M5: Outliers and Missing Data

Luana Lima

02/11/2021

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)

#New packages for M5
#install.packages("outliers")
library(outliers)
#install.packages("tidyverse")
library(tidyverse)
```

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
```

```r
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.

```r
#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"
```

```r
#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
```

Removing zeros in the end on data

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

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()*.

```r
ts_inflow_data <- ts(inflow_data[,2:(2+nhydro-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 1 4782 4076 2518 2450 2649 1462 450  968 246 2636  452 4870  452 17342
## Feb 1 7323 7681 4188  150 2401  758 554  219  74 4158  457 4550  796 21530
## Mar 1 8266 5921 3253 2389 3261  707 615  333 123 3847  631 6537  804 33299
## Apr 1 6247 4600 2449 1253 2006  469 474  297 113 3291  510 7298  644 34674
## May 1 3642 2789 1651 2374 2454 3167 378 3295 938 1956  276 4942  421 15184
## Jun 1 2425 2062 1270 2672 2433 3236 301 2547 951 1371  201 2478  305  8611
## Jul 1 2158 1644 1204 1238 1798 1957 256 2585 883 1186  213 1905  261  5939
## Aug 1 1854 1301 1152  605 1160  844 244 1173 404 1049  196 1647  246  4259
## Sep 1 1839 1439 1297 1016 1584 1937 222 3596 378 1162  161 1453  250  3282
## Oct 1 1896 1340 1259  674 1563 1484 355 1140 211 1507  208 1358  328  3305
```

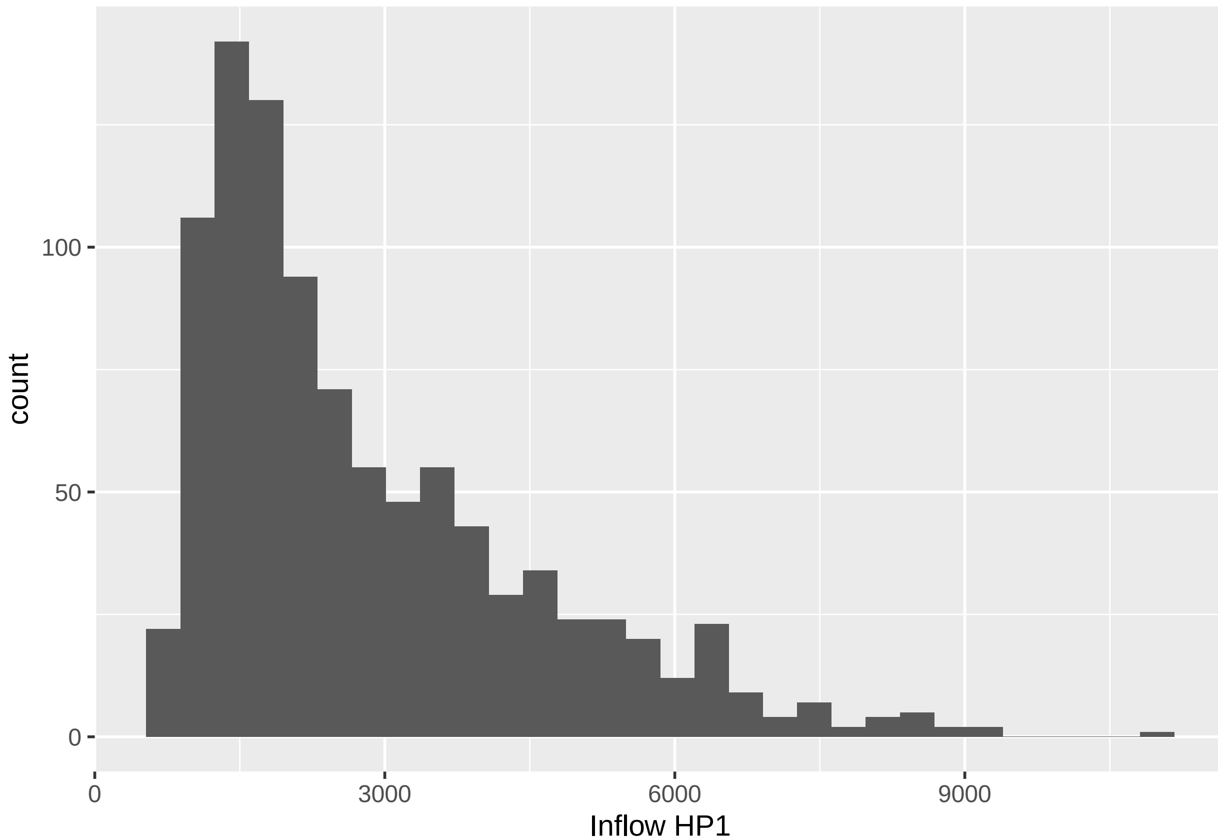
```
## Nov 1 2095 1447 1218  674 1404  835 371  563 252 1996  596 1905  319  6500
## Dec 1 2725 2479 2013 1278 2272 1073 419  512 197 3015  381 2121  335  8461
## Jan 2 4679 4021 2435 1259 1995 1044 520  609 159 3978  711 3811  467 14002
## Feb 2 5535 4082 2262 1895 2996 1454 525 1219 268 2615  316 4681  531 20596
## Mar 2 4310 3398 2065 1686 2392 1888 674 1332 304 2269  271 3329  501 21638
##        HP15
## Jan 1 31270
## Feb 1 43827
## Mar 1 49884
## Apr 1 43962
## May 1 35156
## Jun 1 25764
## Jul 1 18109
## Aug 1 13320
## Sep 1  8225
## Oct 1  8900
## Nov 1 13766
## Dec 1 20880
## Jan 2 33160
## Feb 2 39791
## Mar 2 48274
```

Initial plots for outlier detection

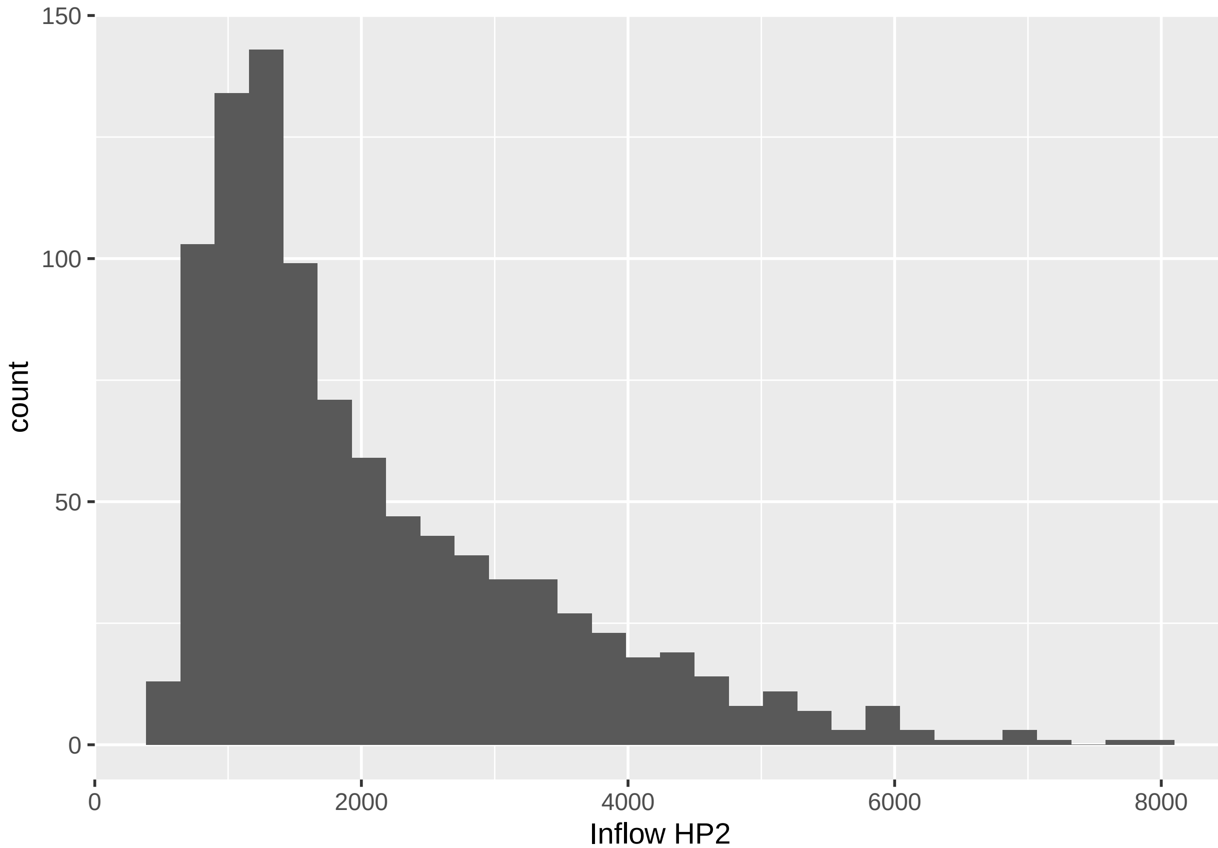
Common plots for outlier detection are histograms and boxplots. Histograms will help you understand the shape and spread of the data and to identify any potential outliers. And boxplots will give more information on the spread of the data.

