

Trend and Seasonality

Luana Lima

01/26/2022

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)
```

```
## Warning: package 'forecast' was built under R version 4.1.2
```

```
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="/Users/kristenpulley/Desktop/Time Series/ENV790_TimeSeriesAnalysis_1.txt")

#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"
```

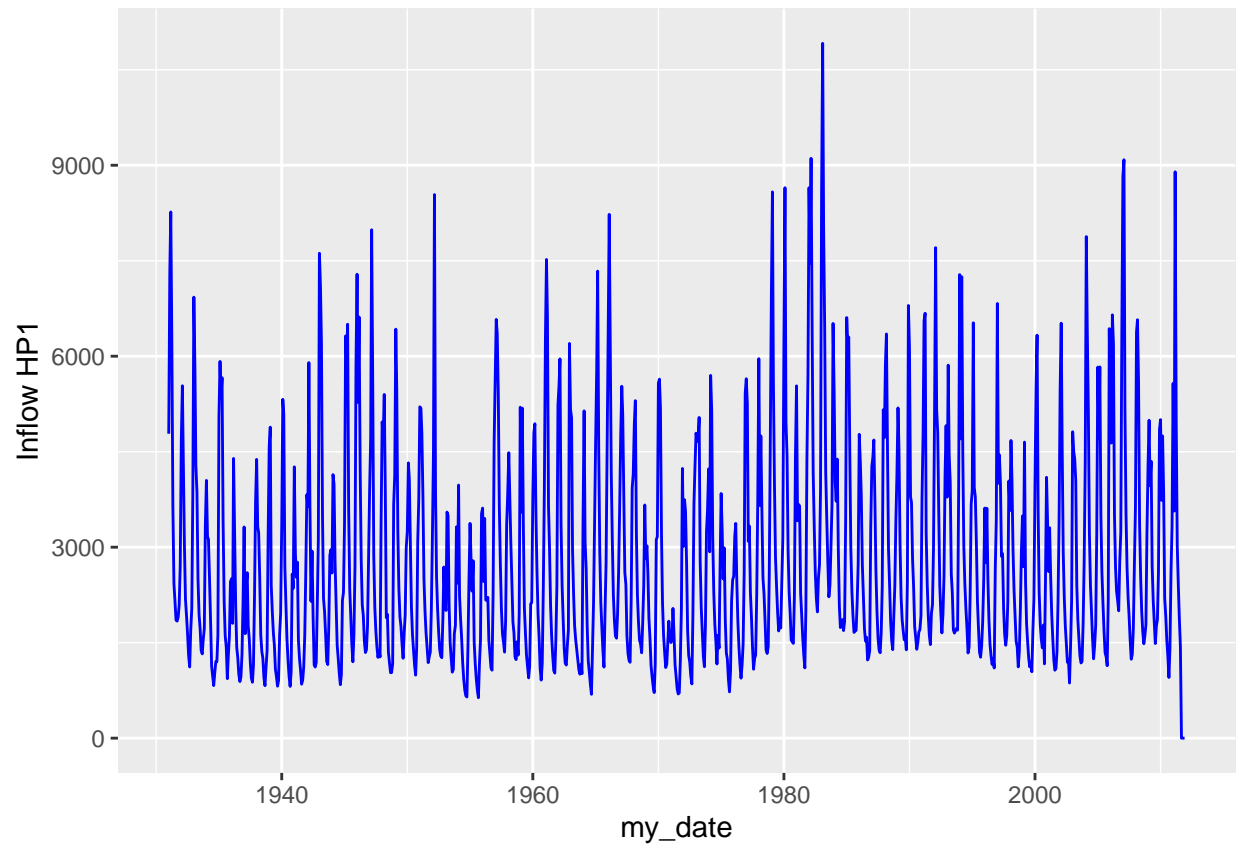
```
#add that to inflow_data and store in a new data frame
inflow_data <- cbind(my_date,raw_inflow_data[,3:(3+nhydro-1)])
head(inflow_data)
```

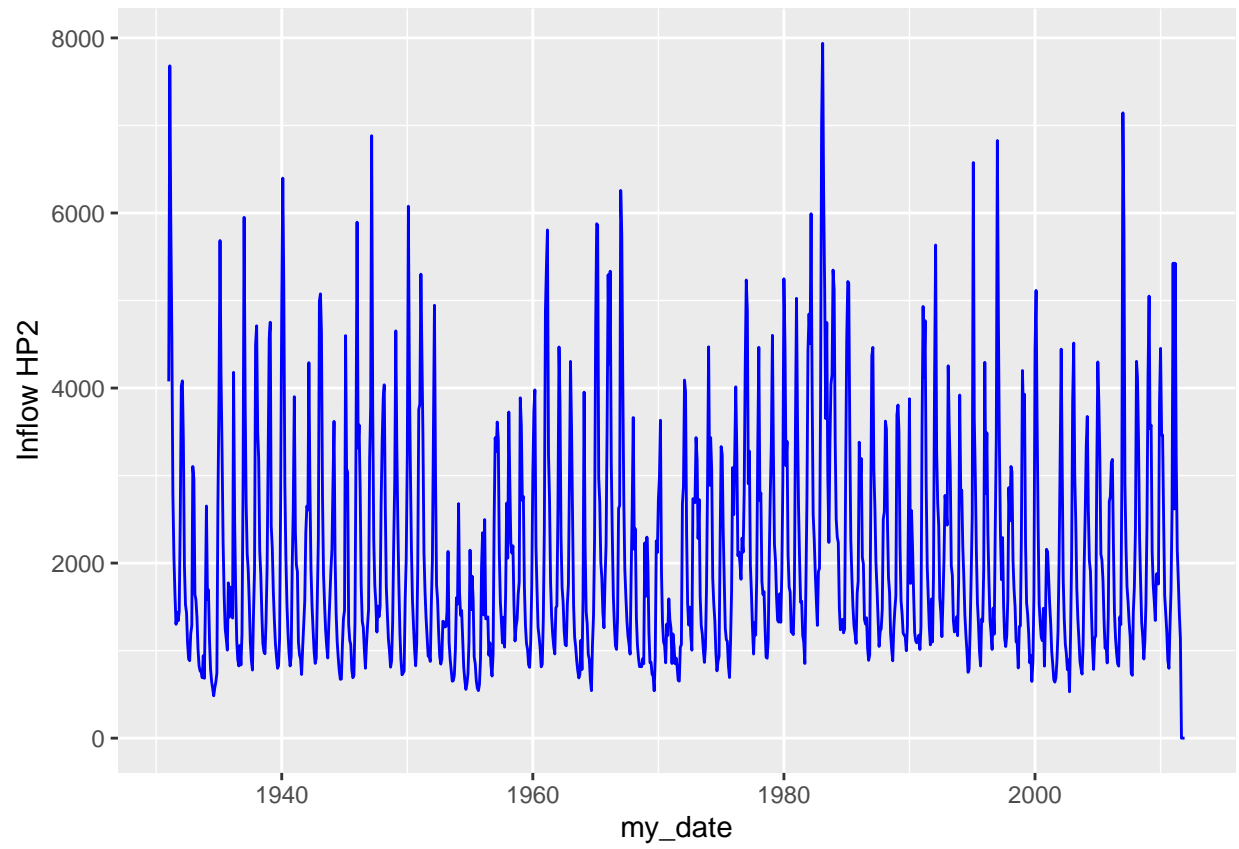
```
##      my_date HP1 HP2 HP3 HP4 HP5 HP6 HP7 HP8 HP9 HP10 HP11 HP12 HP13
## 1 1931-01-01 4782 4076 2518 2450 2649 1462 450 968 246 2636 452 4870 452
## 2 1931-02-01 7323 7681 4188 150 2401 758 554 219 74 4158 457 4550 796
## 3 1931-03-01 8266 5921 3253 2389 3261 707 615 333 123 3847 631 6537 804
##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

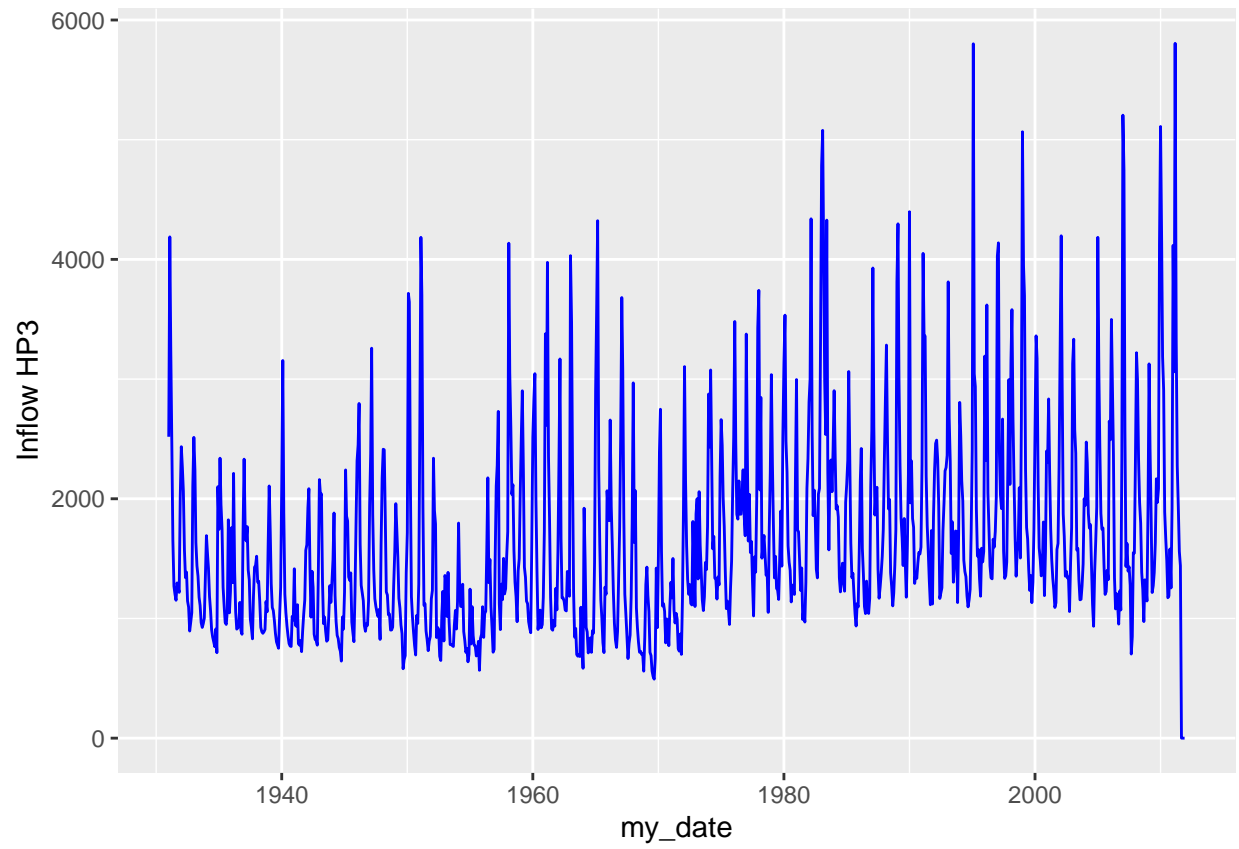
Initial Plots

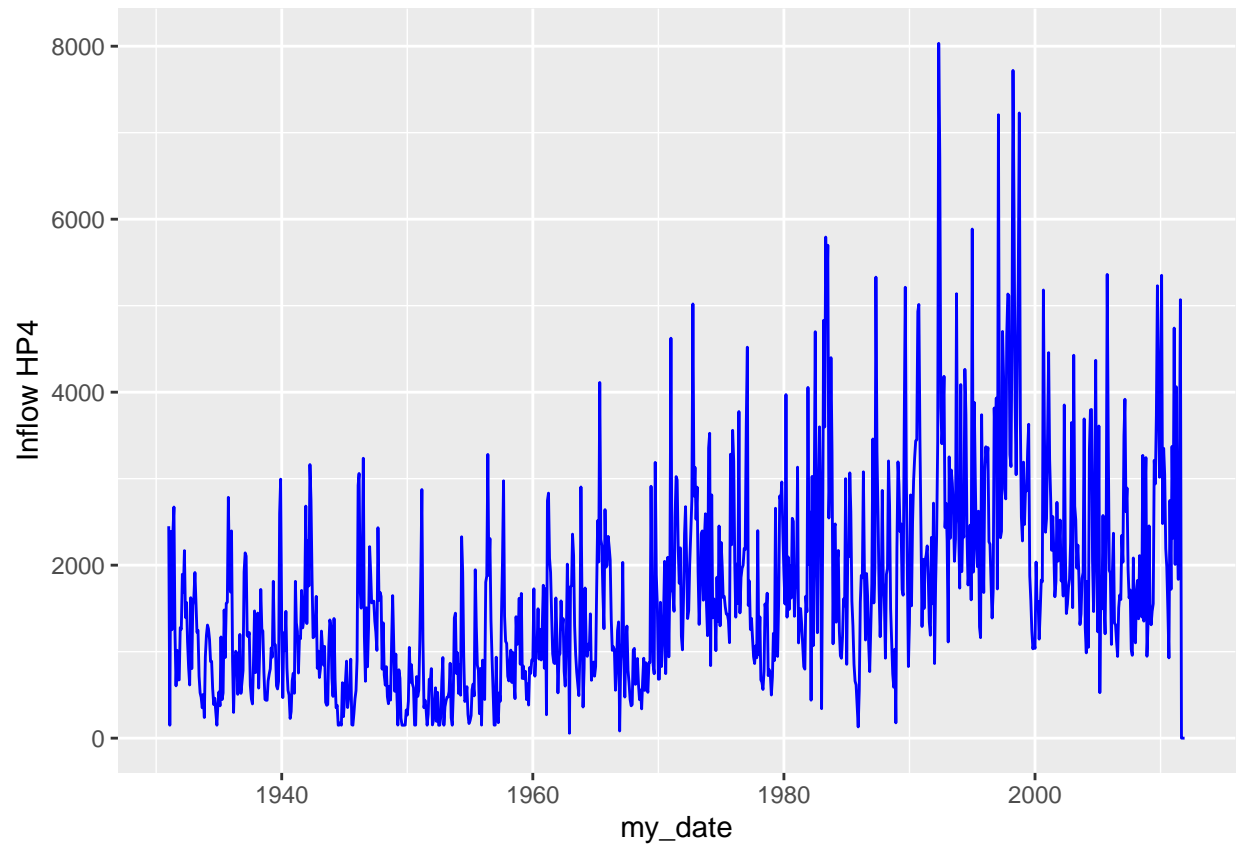
Initial time series plot.

