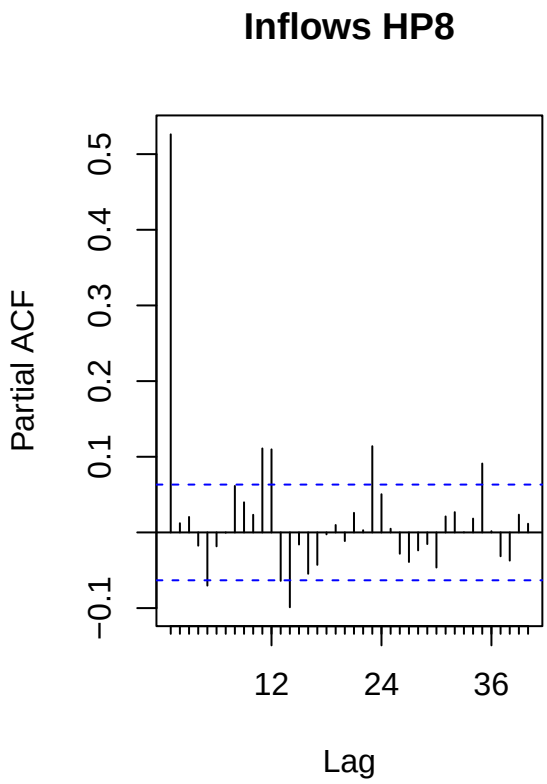

Inflows HP6
ACF
0.5
0.4
0.3
0.2
0.1
−0.1
0
12
24
36
Lag

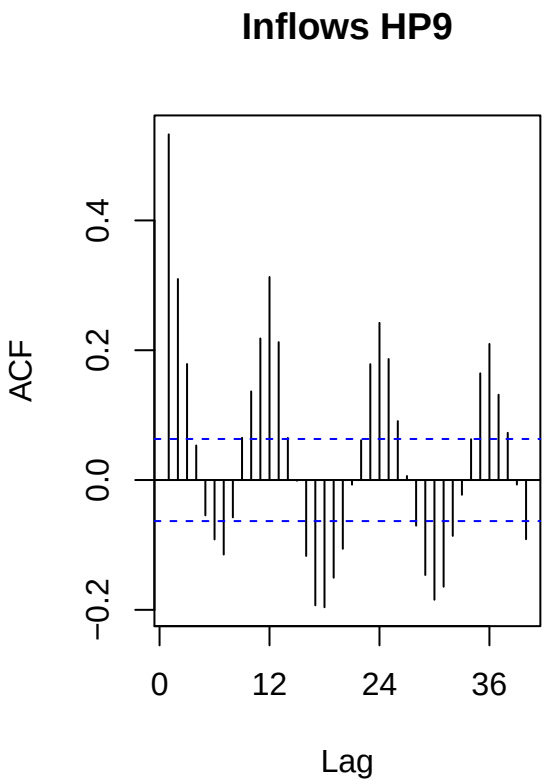

Inflows HP6
Partial ACF
0.5
0.4
0.3
0.2
0.1
−0.1
12
24
36
Lag

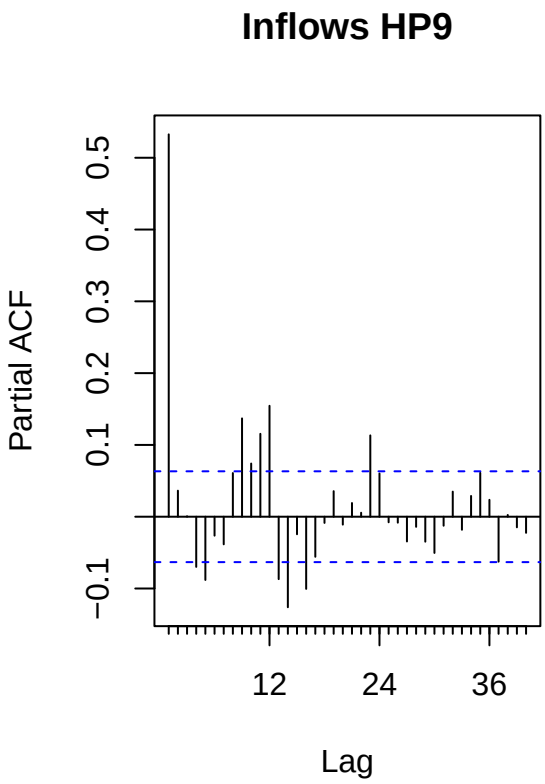

Inflows HP7
ACF
0.6
0.2
−0.2
−0.6
0
12
24
36
Lag

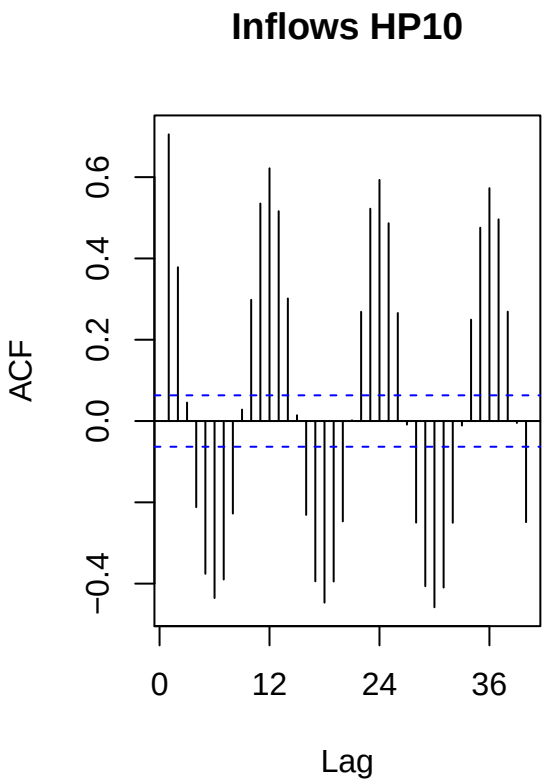

Inflows HP7
Partial ACF
0.6
0.4
0.2
0.0
−0.2
12
24