```
#using package ggplot2 to make histograms
for(i in 1:nhydro){
  print(ggplot(inflow_data, aes(inflow_data[,(1+i)])) +
          geom_histogram() +
          xlab(paste0("Inflow ",colnames(inflow_data)[(1+i)],sep=""))
        )
}
```

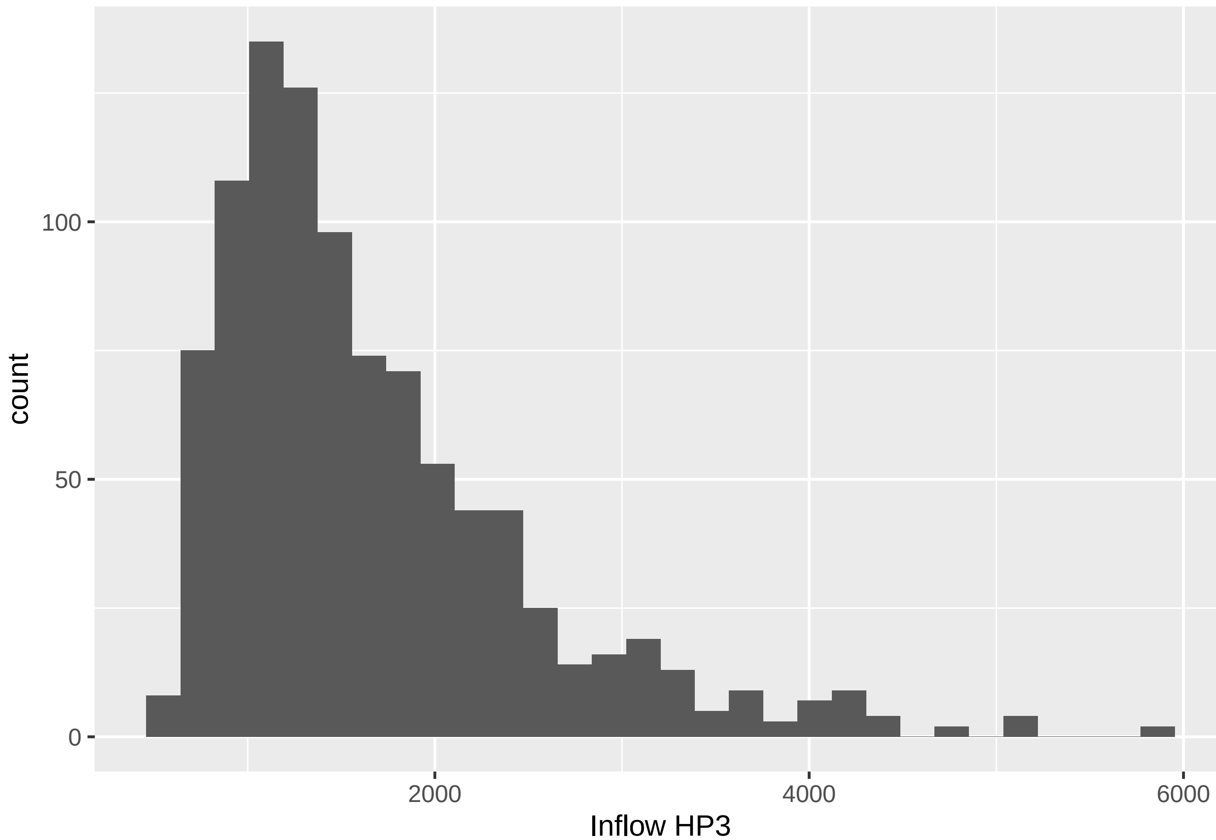
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



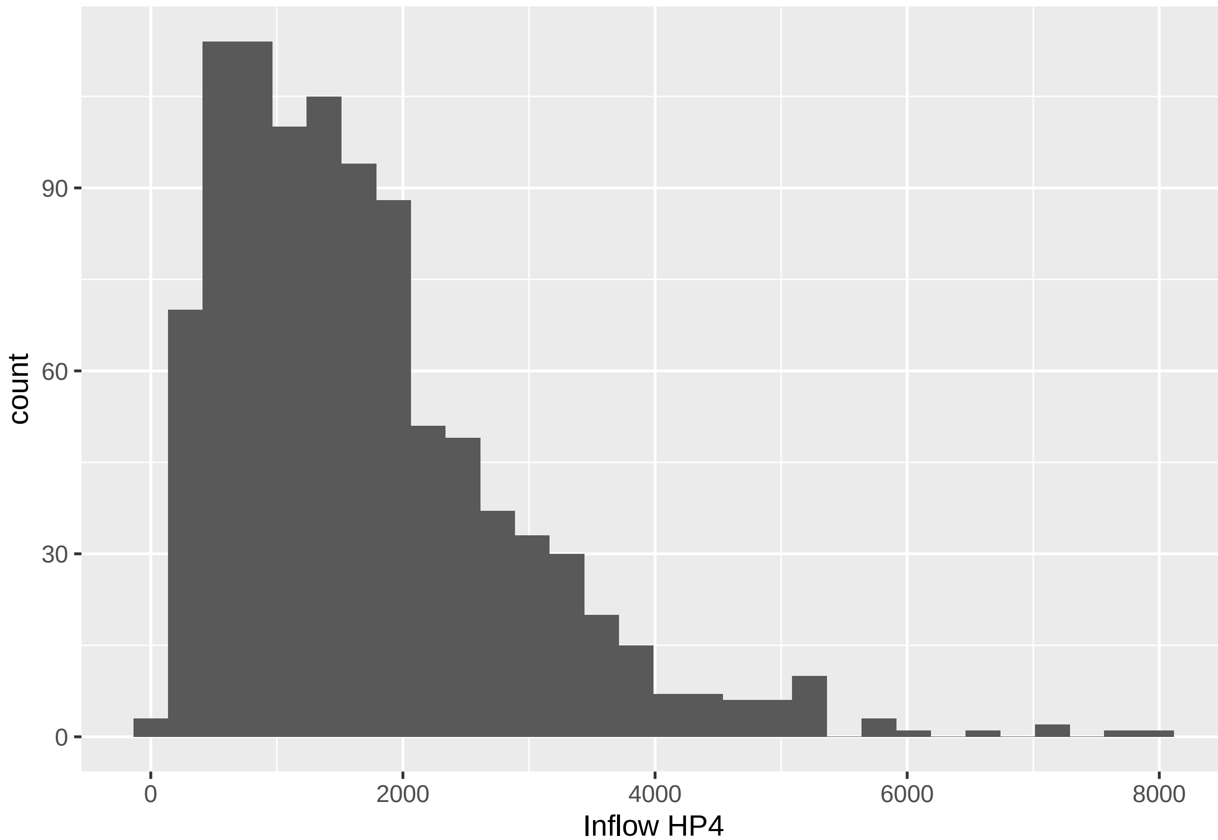

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



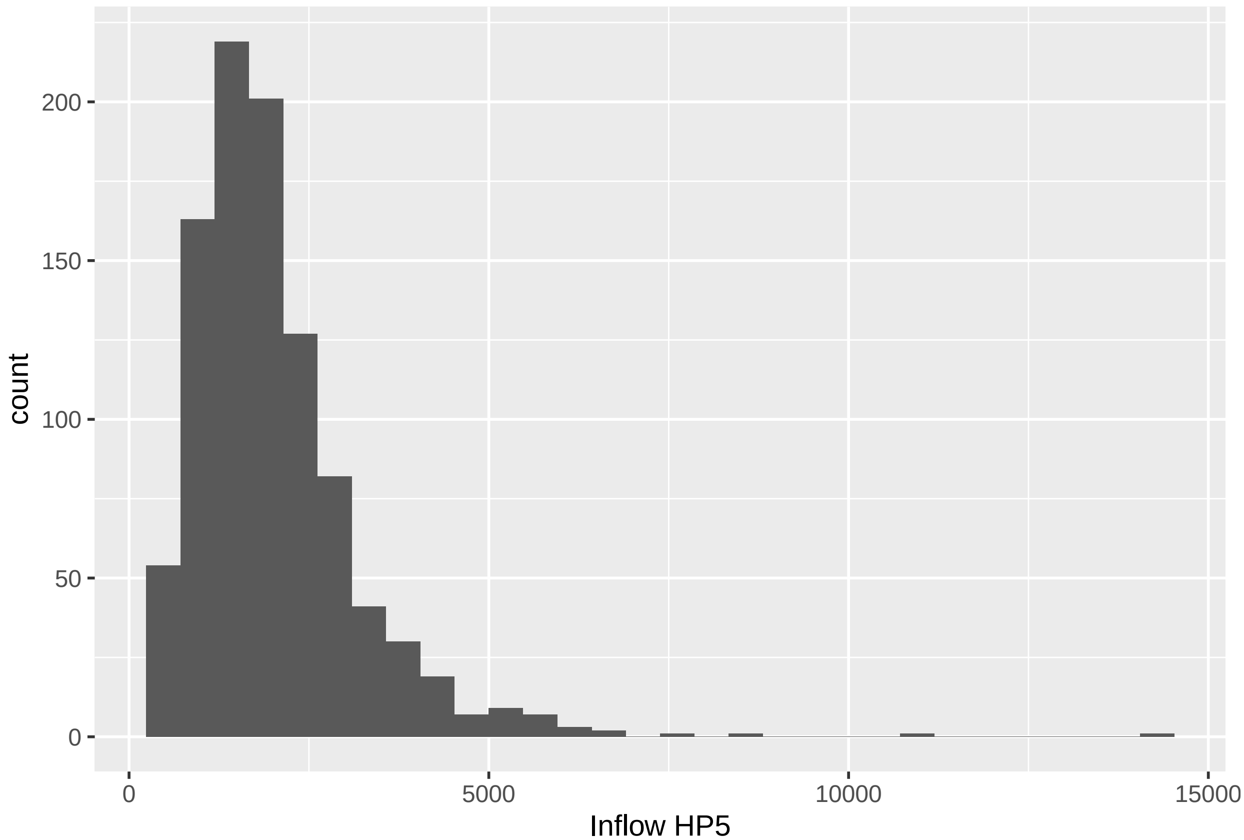

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