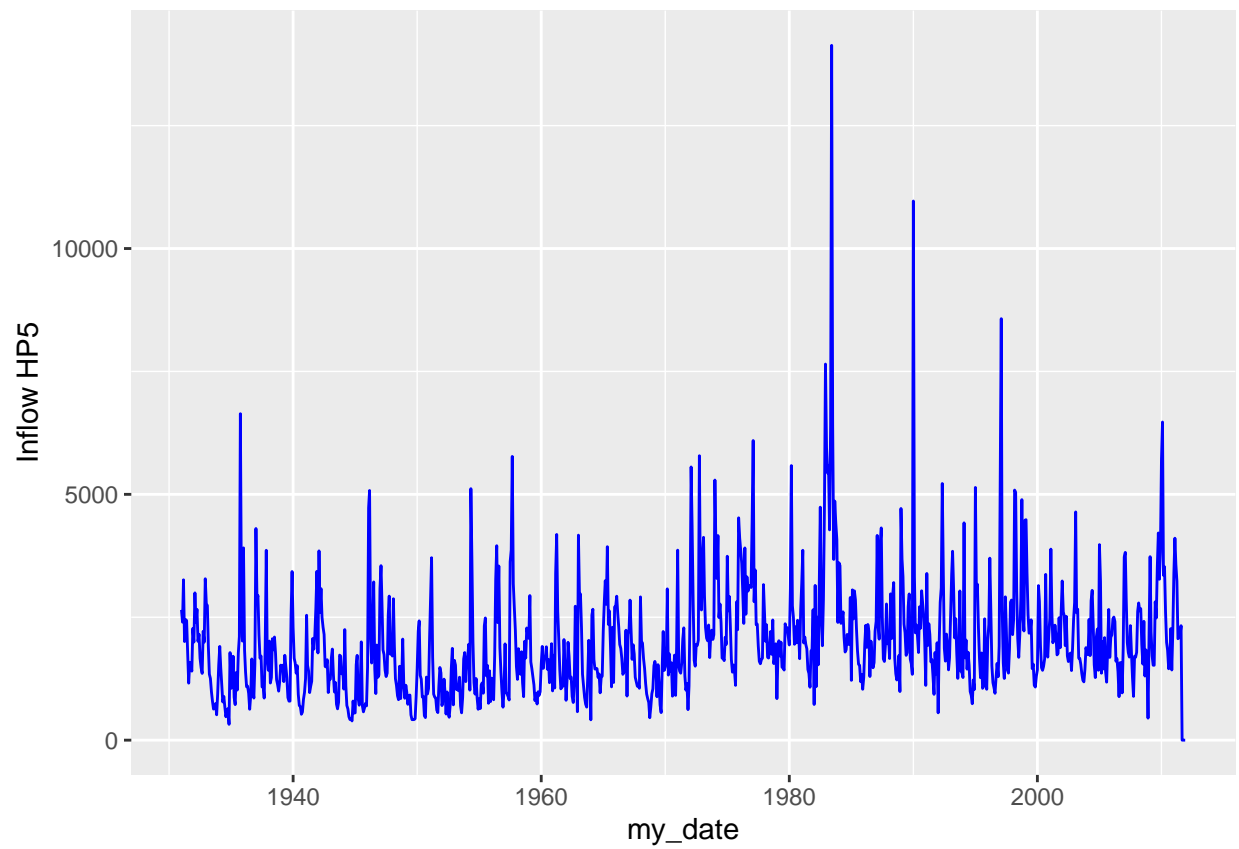
```
#using package ggplot2
for(i in 1:nhydro){
  print(ggplot(inflow_data, aes(x=my_date, y=inflow_data[, (1+i)])) +
    geom_line(color="blue") +
    ylab(paste0("Inflow ", colnames(inflow_data)[(1+i)], sep=""))
  )
}
```