36
Lag

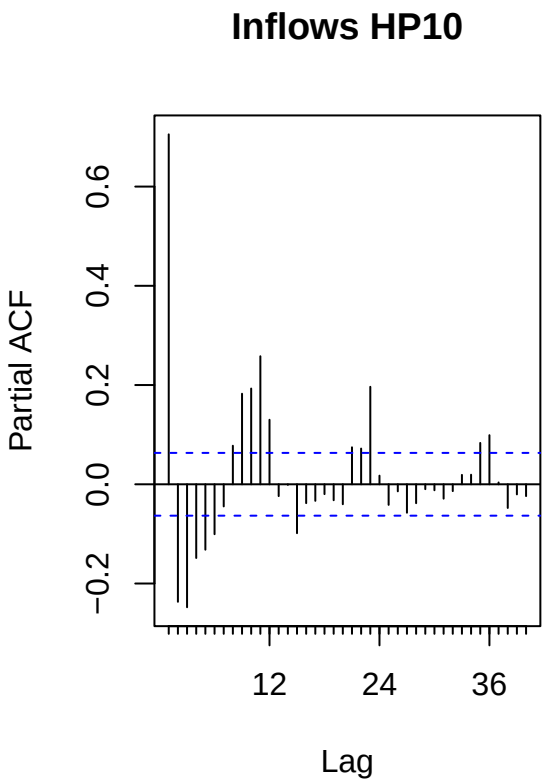

Inflows HP8
ACF
0.5
0.4
0.3
0.2
0.1
−0.1
0
12
24
36
Lag

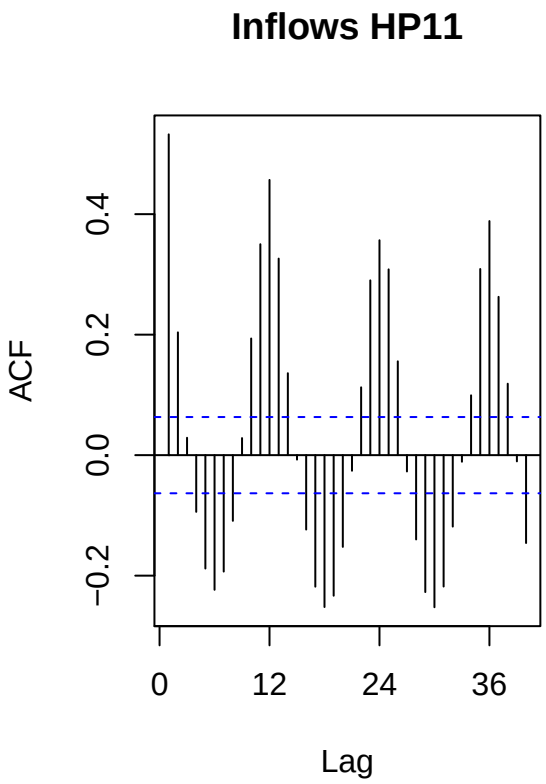

Inflows HP8
Partial ACF
0.5
0.4
0.3
0.2
0.1
−0.1
12
24
36
Lag

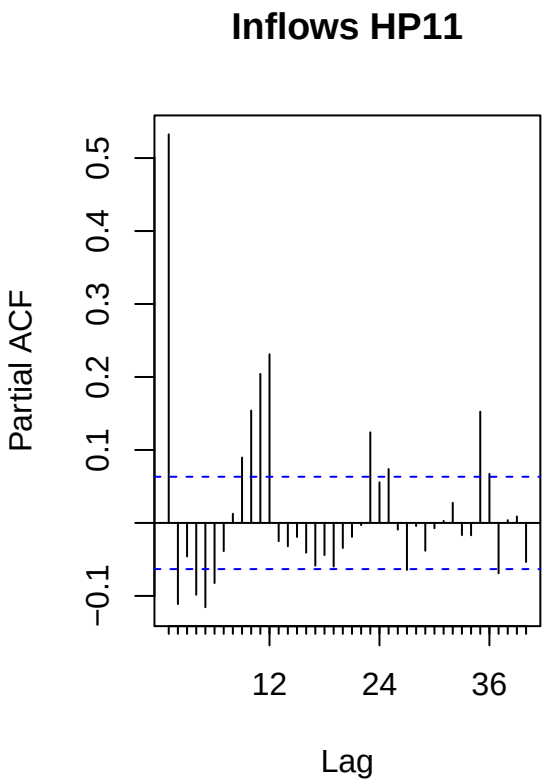

Inflows HP9
ACF
0.4
0.2
0.0
−0.2
0
12
24
36
Lag

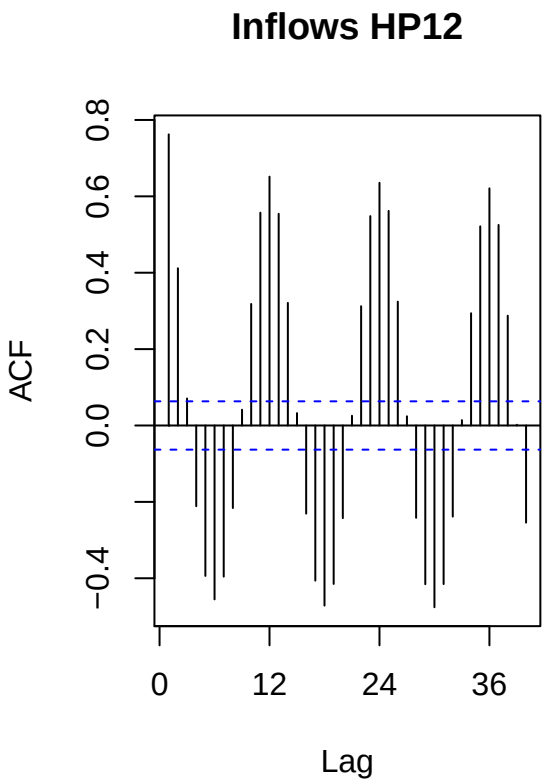

Inflows HP9
Partial ACF
0.5
0.4
0.3
0.2
0.1