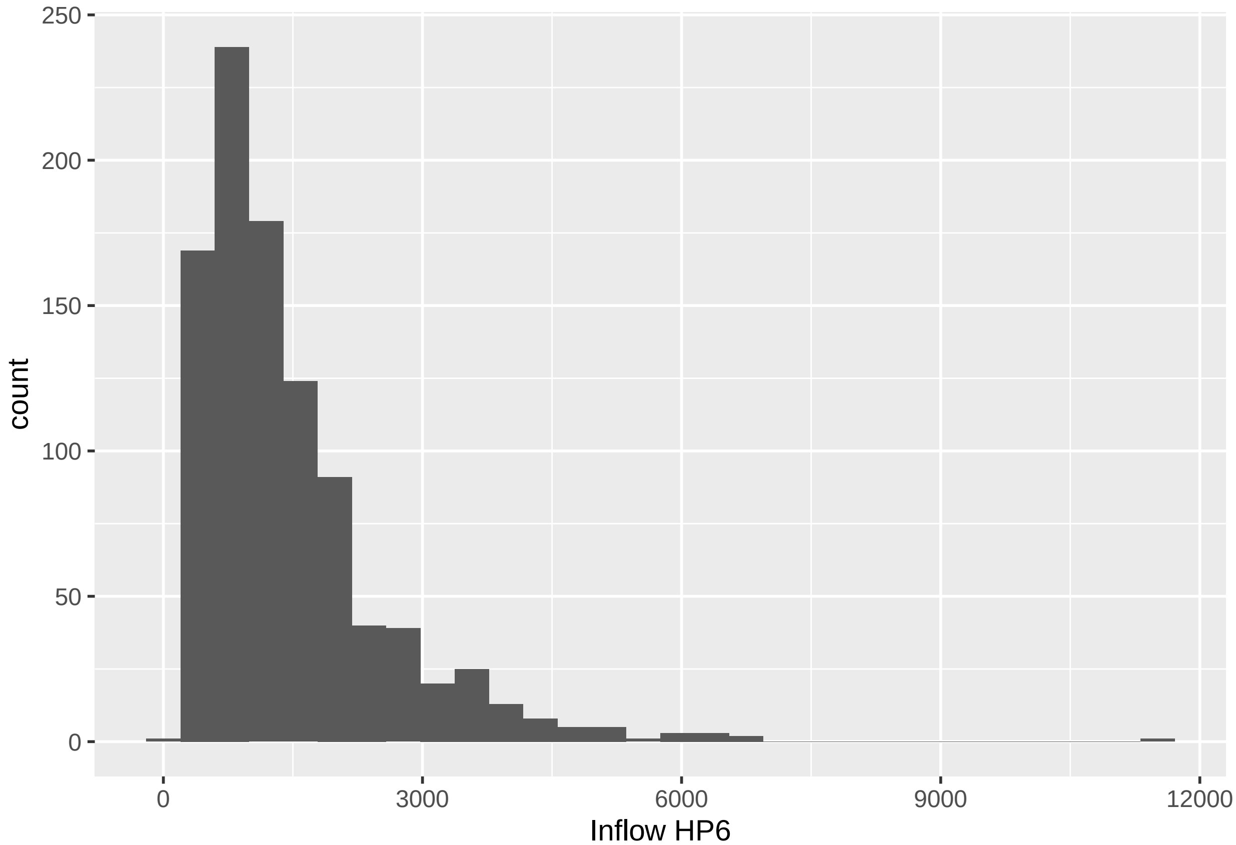

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



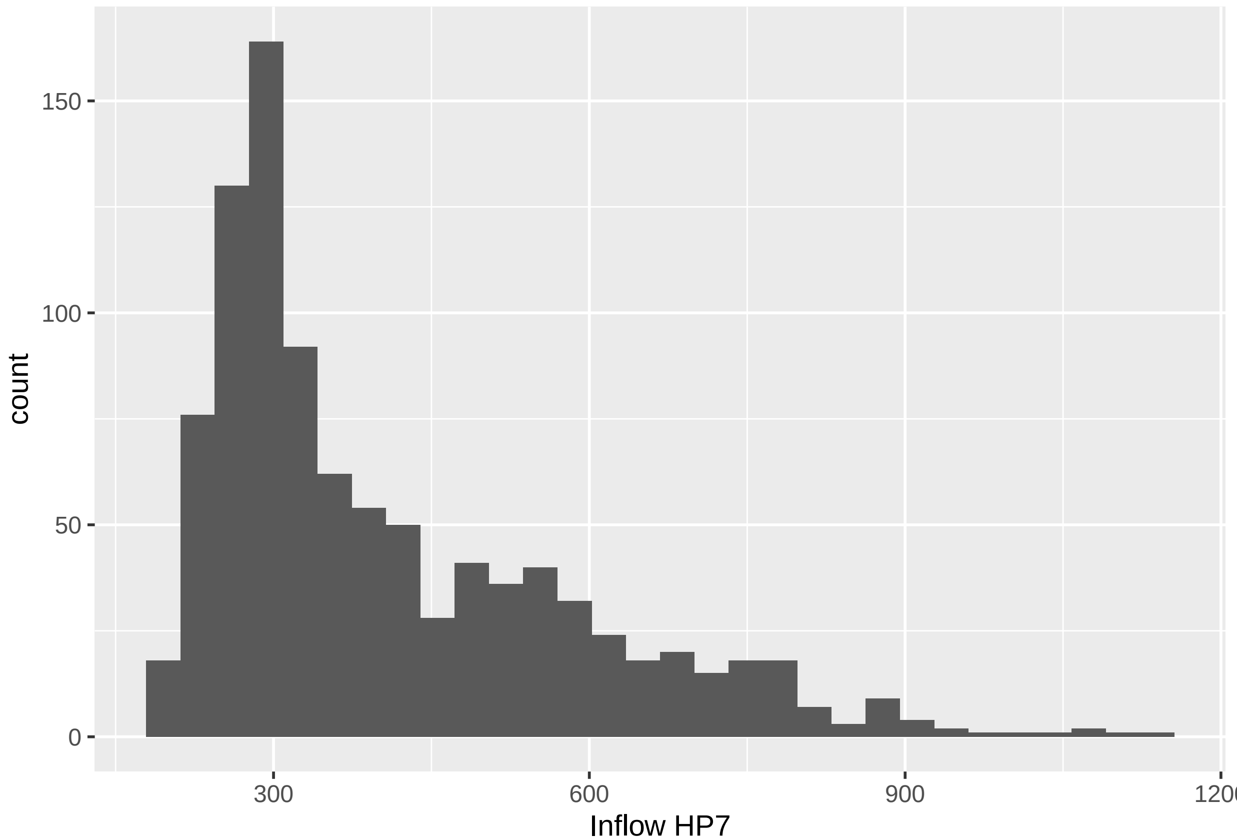

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



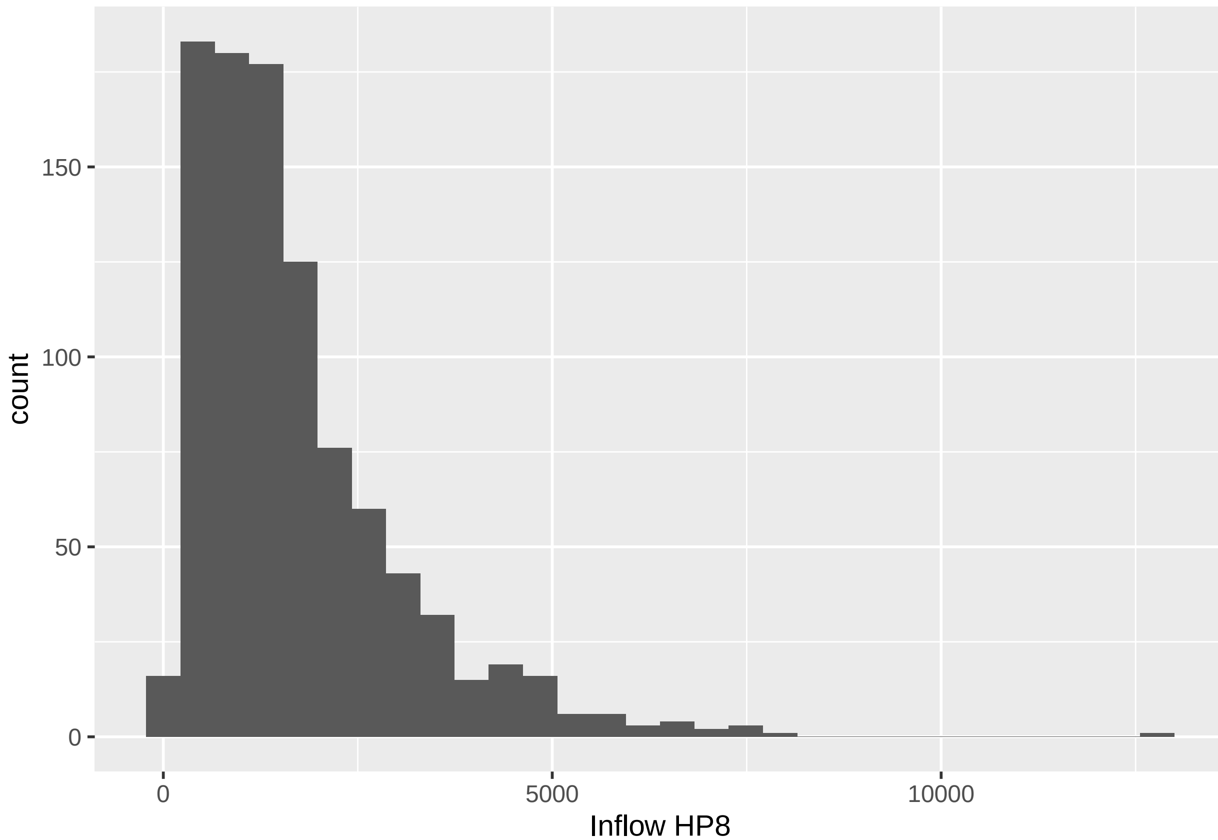