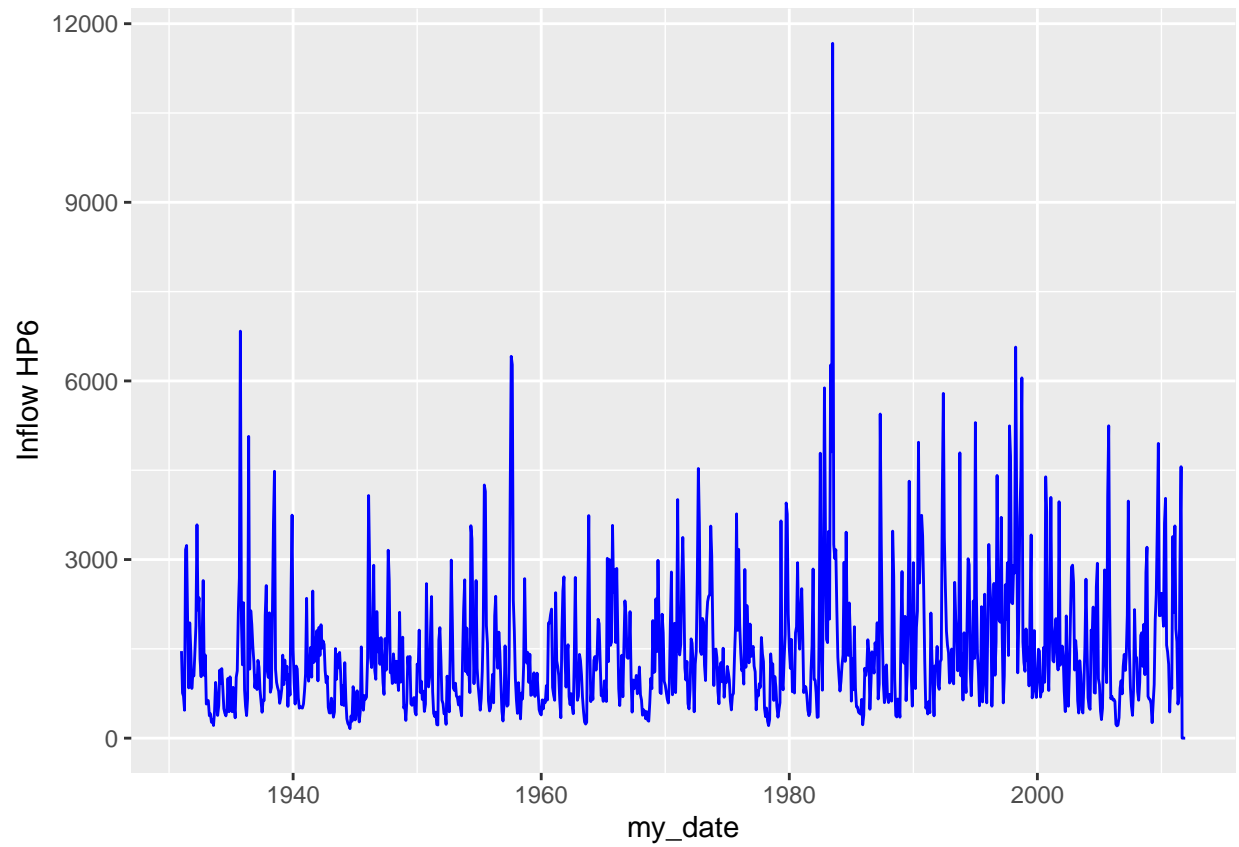


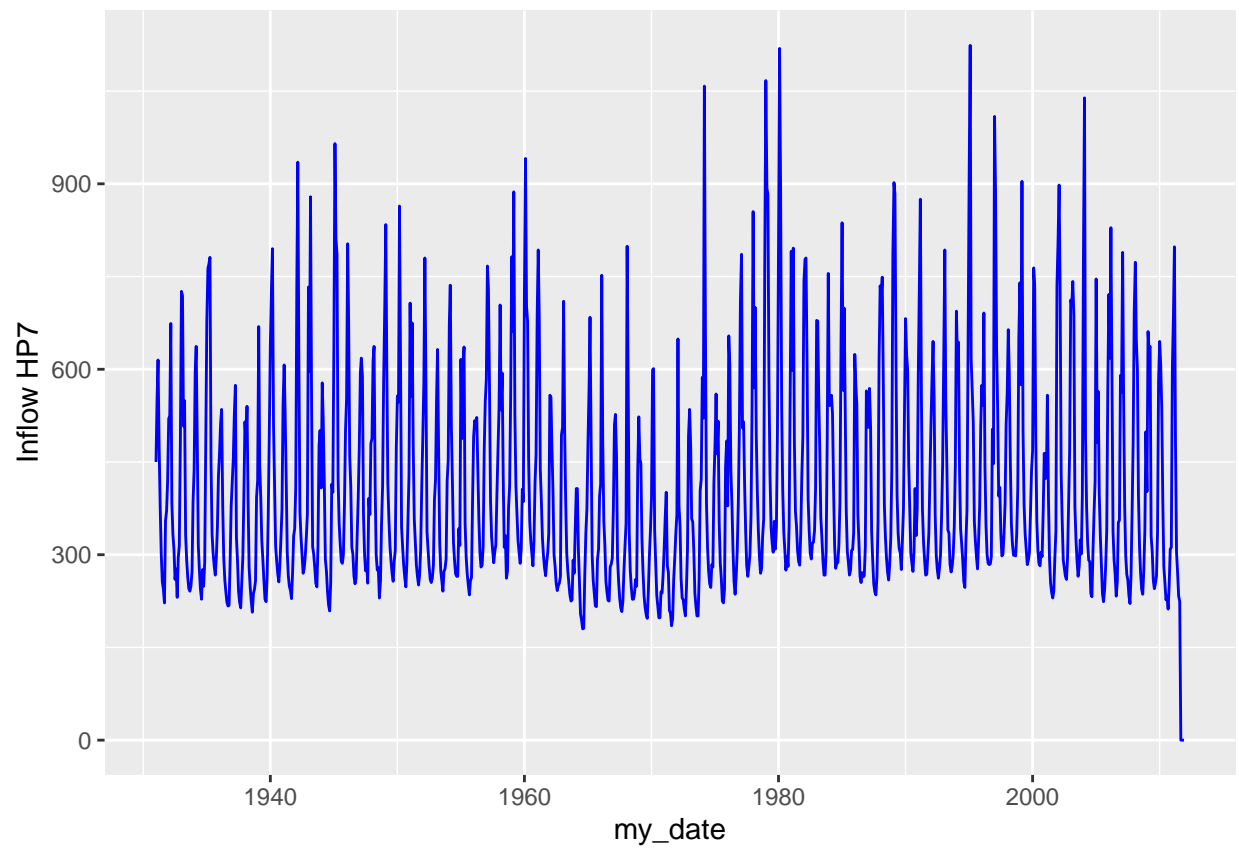


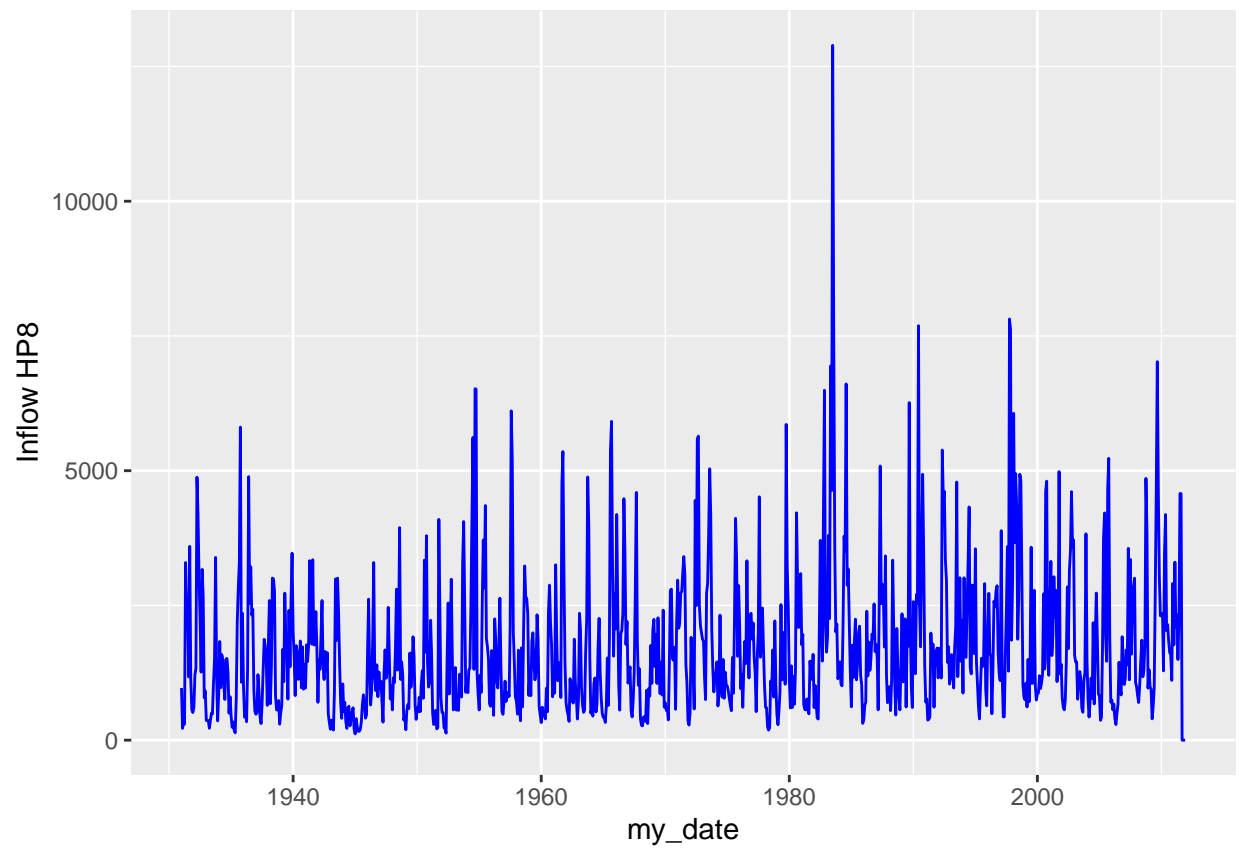


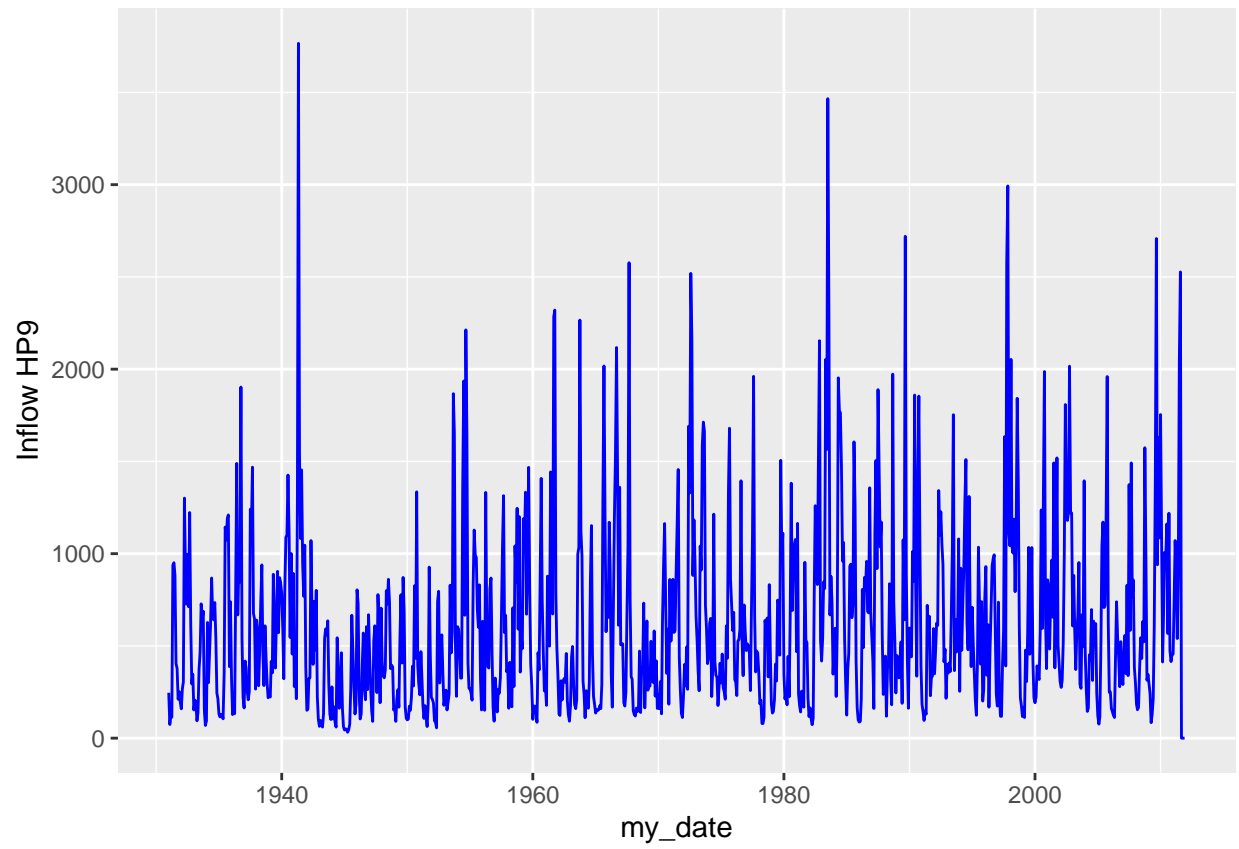


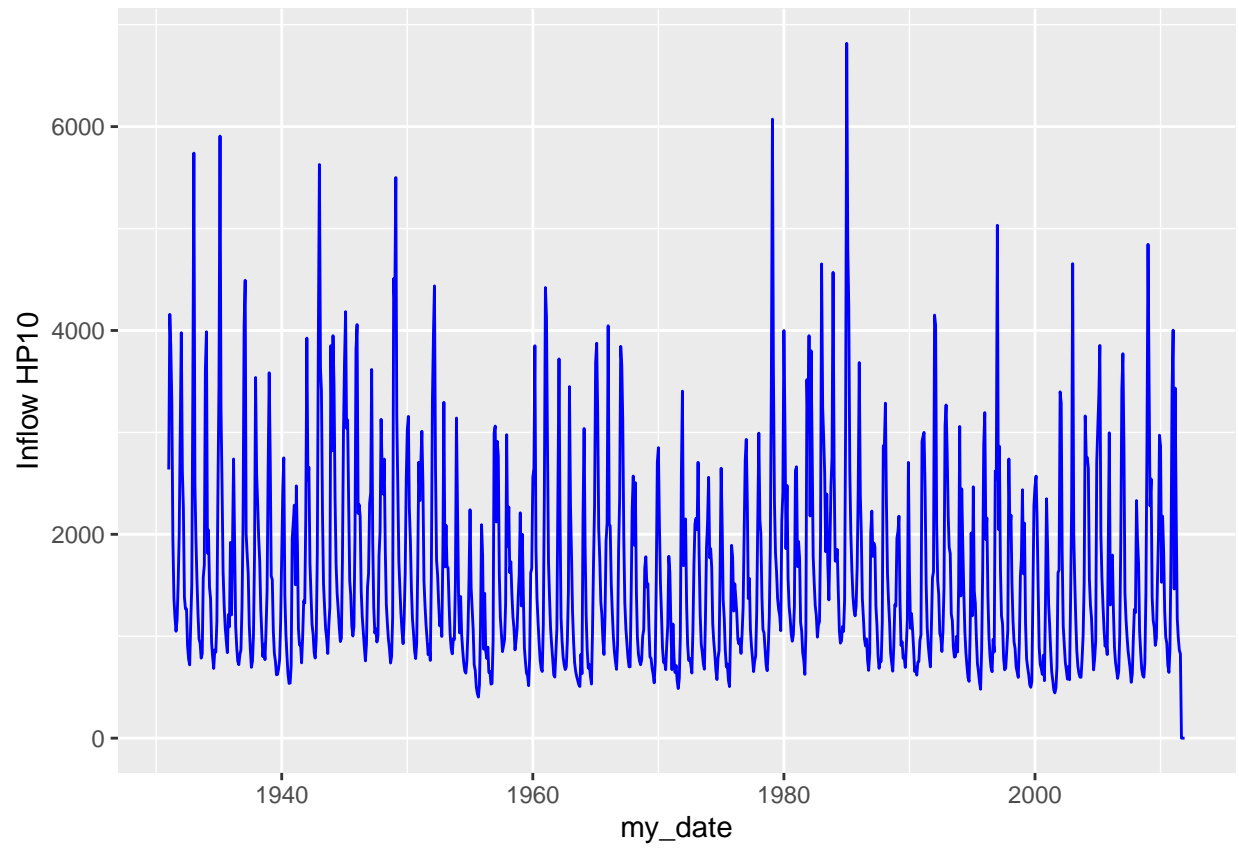


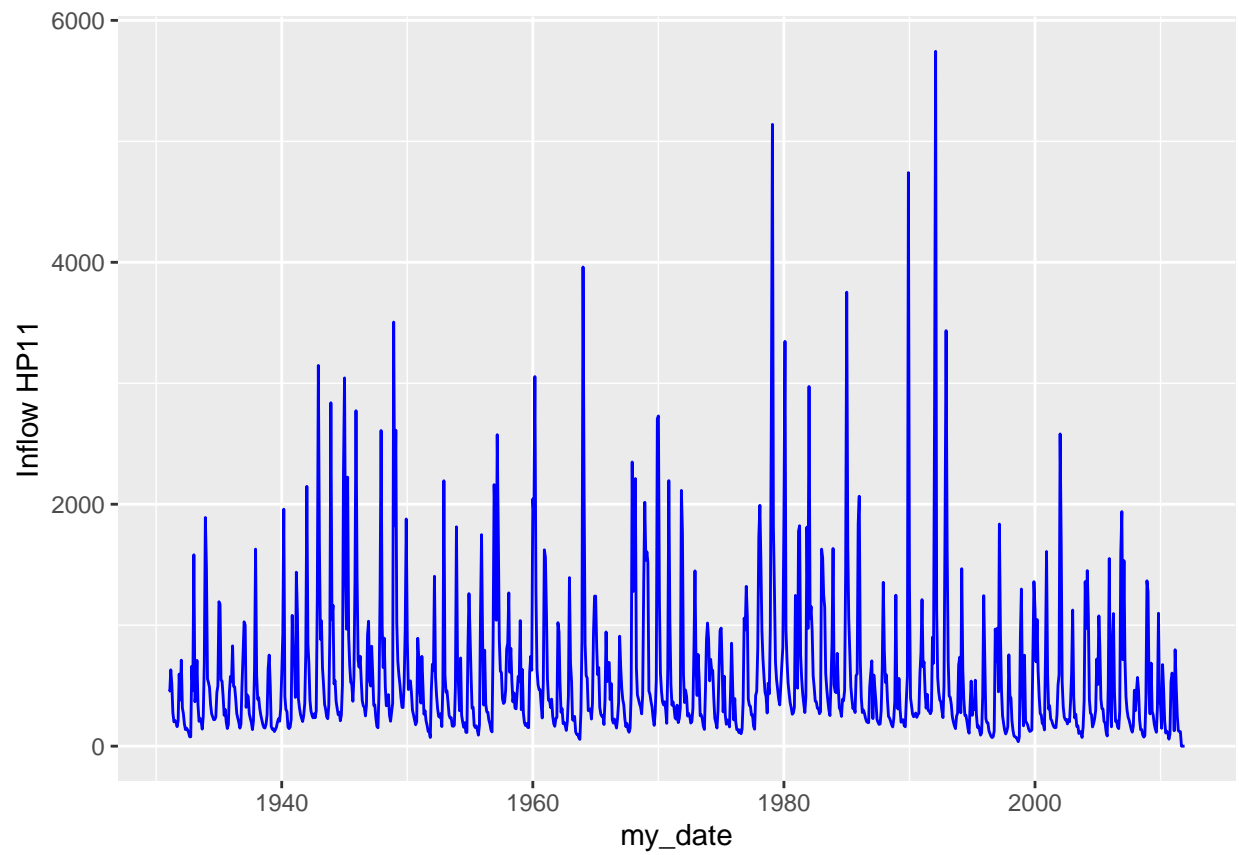


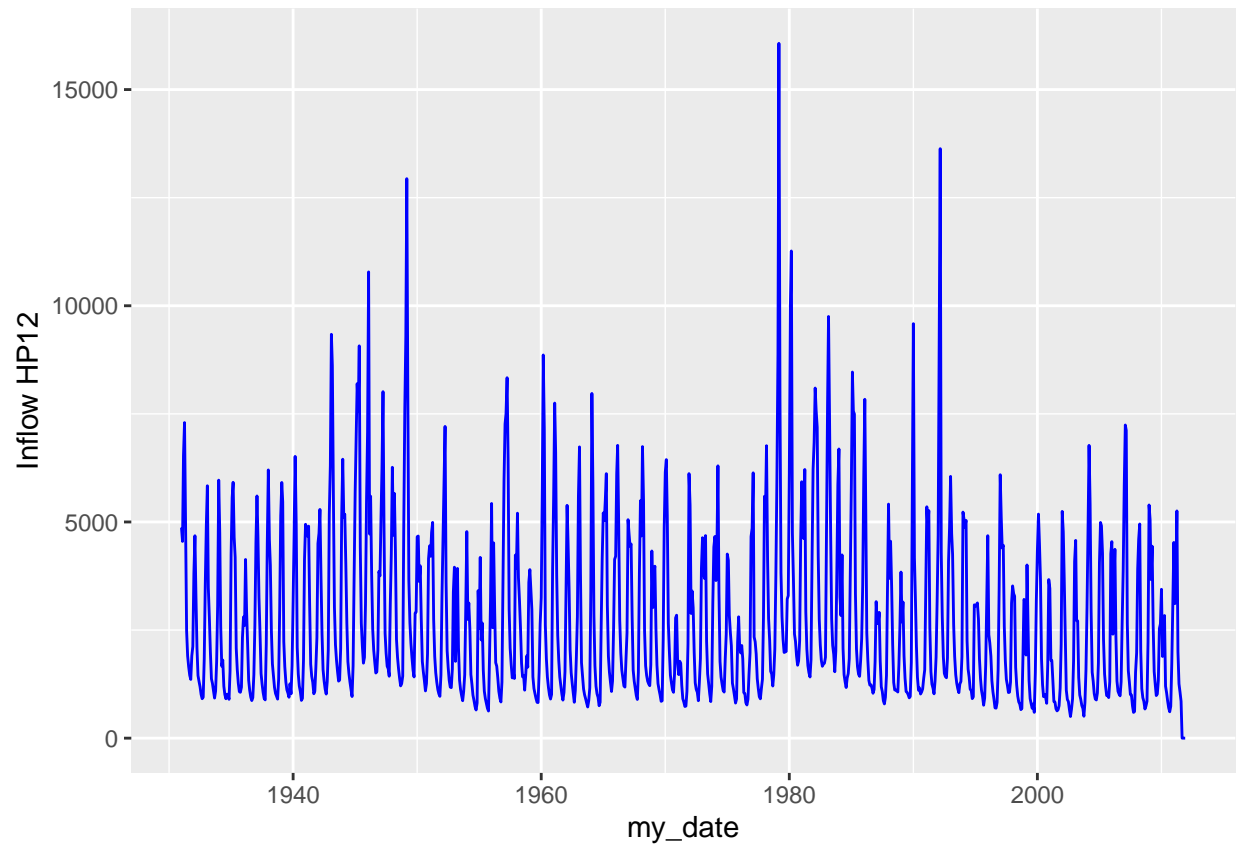


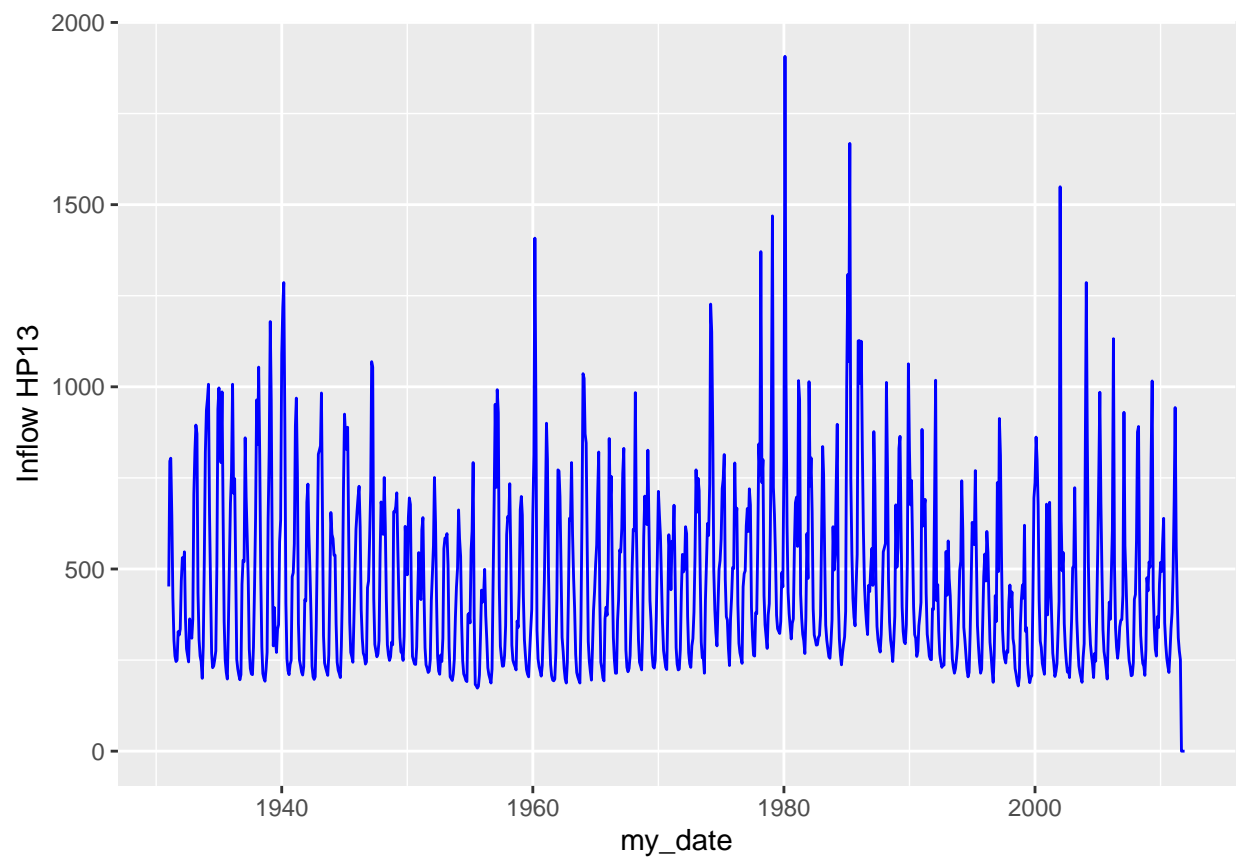


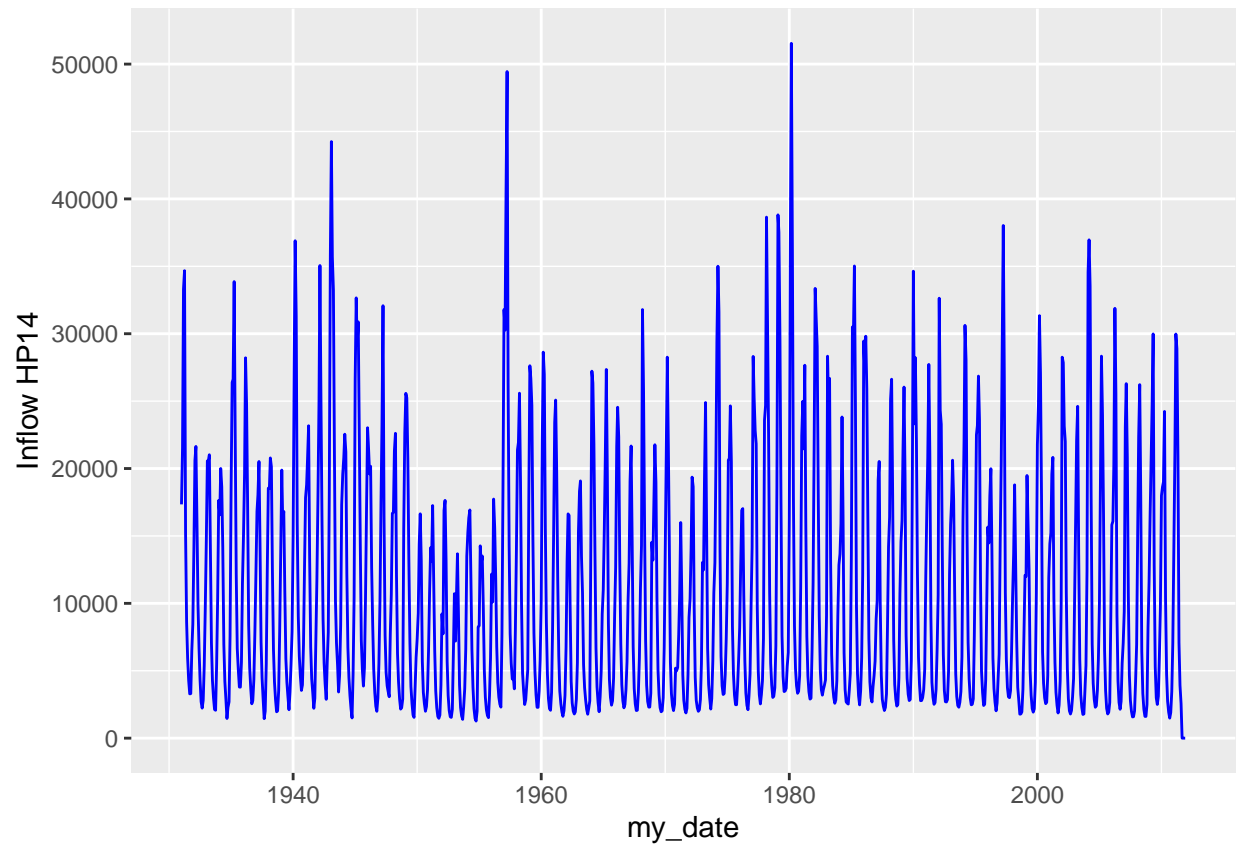


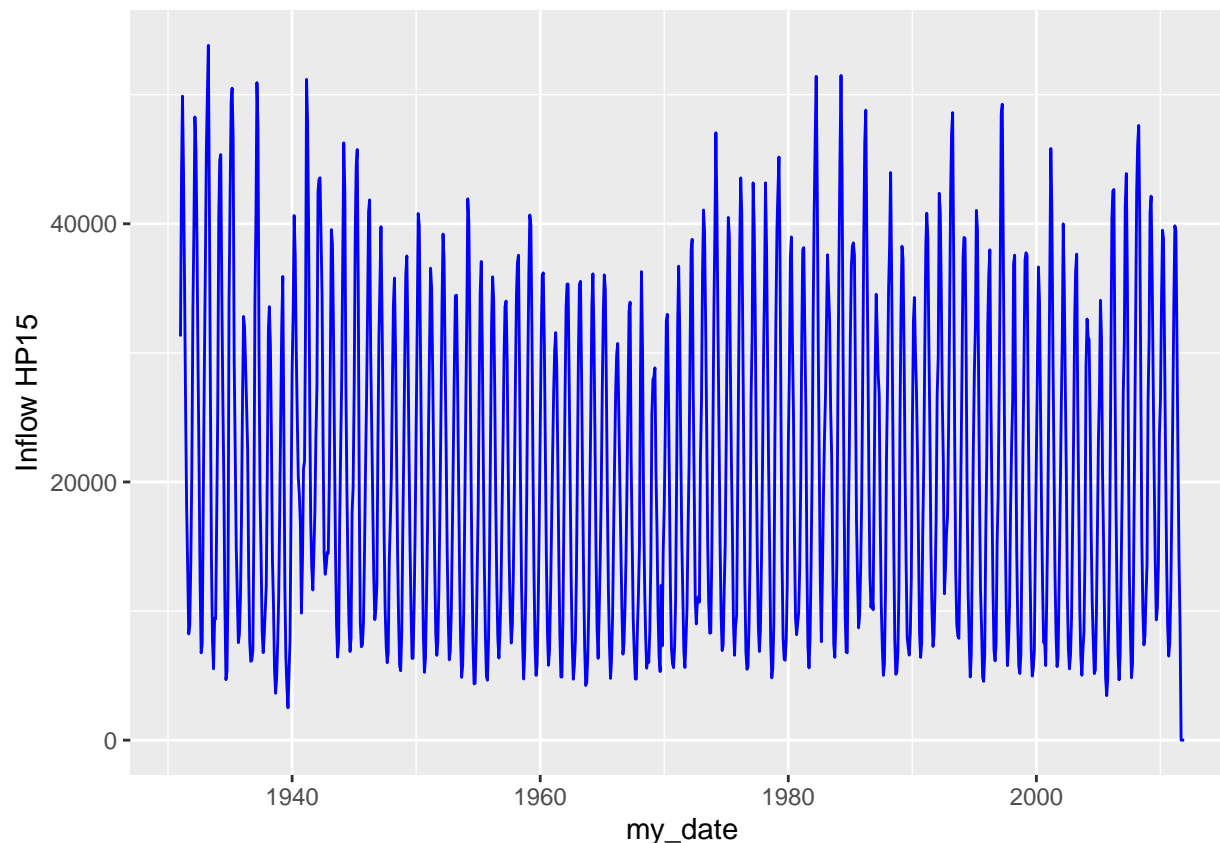












```
## Zeros in the end on data
```

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 for rows by replacing current data frame
inflow_data <- inflow_data[1:(nobs-4),]
```

```
#update object with number of observations
nobs <- nobs-4
```

```
#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
## 963 2011-03-01 8897 5426 5805 2009 3576 1834 798 2097 1071 3435 797 3693 943
## 964 2011-04-01 4991 3207 3323 4063 3235 1620 481 2325 902 2173 493 5255 563
## 965 2011-05-01 3025 2156 2274 2351 2063 572 304 1496 540 1175 254 1998 415
## 966 2011-06-01 2415 1813 1936 1836 2087 713 270 2294 898 985 130 1256 311
## 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
##      HP14 HP15
## 963 29976 39843
## 964 28892 39441
## 965 20978 31023
## 966 7081 21840
## 967 3910 14162
## 968 2561 8896
```

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=my_date[1],frequency=12)
#note that we are only transforming columns with inflow data, not the date columns
#remove start and see what happens
head(ts_inflow_data,15)
```

```
##      HP1 HP2 HP3 HP4 HP5 HP6 HP7 HP8 HP9 HP10 HP11 HP12 HP13 HP14
## Jan -14245 4782 4076 2518 2450 2649 1462 450 968 246 2636 452 4870 452 17342
## Feb -14245 7323 7681 4188 150 2401 758 554 219 74 4158 457 4550 796 21530
## Mar -14245 8266 5921 3253 2389 3261 707 615 333 123 3847 631 6537 804 33299
## Apr -14245 6247 4600 2449 1253 2006 469 474 297 113 3291 510 7298 644 34674
## May -14245 3642 2789 1651 2374 2454 3167 378 3295 938 1956 276 4942 421 15184
## Jun -14245 2425 2062 1270 2672 2433 3236 301 2547 951 1371 201 2478 305 8611
## Jul -14245 2158 1644 1204 1238 1798 1957 256 2585 883 1186 213 1905 261 5939
## Aug -14245 1854 1301 1152 605 1160 844 244 1173 404 1049 196 1647 246 4259
## Sep -14245 1839 1439 1297 1016 1584 1937 222 3596 378 1162 161 1453 250 3282
## Oct -14245 1896 1340 1259 674 1563 1484 355 1140 211 1507 208 1358 328 3305
## Nov -14245 2095 1447 1218 674 1404 835 371 563 252 1996 596 1905 319 6500
## Dec -14245 2725 2479 2013 1278 2272 1073 419 512 197 3015 381 2121 335 8461
## Jan -14244 4679 4021 2435 1259 1995 1044 520 609 159 3978 711 3811 467 14002
## Feb -14244 5535 4082 2262 1895 2996 1454 525 1219 268 2615 316 4681 531 20596
```