−0.1
12
24
36
Lag

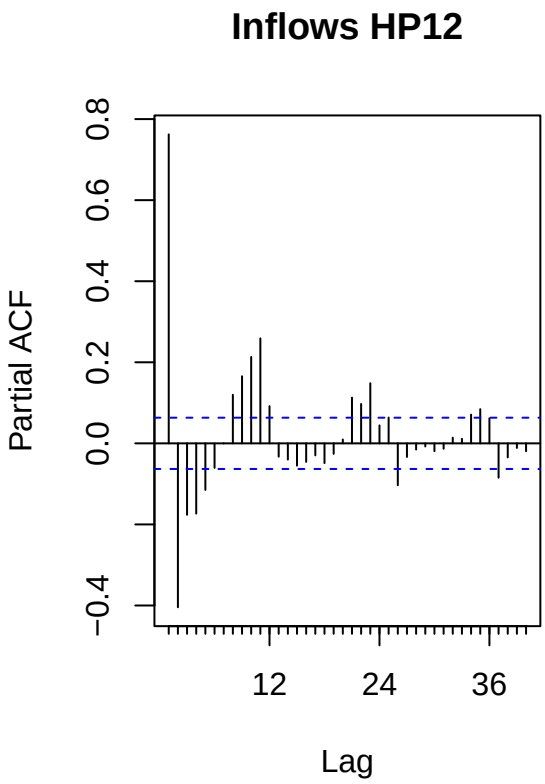

Inflows HP10
ACF
0.6
0.4
0.2
0.0
−0.4
0
12
24
36
Lag

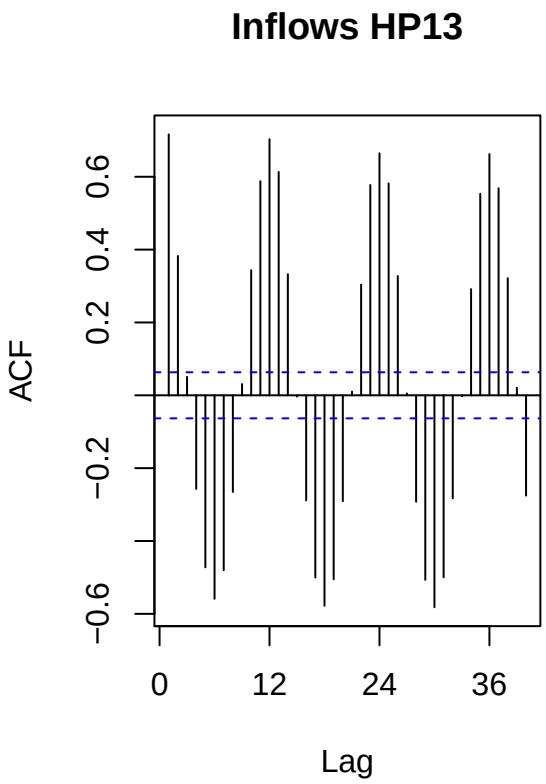

Inflows HP10
Partial ACF
0.6
0.4
0.2
0.0
−0.2
12
24
36
Lag

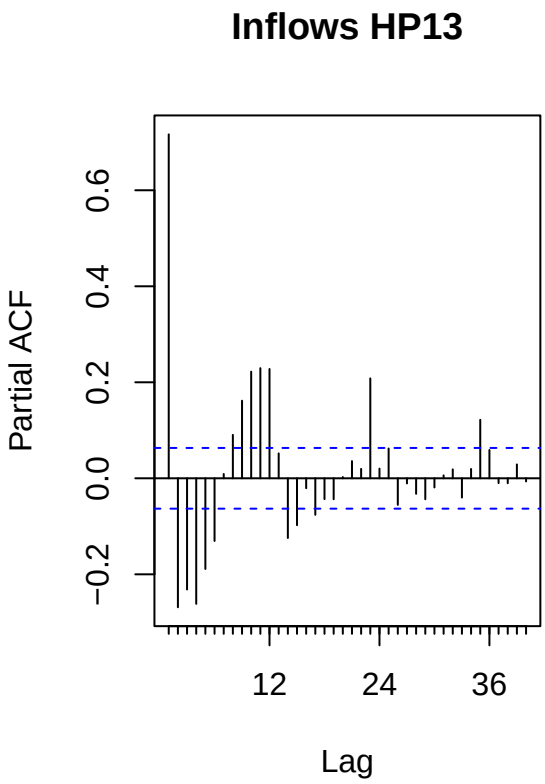

Inflows HP11
ACF
0.4
0.2
0.0
−0.2
0
12
24
36
Lag

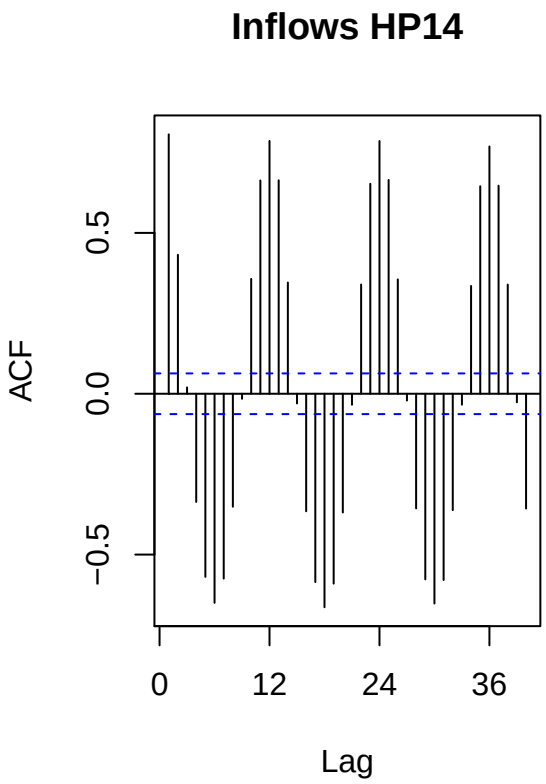

Inflows HP11
Partial ACF
0.5
0.4
0.3