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



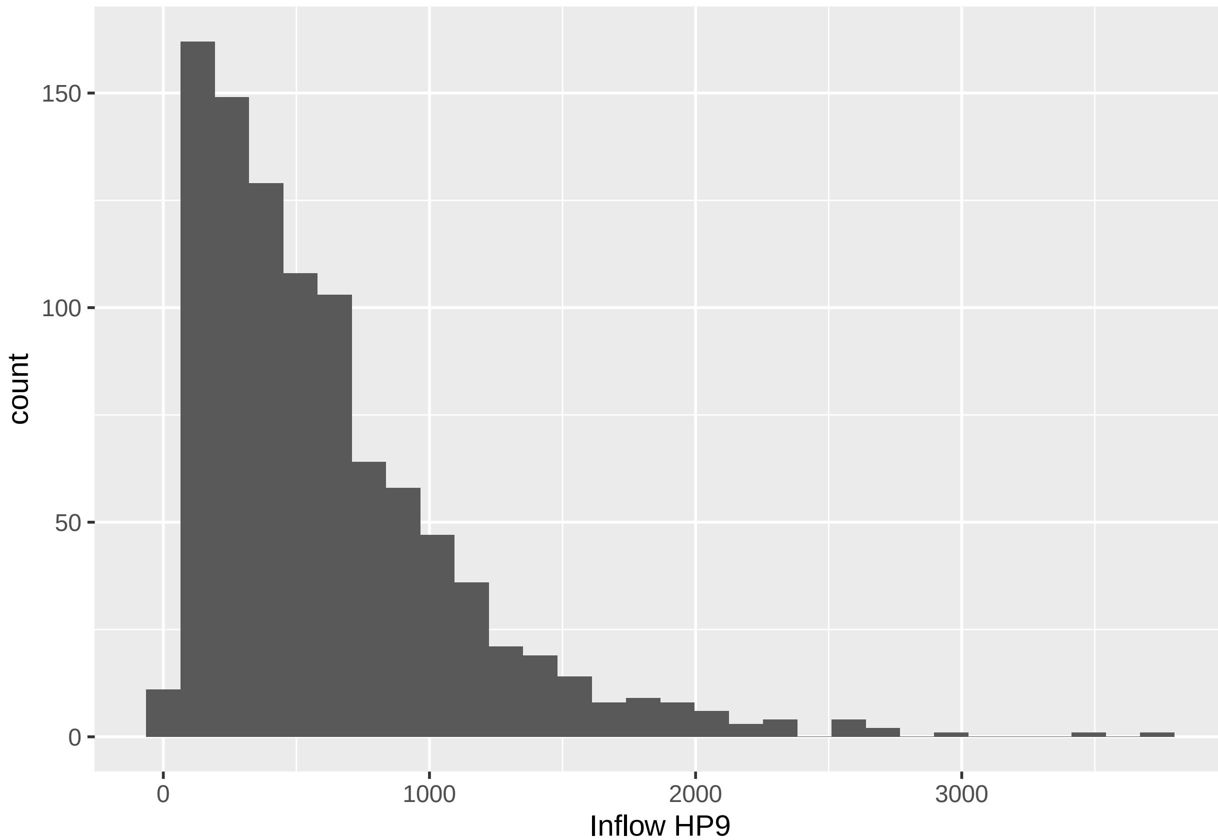

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



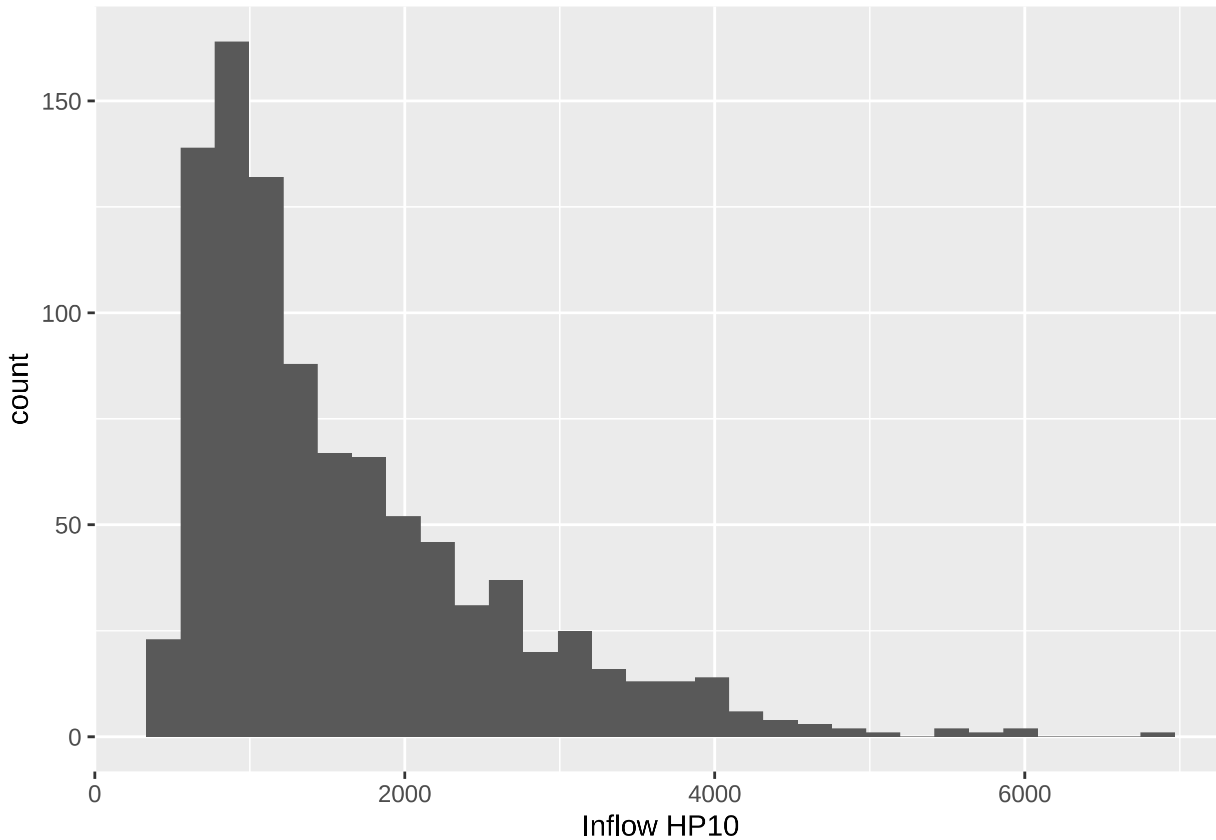

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