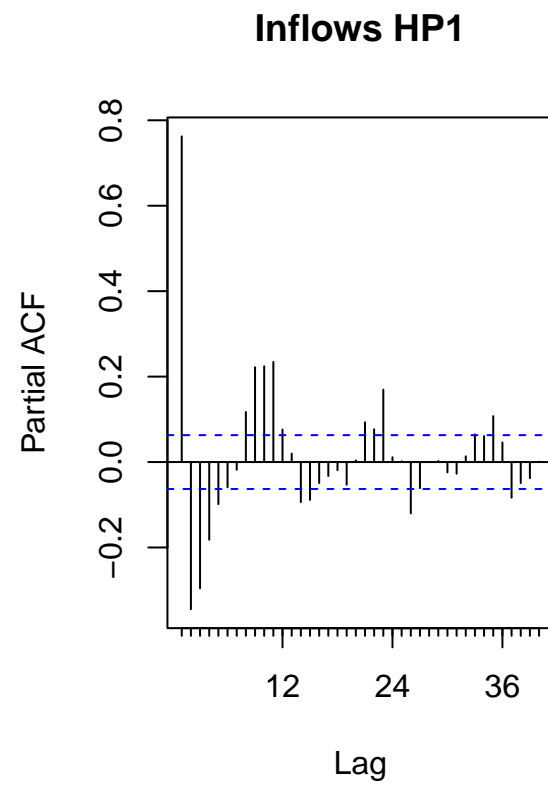
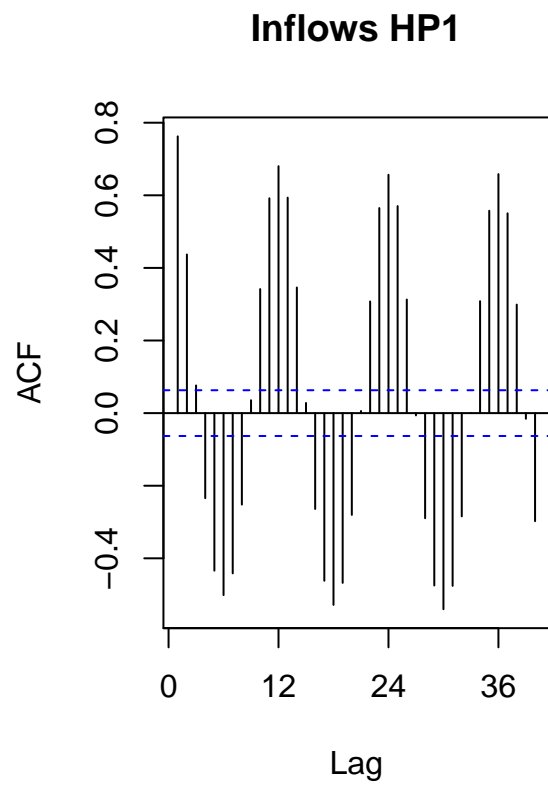
```
## Mar -14244 4310 3398 2065 1686 2392 1888 674 1332 304 2269 271 3329 501 21638
## HP15
## Jan -14245 31270
## Feb -14245 43827
## Mar -14245 49884
## Apr -14245 43962
## May -14245 35156
## Jun -14245 25764
## Jul -14245 18109
## Aug -14245 13320
## Sep -14245 8225
## Oct -14245 8900
## Nov -14245 13766
## Dec -14245 20880
## Jan -14244 33160
## Feb -14244 39791
## Mar -14244 48274
```

Note that `ts_inflow_data` has information on start, end and frequency.