0.2
0.1
−0.1
12
24
36
Lag

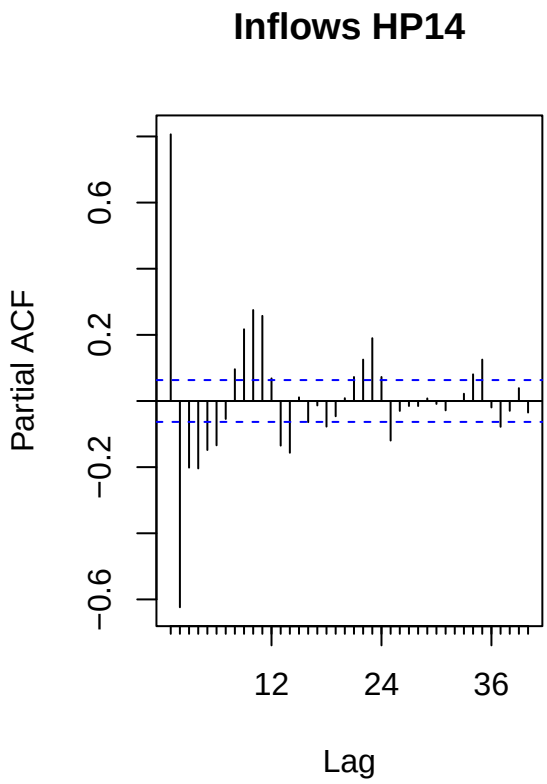

Inflows HP12
ACF
0.8
0.6
0.4
0.2
0.0
−0.4
0
12
24
36
Lag

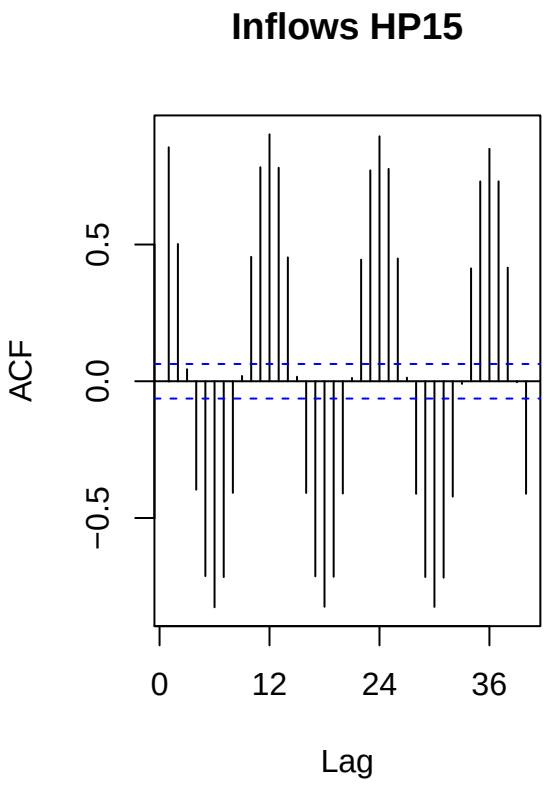

Inflows HP12
Partial ACF
0.8
0.6
0.4
0.2
0.0
−0.4
12
24
36
Lag

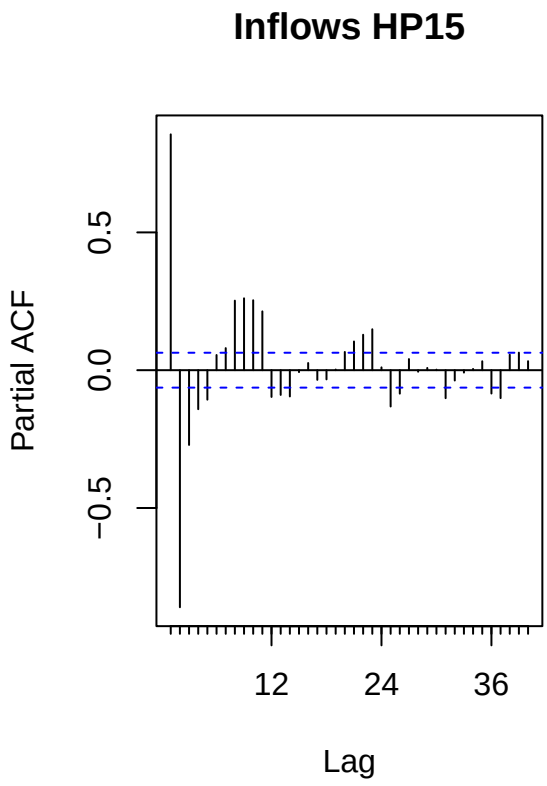

Inflows HP13
ACF
0.6
0.4
0.2
−0.2
−0.6
0
12
24
36
Lag


Inflows HP13
Partial ACF
0.6
0.4
0.2
0.0
−0.2
12
24
36
Lag


Inflows HP14
ACF
0.5
0.0
−0.5
0
12
24
36
Lag


Inflows HP14
Partial ACF
0.6
0.2
−0.2
−0.6
12
24
36
Lag





Trend Component - Linear Model

Let's identify and remove trend component. You start by fitting a linear model to $Y_t = \beta_0 + \beta_1 * t + \epsilon_t$.

```
iHP <- 4
#Create vector t
t <- c(1:nobs)

#Fit a linear trend to TS of iHP
linear_trend_model <- lm(inflow_data[,iHP+1] ~ t) # + 1 refers to date column
summary(linear_trend_model)
```

```
##
## Call:
## lm(formula = inflow_data[, iHP + 1] ~ t)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2057.2  -691.1  -217.4   503.6  5786.1
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 637.5519    65.4250   9.745   <2e-16 ***
## t             2.1836     0.1179  18.513   <2e-16 ***
## ---
```