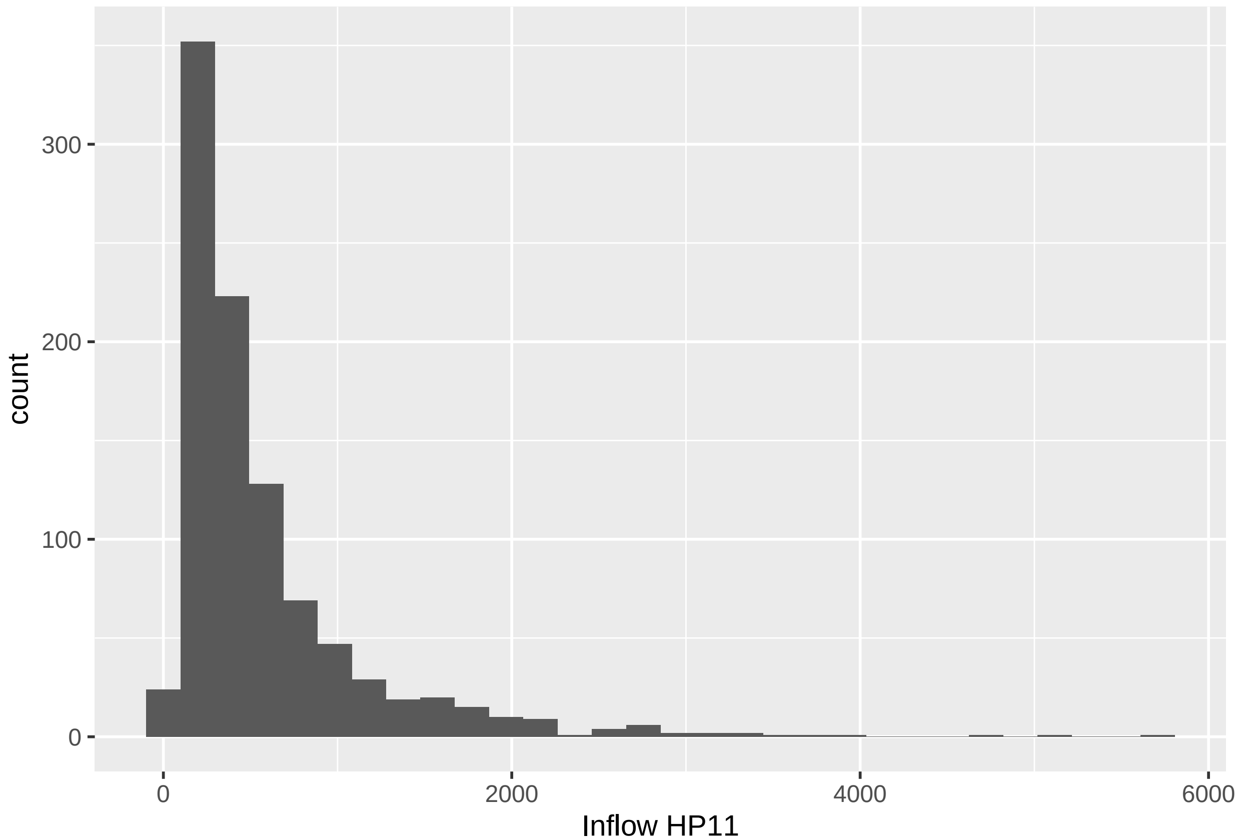

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



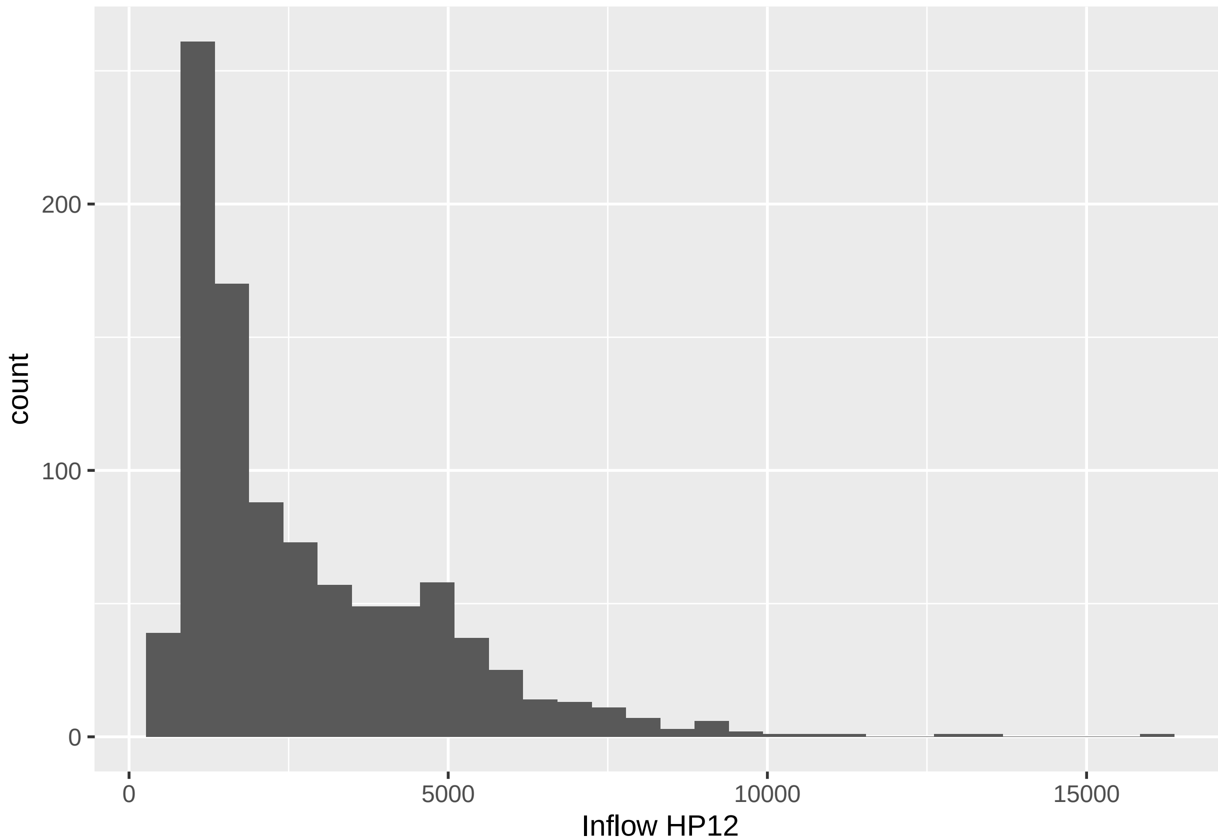

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



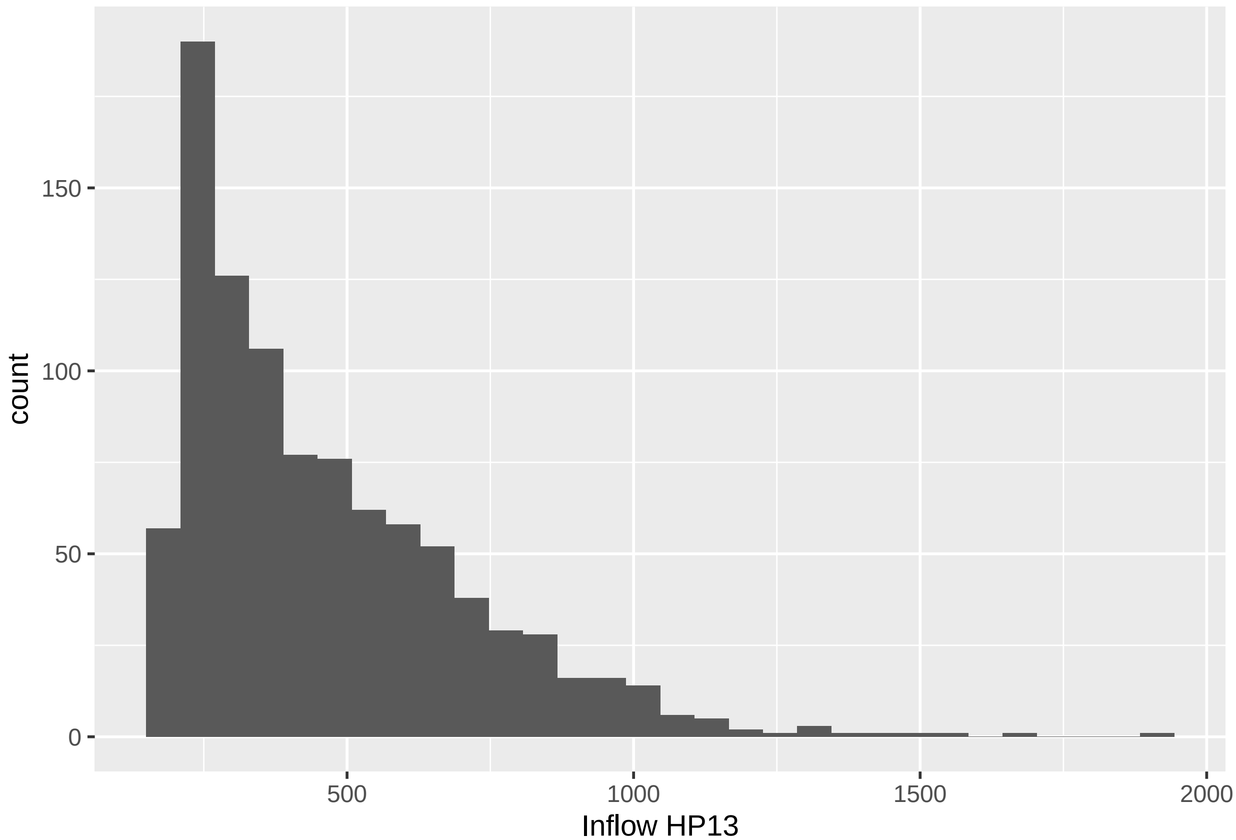