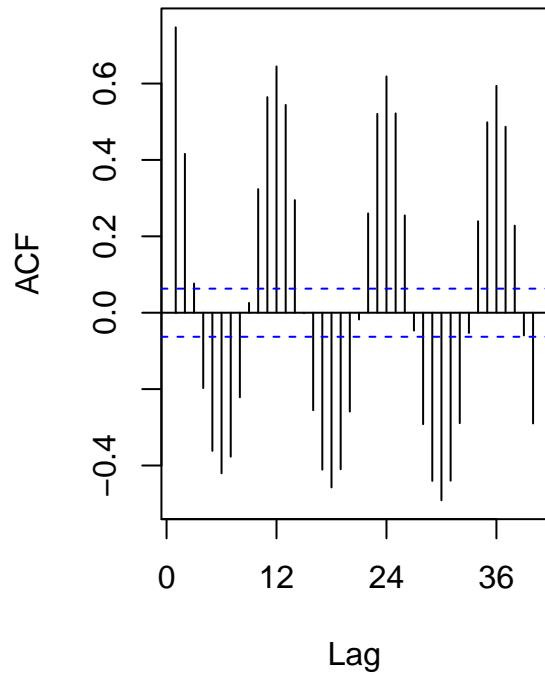
Plotting ACF and PACF

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

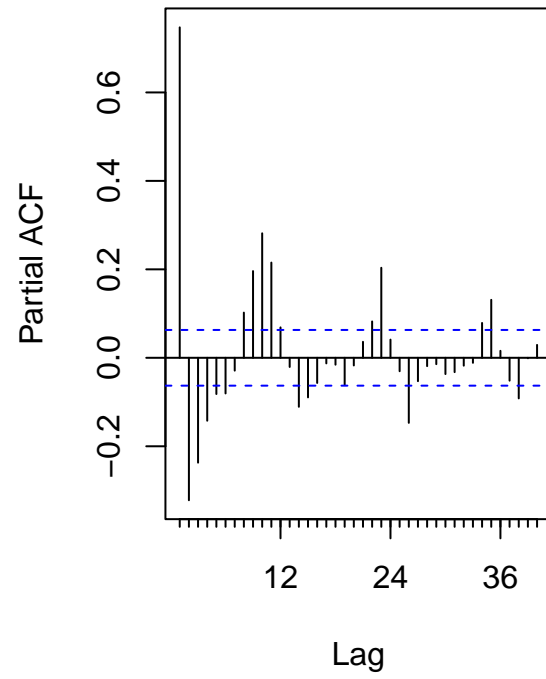
```
#Acf and Pacf for HP1
for(i in 1:nhydro){
  par(mfrow=c(1,2)) #place plot side by side
  Acf(ts_inflow_data[,i],lag.max=40,main=paste("Inflows HP",i,sep=""))
  # because I am not storing Acf() into any object, I don't need to specify plot=TRUE
  Pacf(ts_inflow_data[,i],lag.max=40,main=paste("Inflows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



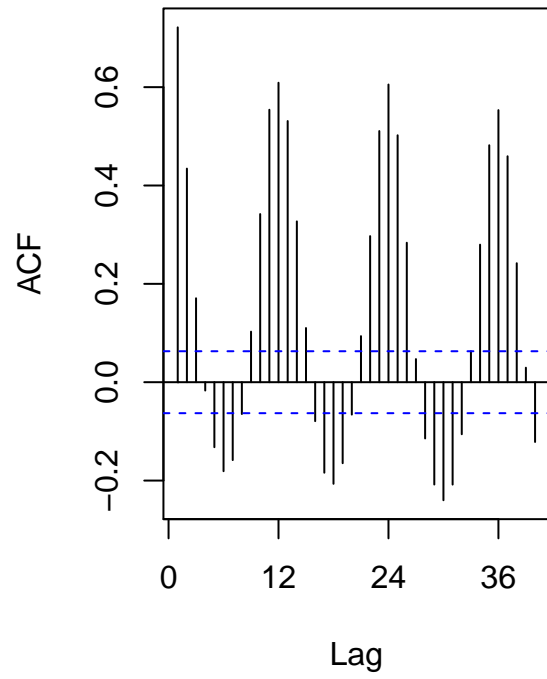
Inflows HP2



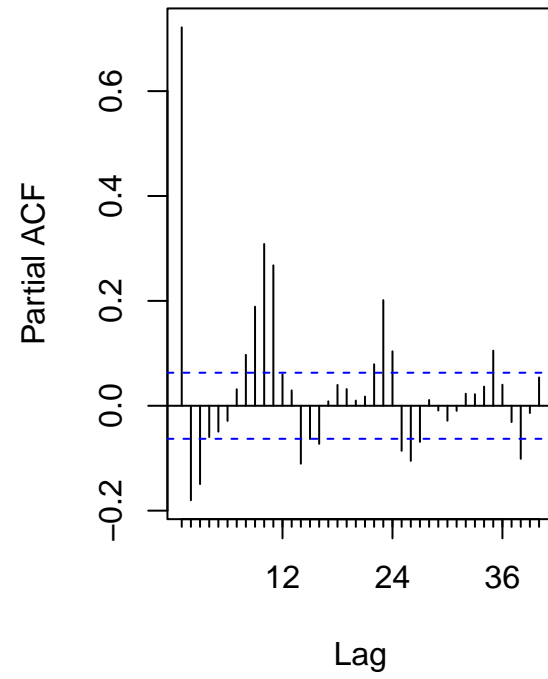
Inflows HP2



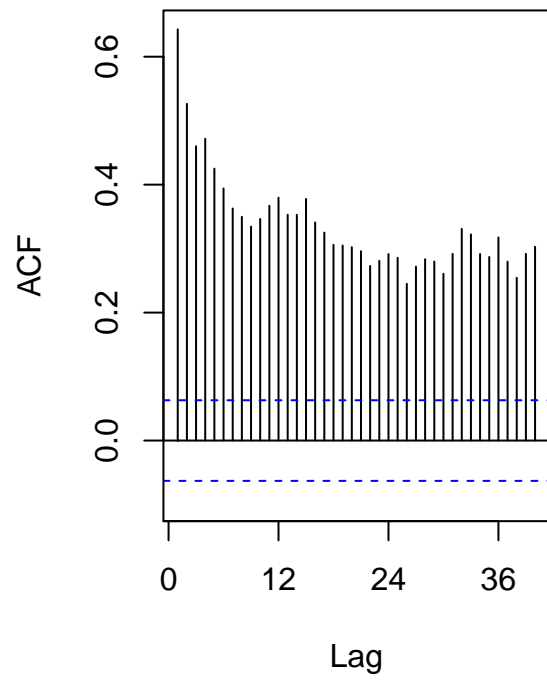
Inflows HP3



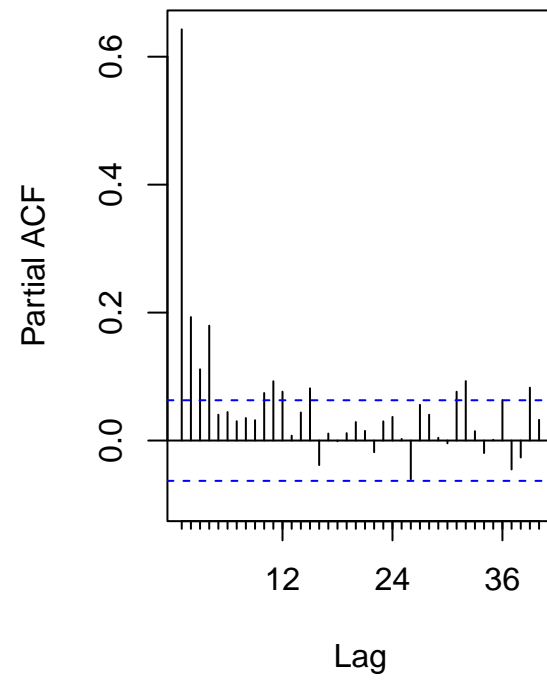
Inflows HP3



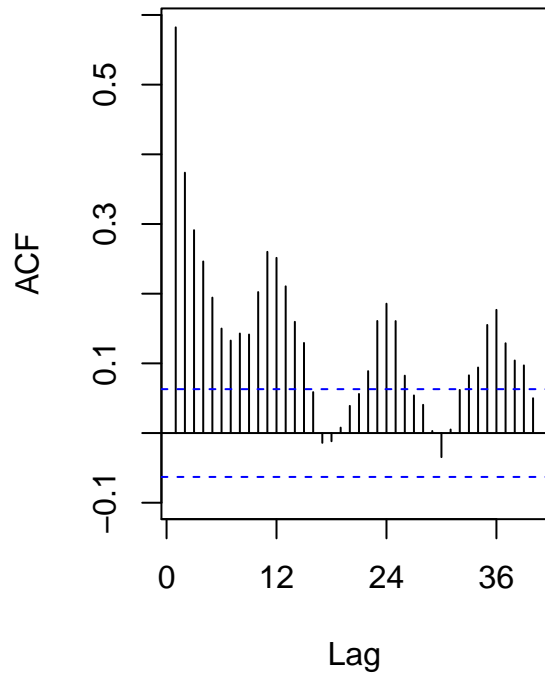
Inflows HP4



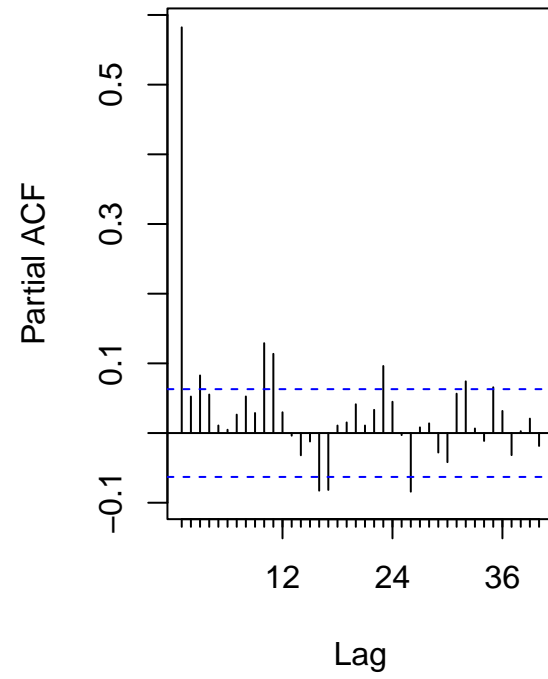
Inflows HP4



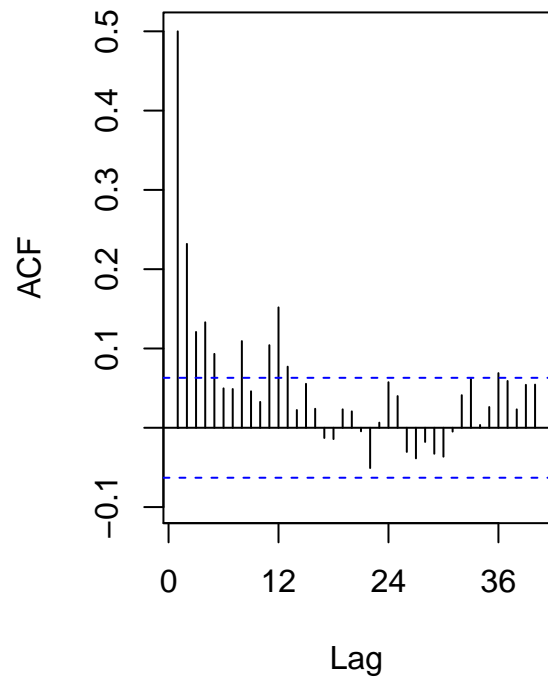
Inflows HP5



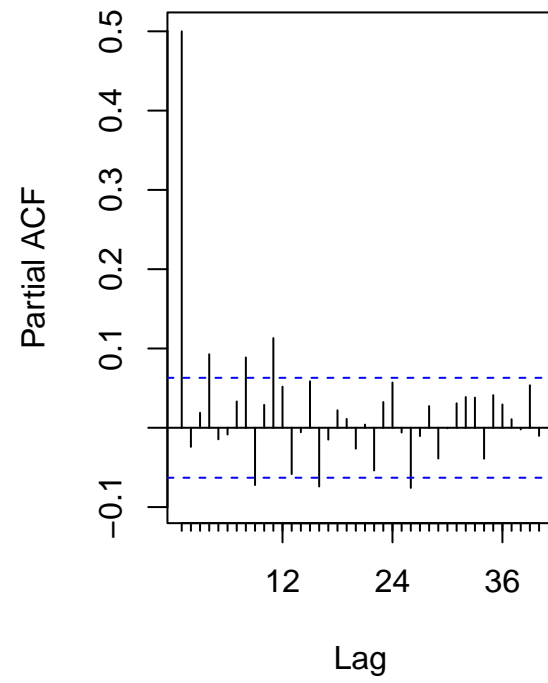
Inflows HP5



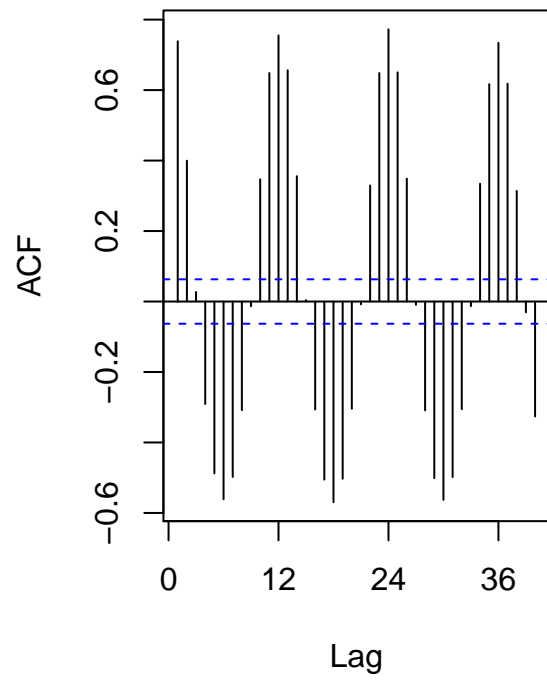
Inflows HP6



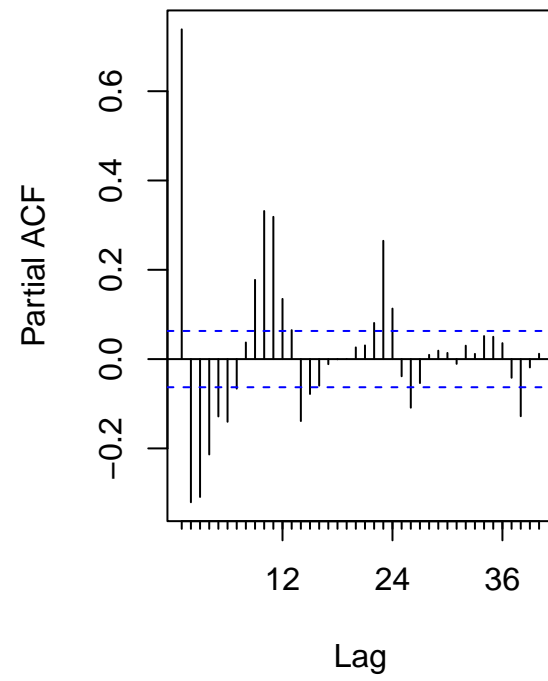
Inflows HP6



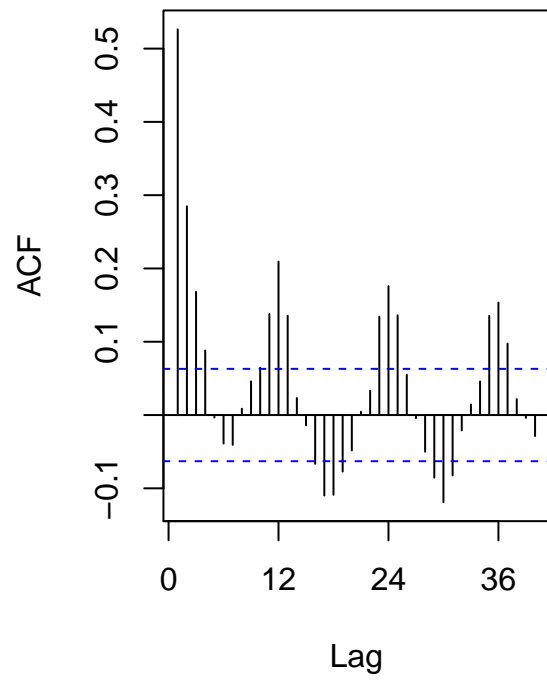
Inflows HP7



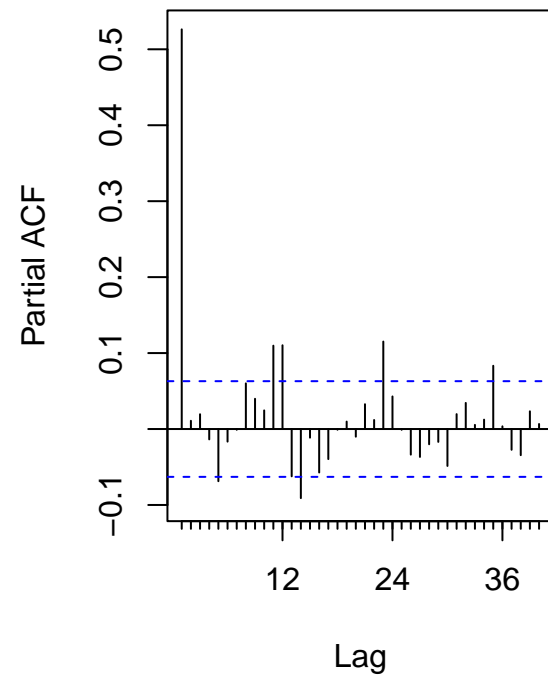
Inflows HP7



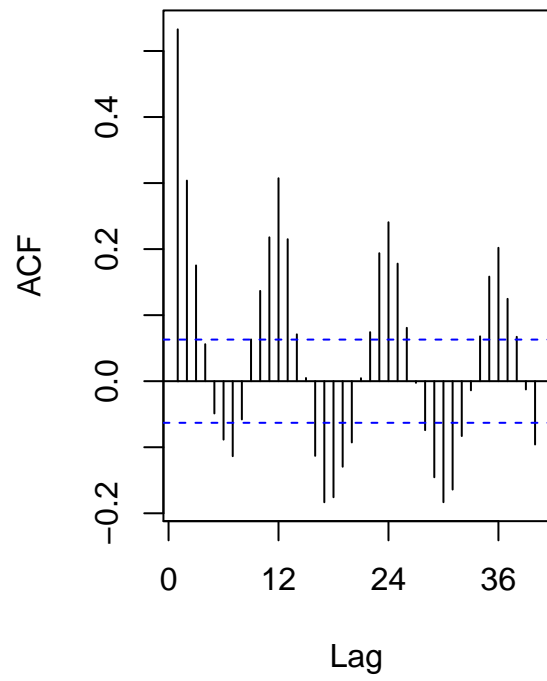
Inflows HP8



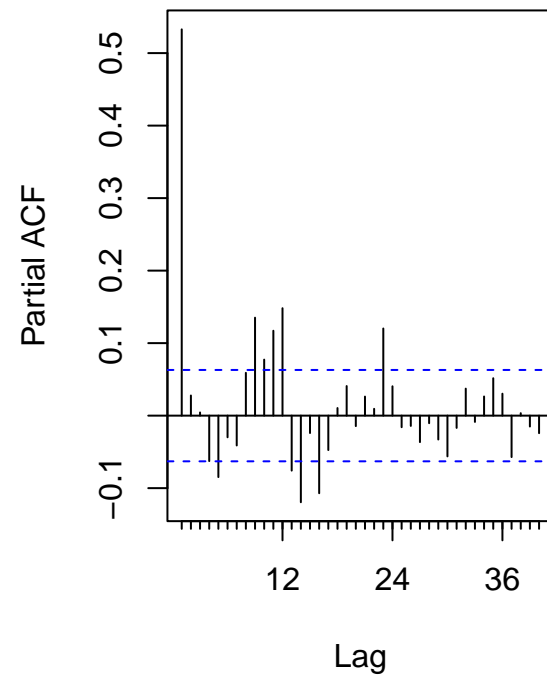
Inflows HP8



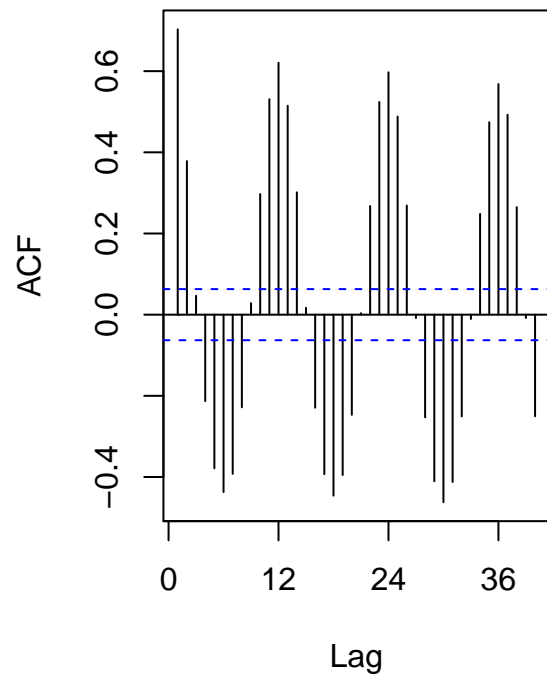
Inflows HP9



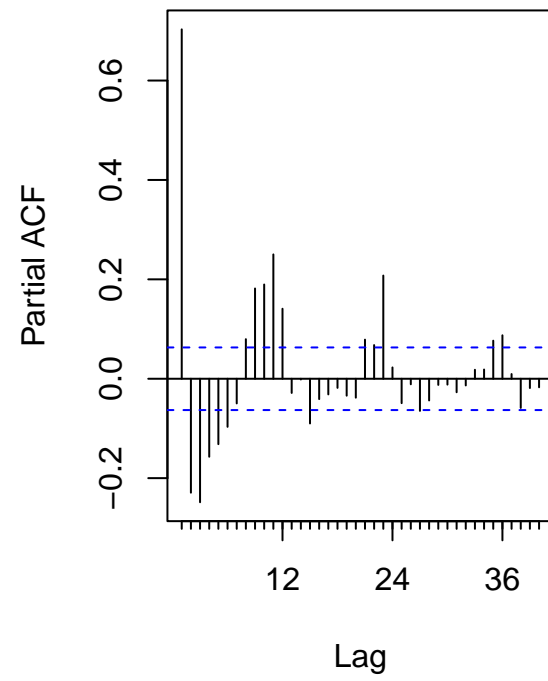
Inflows HP9



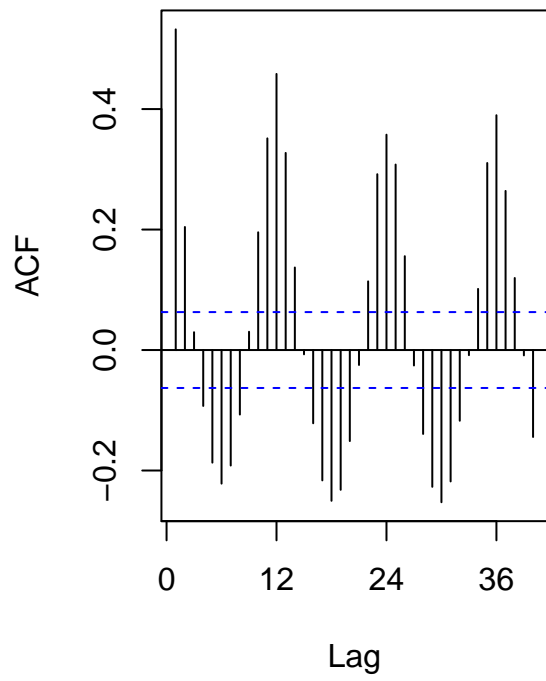
Inflows HP10



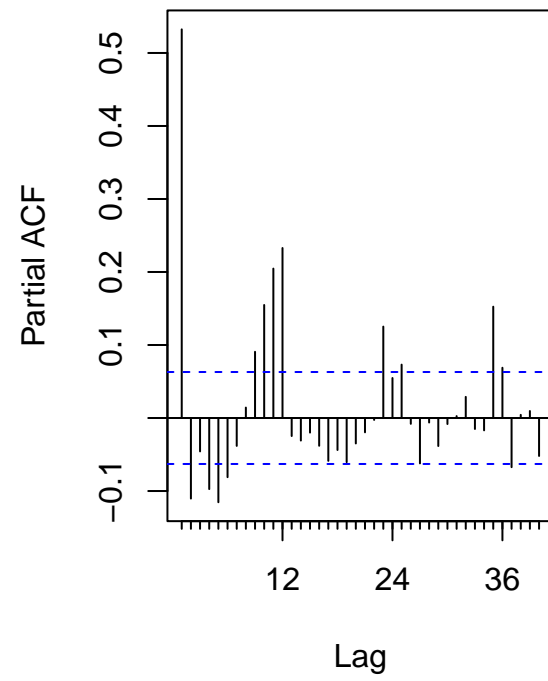
Inflows HP10

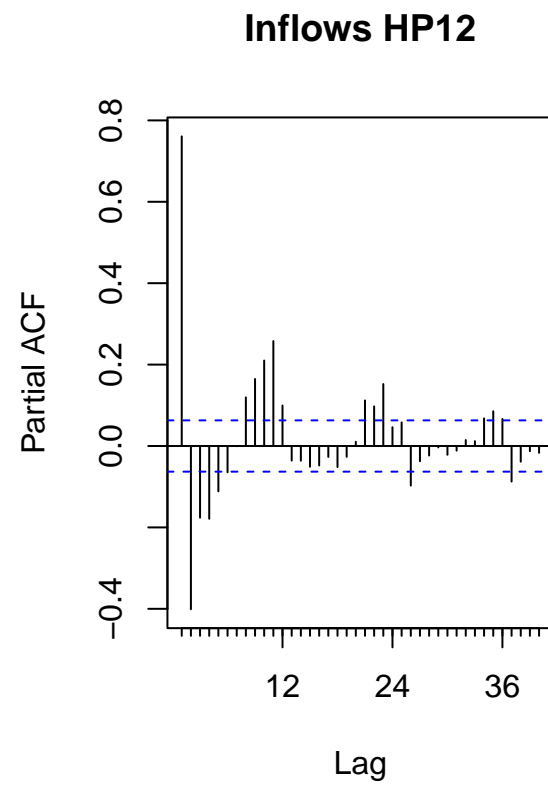
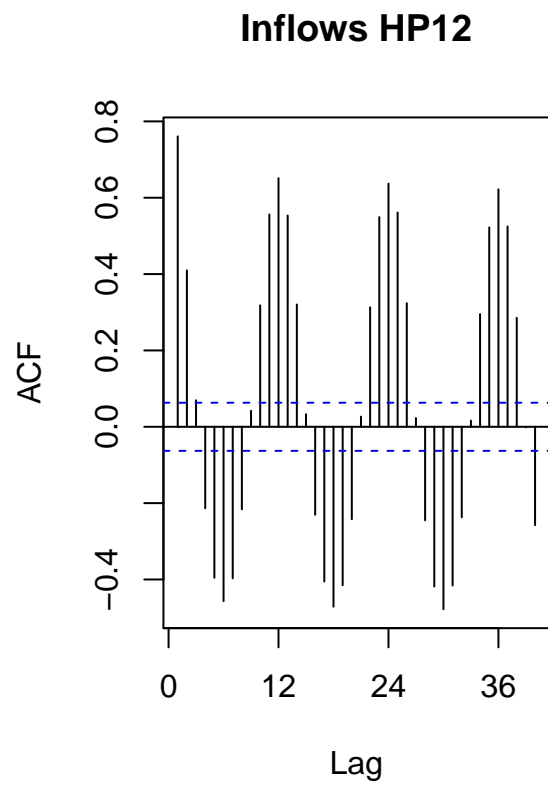


Inflows HP11

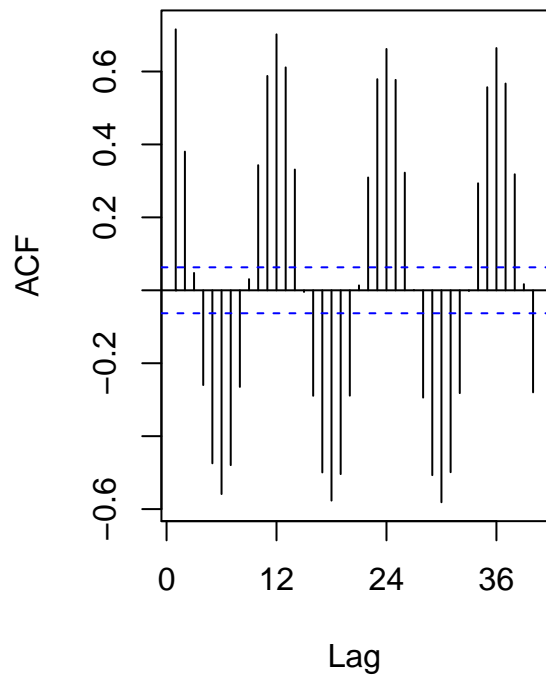


Inflows HP11

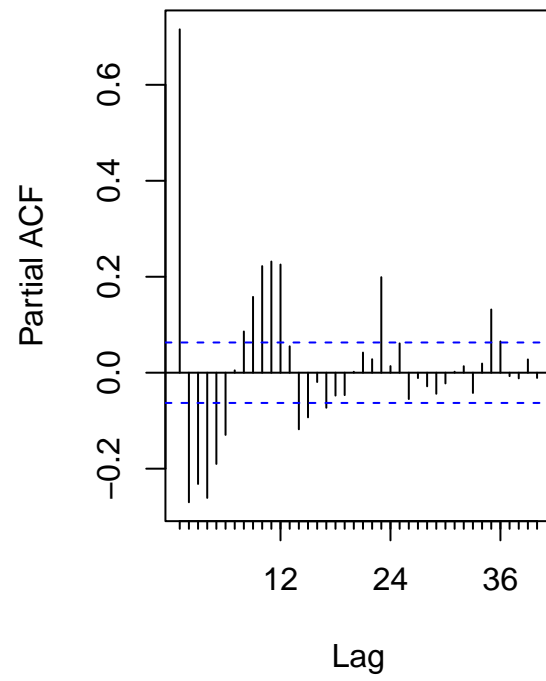




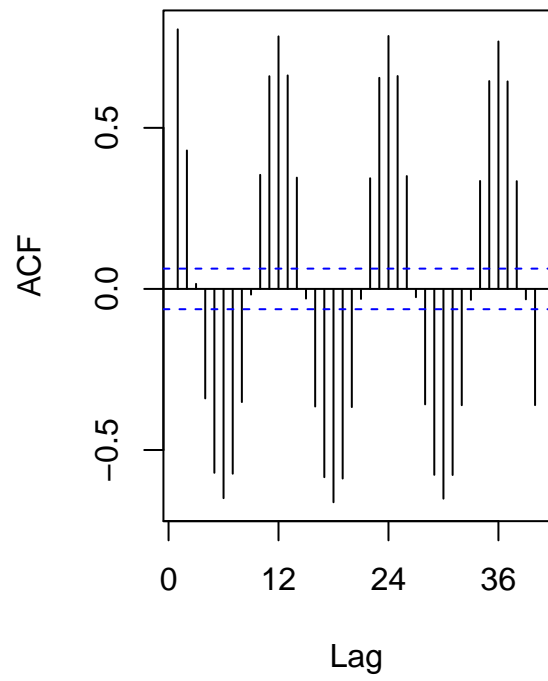
Inflows HP13



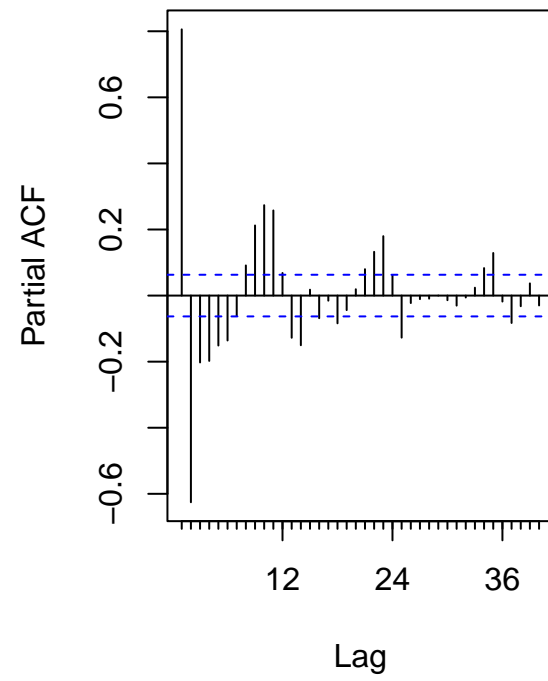
Inflows HP13