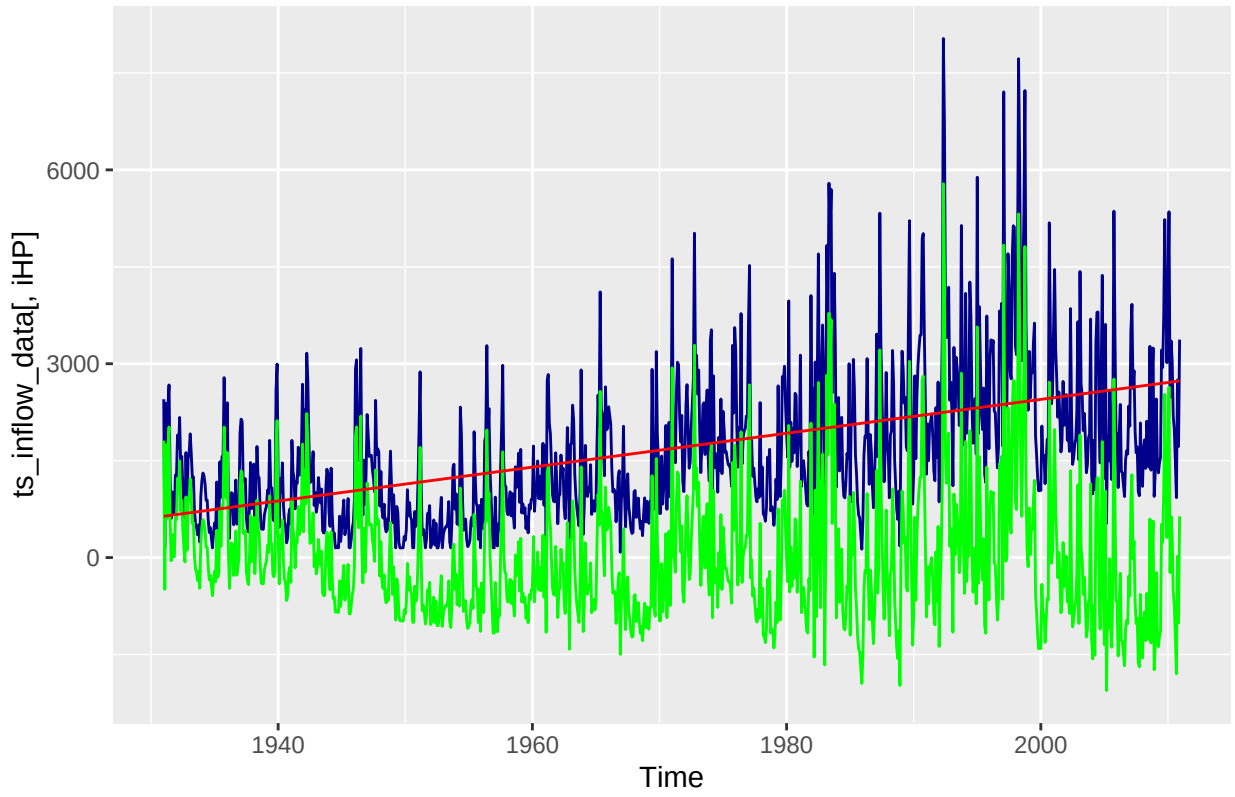
```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1013 on 958 degrees of freedom
## Multiple R-squared:  0.2635, Adjusted R-squared:  0.2627
## F-statistic: 342.7 on 1 and 958 DF,  p-value: < 2.2e-16
```

```
beta0 <- as.numeric(linear_trend_model$coefficients[1])
beta1 <- as.numeric(linear_trend_model$coefficients[2])

#detrend inflow
linear_trend <- beta0 + beta1 * t
ts_linear <- ts(linear_trend,star=c(1931,1),frequency=12)

detrend_inflow <- inflow_data[,iHP+1] - linear_trend
ts_detrend <- ts(detrend_inflow, start = c(1931,1),frequency = 12)

#Plot
autoplot(ts_inflow_data[,iHP],color="darkblue")+
  autolayer(ts_detrend,series="Detrended",color="green")+
  autolayer(ts_linear,series="Linear Component",color="red")
```



Note that blue line is our original series, red line is our trend, green line is our original series minus the trend or in other words the detrended series. And in orange is the trend line for the detrended series which has slope 0 meaning we were able to effectively eliminate the trend with a linear model.

Seasonal Component

Now let's shift attention to the seasonal component.

```
#Use seasonal means model
dummies <- seasonaldummy(ts_detrend)

seas_means_model <- lm(detrend_inflow ~ dummies)
summary(seas_means_model)
```

```
##
## Call:
## lm(formula = detrend_inflow ~ dummies)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2121.6  -676.4  -185.8   486.7  5592.9
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -111.82     111.95  -0.999   0.3181
## dummiesJan    -23.26     158.32  -0.147   0.8833
## dummiesFeb    331.62     158.32   2.095   0.0365 *
## dummiesMar    176.23     158.32   1.113   0.2660
## dummiesApr    138.81     158.32   0.877   0.3809
## dummiesMay    305.05     158.32   1.927   0.0543 .
## dummiesJun    294.89     158.32   1.863   0.0628 .
## dummiesJul     42.58     158.32   0.269   0.7880
## dummiesAug   -247.80     158.32  -1.565   0.1179
## dummiesSep   -131.19     158.32  -0.829   0.4075
## dummiesOct    342.50     158.32   2.163   0.0308 *
## dummiesNov    112.42     158.32   0.710   0.4778
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1001 on 948 degrees of freedom
## Multiple R-squared:  0.03266,    Adjusted R-squared:  0.02144
## F-statistic:  2.91 on 11 and 948 DF,  p-value: 0.0008744
```