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



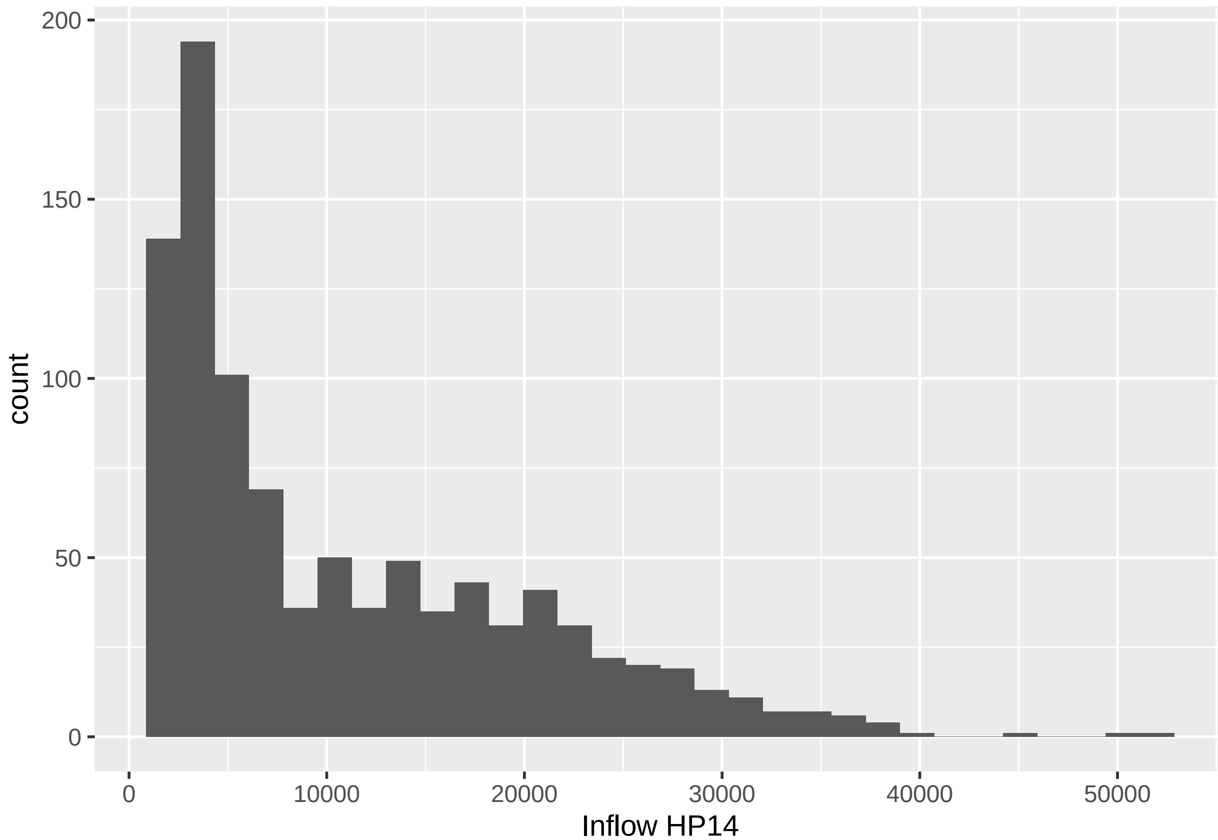

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



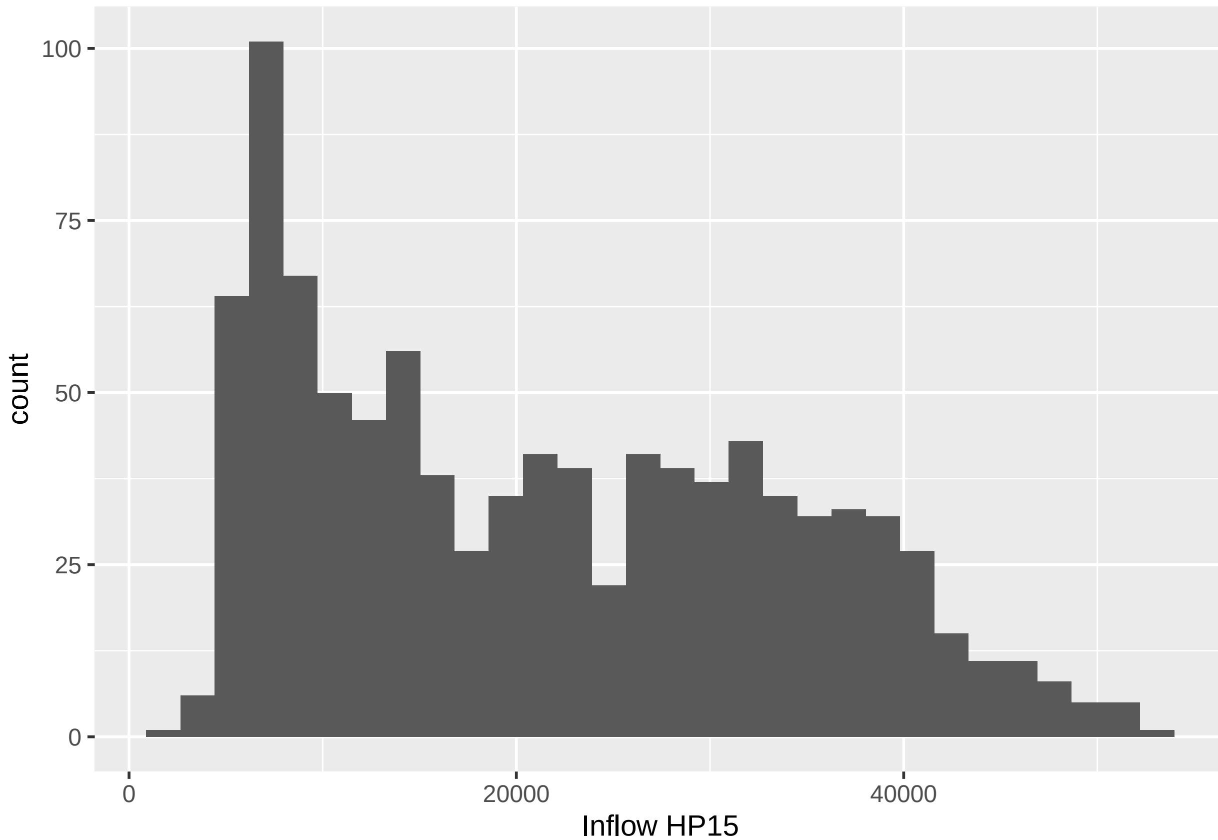

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



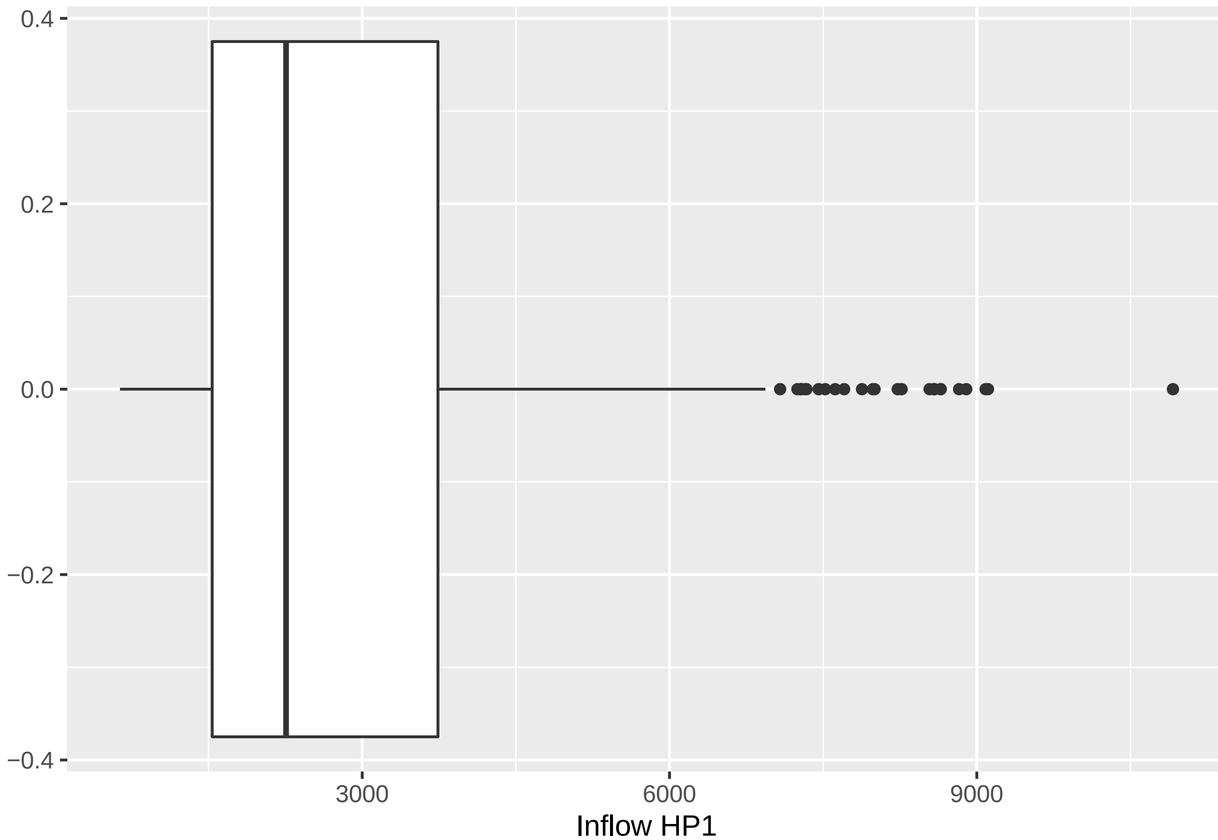