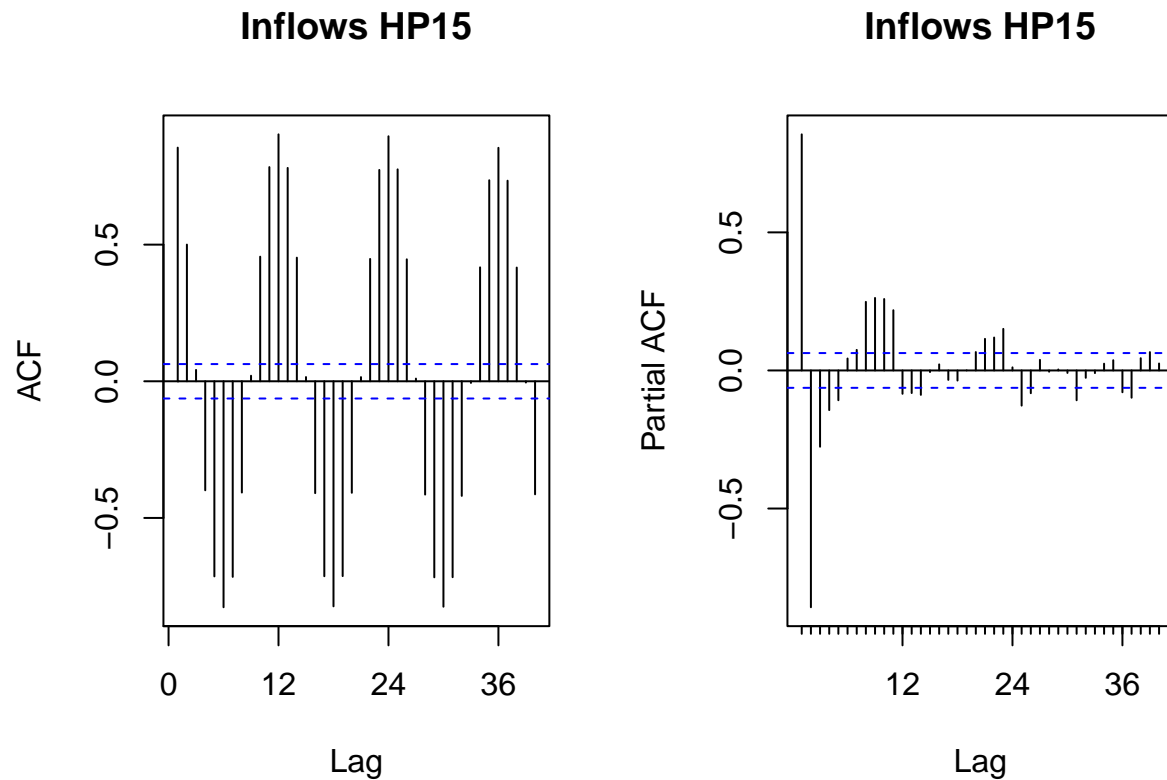


Inflows HP14



Inflows HP14





Trend Component

Let's identify and remove trend component like we leaned on the recorded videos for M4. You start by fitting a linear model to $Y_t = \beta_0 + \beta_1 * t + \epsilon_t$.

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

#Choose one hydro plant to study, as an exercise try to generalize this routine for all 15 HP
#from the plot HP4 seems to have a trend so let's play with that column
iHP=4 #change this to chekc other HP
#prep_data <- data.frame("Inflow"=inflow_data[,iHP], "Time"=t)

#Fit a linear trend to TS of iHP
linear_trend_model=lm(inflow_data[,iHP+1]~t)
summary(linear_trend_model)

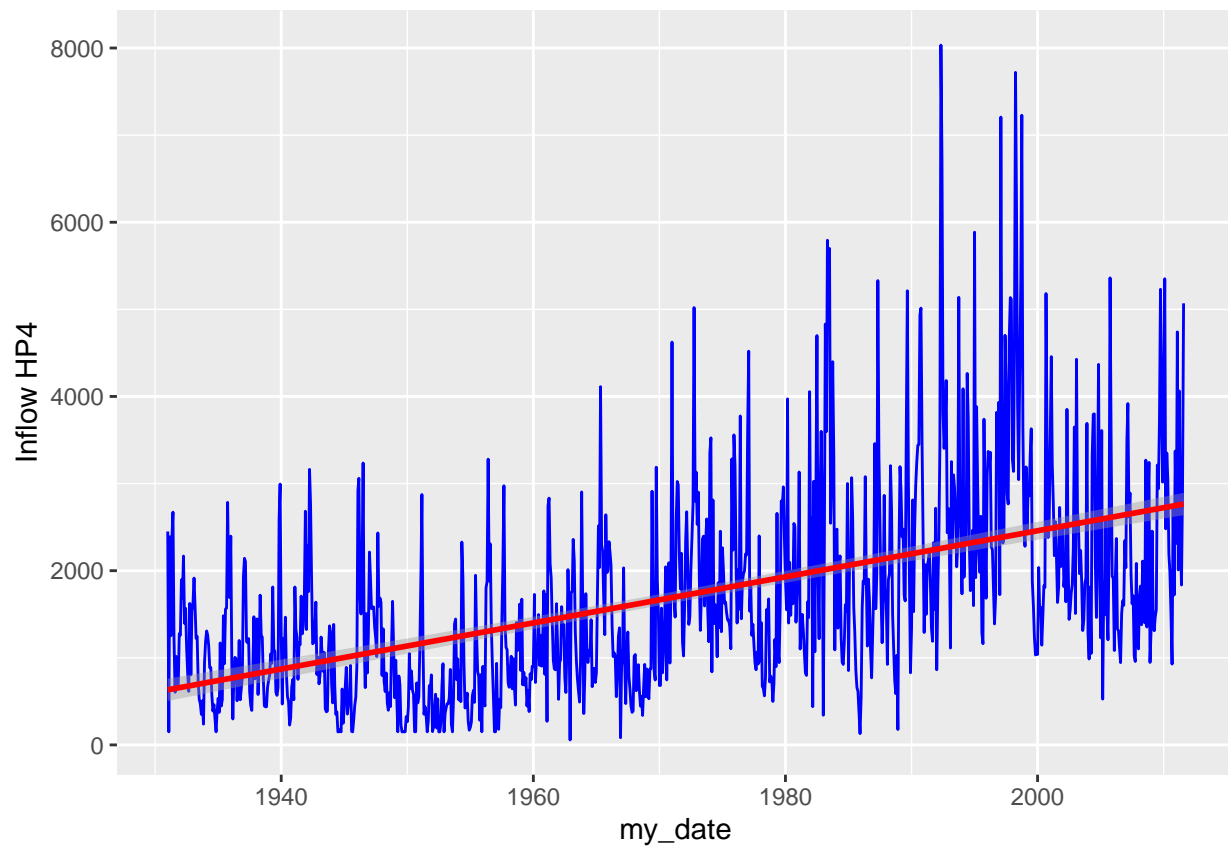
##
## Call:
## lm(formula = inflow_data[, iHP + 1] ~ t)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2069.3  -695.1  -220.5   505.6  5777.2
##
```

```
## Coefficients:
##           Estimate Std. Error t value Pr(>|t|)
## (Intercept) 630.6526    65.3049   9.657  <2e-16 ***
## t           2.2050     0.1168  18.885  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1015 on 966 degrees of freedom
## Multiple R-squared:  0.2696, Adjusted R-squared:  0.2689
## F-statistic: 356.7 on 1 and 966 DF,  p-value: < 2.2e-16
```

```
beta0=as.numeric(linear_trend_model$coefficients[1]) #first coefficient is the intercept term or beta0
beta1=as.numeric(linear_trend_model$coefficients[2]) #second coefficient is the slope or beta1
```

```
#Let's plot the time series with its trend line
ggplot(inflow_data, aes(x=my_date, y=inflow_data[, (1+iHP)])) +
  geom_line(color="blue") +
  ylab(paste0("Inflow ", colnames(inflow_data)[(1+iHP)], sep="")) +
  #geom_abline(intercept = beta0, slope = beta1, color="red")
  geom_smooth(color="red", method="lm")
```

```
## 'geom_smooth()' using formula 'y ~ x'
```