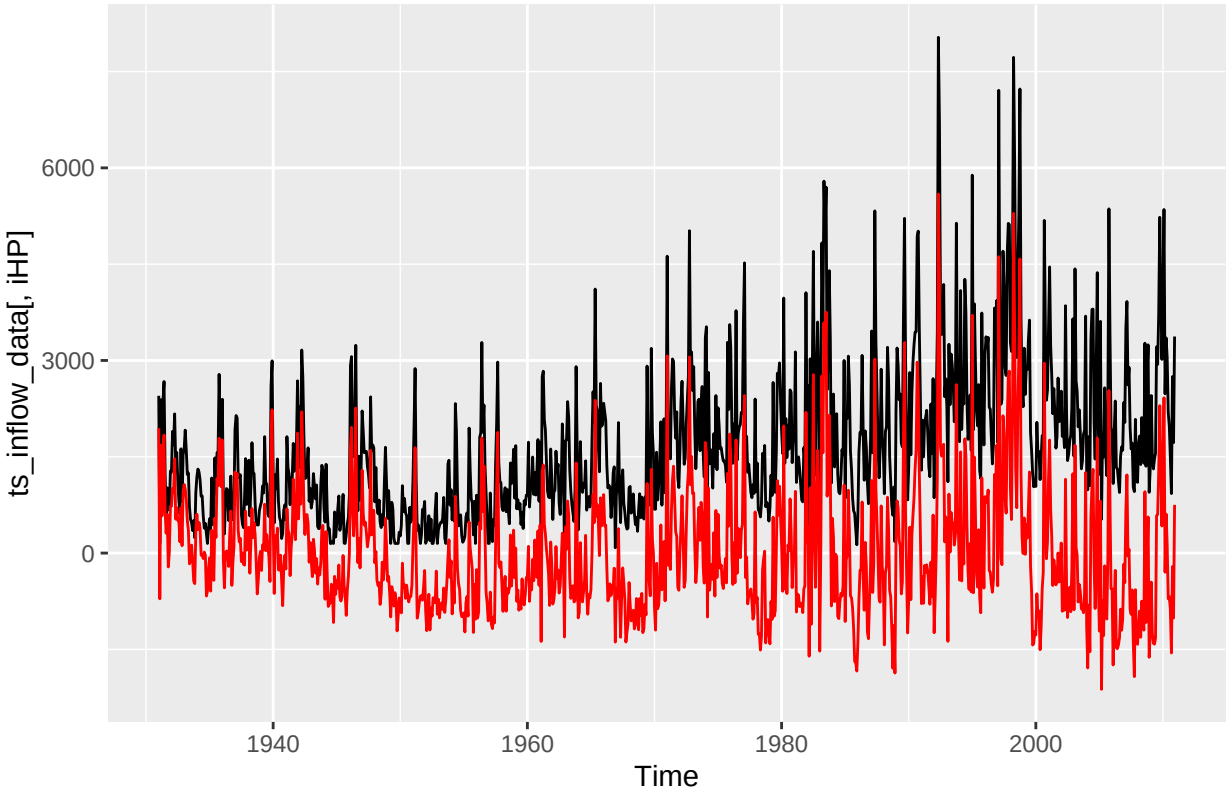
```
beta_intercept <-seas_means_model$coefficients[1]
beta_coeff <-seas_means_model$coefficients[2:12]

inflow_seas_comp <- array(0,nobs)
for(i in 1:nobs){
  inflow_seas_comp[i] <- beta_intercept + beta_coeff %*% dummies[i,]
}

deseason_inflow_data <- detrend_inflow - inflow_seas_comp

ts_deseason_inflow_data <- ts(deseason_inflow_data,start=c(1931,1),
                              frequency = 12)

autoplot(ts_inflow_data[,iHP])+
  autolayer(ts_deseason_inflow_data,color="red")
```



##Exercise

Fit trend and seasonal for the other variables HP2, HP3, ...

Stationarity Tests in R

Some test only work for non-seasonal data. So let's create another series with yearly averages for inflow.

```
#Group data in yearly steps instances
```

Mann Kendall

Check for deterministic trend.

```
#Since I have seasonal data I cannot use the simple MannKendall()
#another example of functions that need a ts object


#Use yearly date to run Mann Kendall
```

Spearman Correlation test

Also check for deterministic trend, for non-seasonal data only.

```r
#Deterministic trend with Spearman Correlation Test
print("Results from Spearman Correlation")
```

```
## [1] "Results from Spearman Correlation"
```

```r
#with cor.test you can get test statistics
```

Augmented Dickey Fuller (ADF)

Used to detect stochastic trend.

```r
#Null hypothesis is that data has a unit root
print("Results for ADF test/n")
```

```
## [1] "Results for ADF test/n"
```

```r
#Now let's try the yearly data
print("Results for ADF test on yearly data/n")
```

```
## [1] "Results for ADF test on yearly data/n"
```

##Exercise

Run the stationarity tests for the other variables HP2, HP3, ...