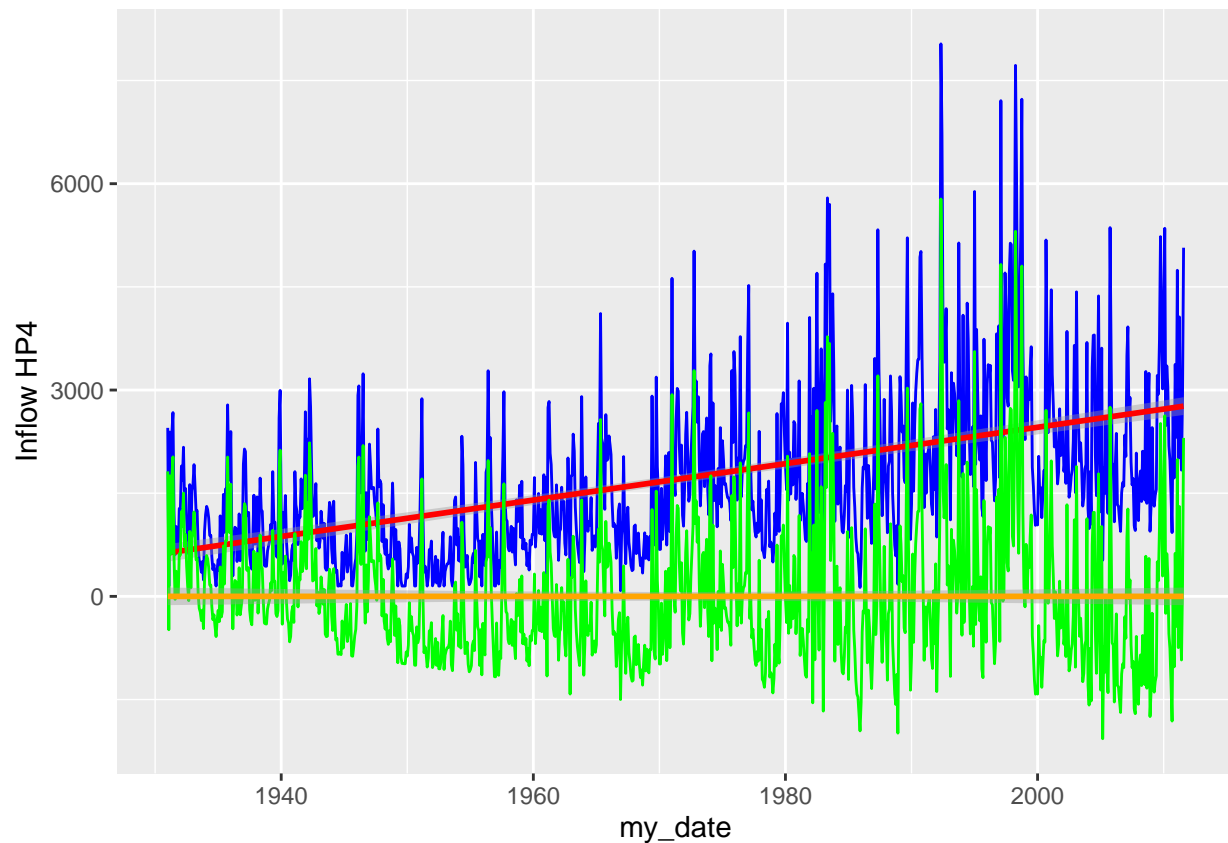
```

#remove the trend from series
detrend_inflow_data <- inflow_data[, (iHP+1)] - (beta0 + beta1*t)

#Understanding what we did
ggplot(inflow_data, aes(x=my_date, y=inflow_data[, (1+iHP)])) +
  geom_line(color="blue") +
  ylab(paste0("Inflow ", colnames(inflow_data)[(1+iHP)], sep="")) +
  #geom_abline(intercept = beta0, slope = beta1, color="red")
  geom_smooth(color="red", method="lm") +
  geom_line(aes(y=detrend_inflow_data), col="green") +
  geom_smooth(aes(y=detrend_inflow_data), color="orange", method="lm")

## 'geom_smooth()' using formula 'y ~ x'
## 'geom_smooth()' using formula 'y ~ x'

```



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.

```

#Let's choose another HP
iHP=1

#Use seasonal means model
#First create the seasonal dummies
dummies <- seasonaldummy(ts_inflow_data[,iHP])
#this function only accepts ts object, no need to add one here because date
#object is not a column

#Then fit a linear model to the seasonal dummies
seas_means_model=lm(inflow_data[, (iHP+1)]~dummies)
summary(seas_means_model)

```

```

##
## Call:
## lm(formula = inflow_data[, (iHP + 1)] ~ dummies)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -3397.3  -456.5   -43.1   340.9  5979.7
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3482.7      115.2  30.229 < 2e-16 ***
## dummiesJan    1213.3      162.4   7.470 1.81e-13 ***
## dummiesFeb    1452.6      162.4   8.943 < 2e-16 ***
## dummiesMar    1427.6      162.4   8.789 < 2e-16 ***
## dummiesApr     257.4      162.4   1.585  0.113
## dummiesMay   -992.8      162.4  -6.112 1.43e-09 ***
## dummiesJun  -1524.0      162.4  -9.382 < 2e-16 ***
## dummiesJul  -1883.6      162.4 -11.597 < 2e-16 ***
## dummiesAug  -2154.3      162.4 -13.263 < 2e-16 ***
## dummiesSep  -2245.3      162.9 -13.781 < 2e-16 ***
## dummiesOct  -2018.1      162.9 -12.386 < 2e-16 ***
## dummiesNov  -1335.4      162.9  -8.196 7.95e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1030 on 956 degrees of freedom
## Multiple R-squared:  0.6467, Adjusted R-squared:  0.6426
## F-statistic: 159.1 on 11 and 956 DF, p-value: < 2.2e-16

```

```

#Look at the regression coefficient. These will be the values of Beta

```

```

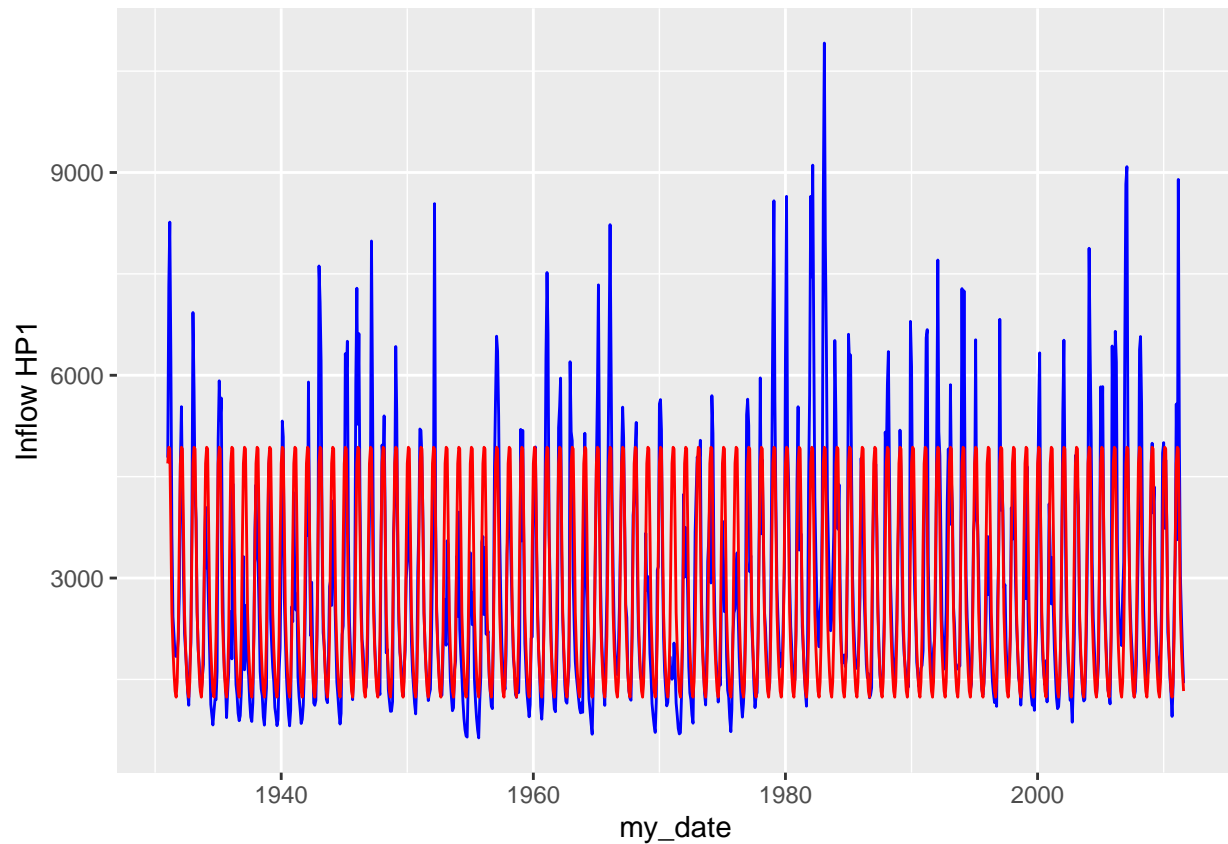
#Store regression coefficients
beta_int=seas_means_model$coefficients[1]
beta_coeff=seas_means_model$coefficients[2:12]

#compute seasonal component
inflow_seas_comp=array(0,nobs)
for(i in 1:nobs){
  inflow_seas_comp[i]=(beta_int+beta_coeff%*%dummies[i,])
}

```

```
#Understanding what we did
```

```
ggplot(inflow_data, aes(x=my_date, y=inflow_data[, (1+iHP)])) +  
  geom_line(color="blue") +  
  ylab(paste0("Inflow ", colnames(inflow_data)[(1+iHP)], sep="")) +  
  geom_line(aes(y=inflow_seas_comp), col="red")
```



```
#Removing seasonal component
```

```
deseason_inflow_data <- inflow_data[, (1+iHP)] - inflow_seas_comp
```

```
#Understanding what we did
```

```
ggplot(inflow_data, aes(x=my_date, y=inflow_data[, (1+iHP)])) +  
  geom_line(color="blue") +  
  ylab(paste0("Inflow ", colnames(inflow_data)[(1+iHP)], sep="")) +  
  geom_line(aes(y=deseason_inflow_data), col="green")
```