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

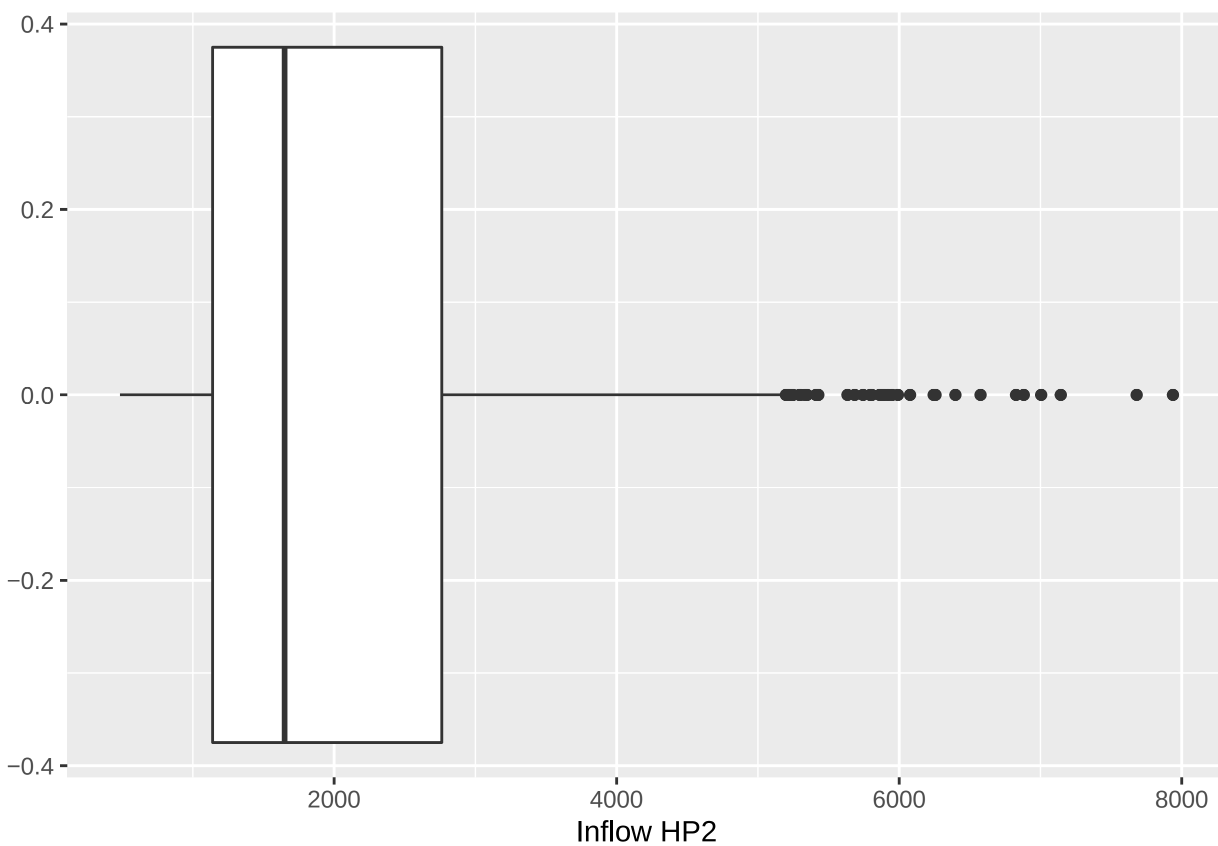


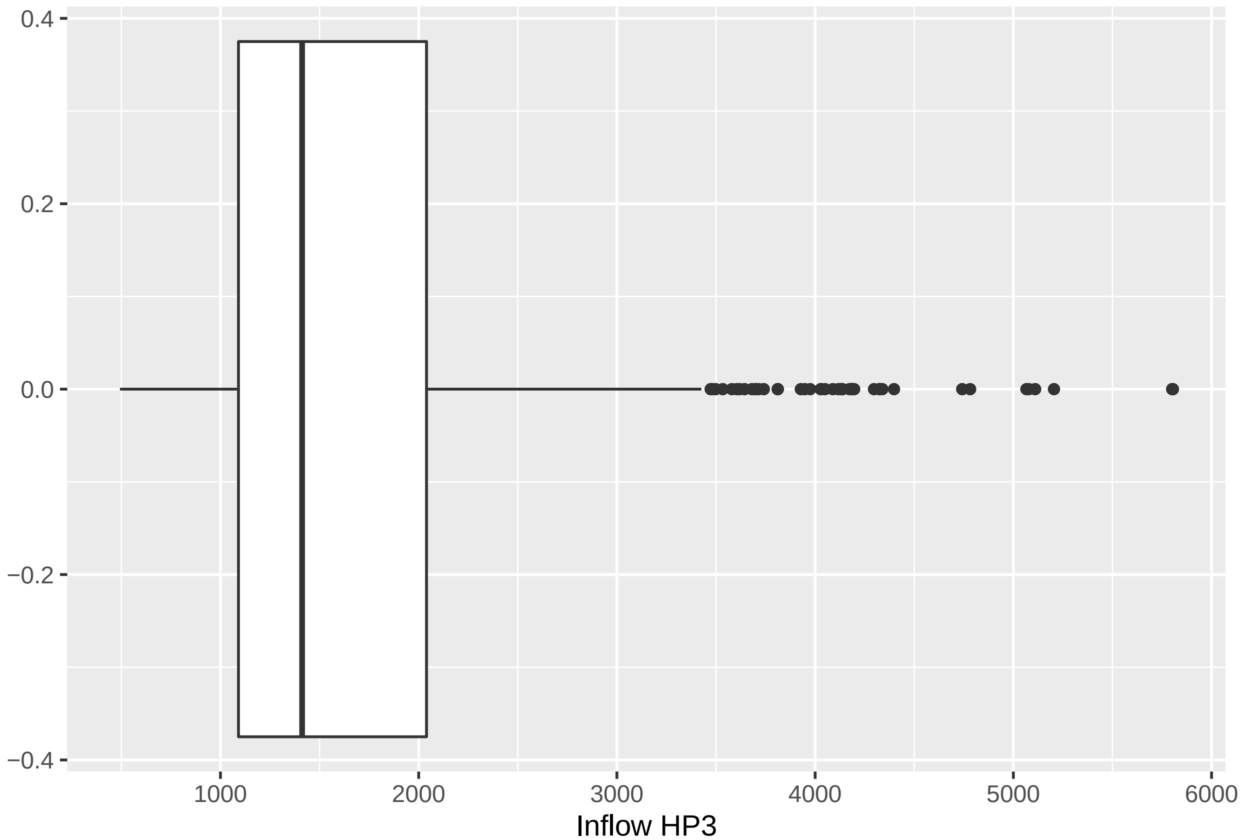

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

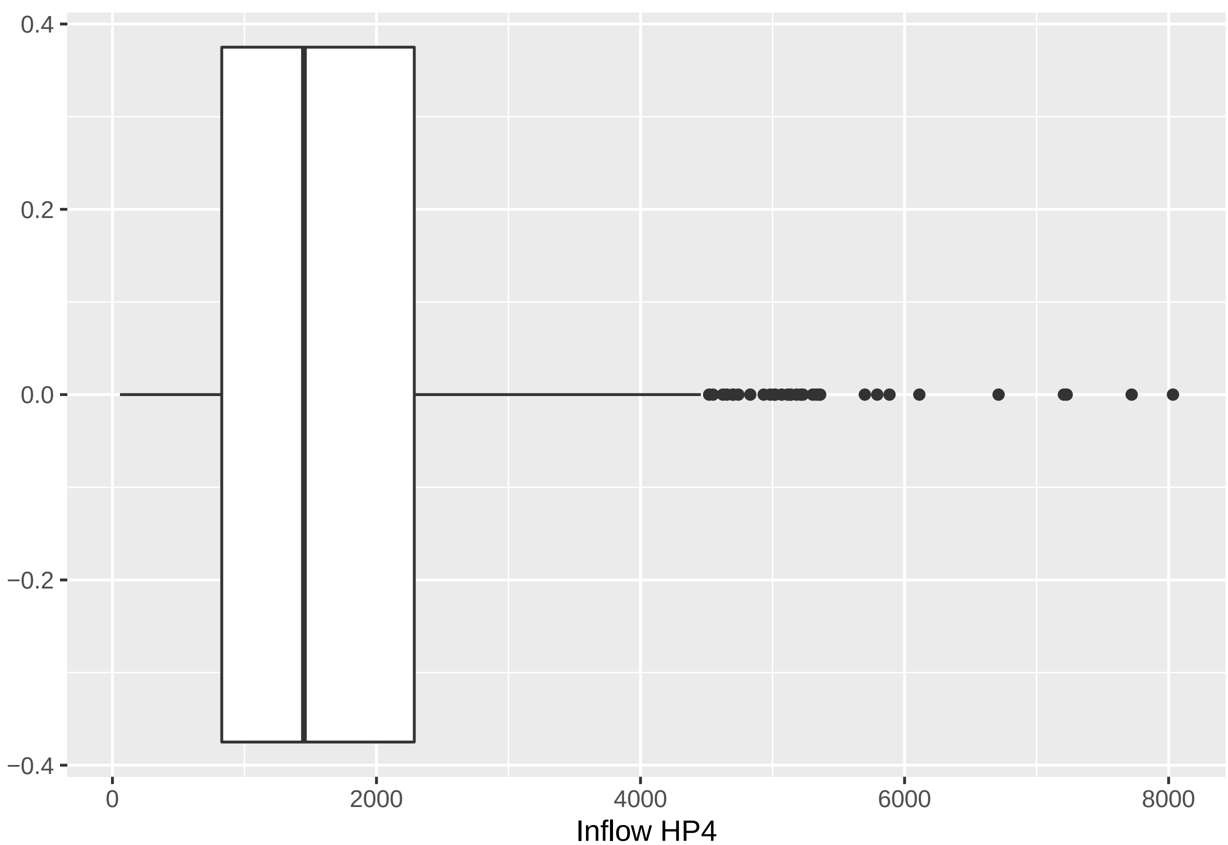


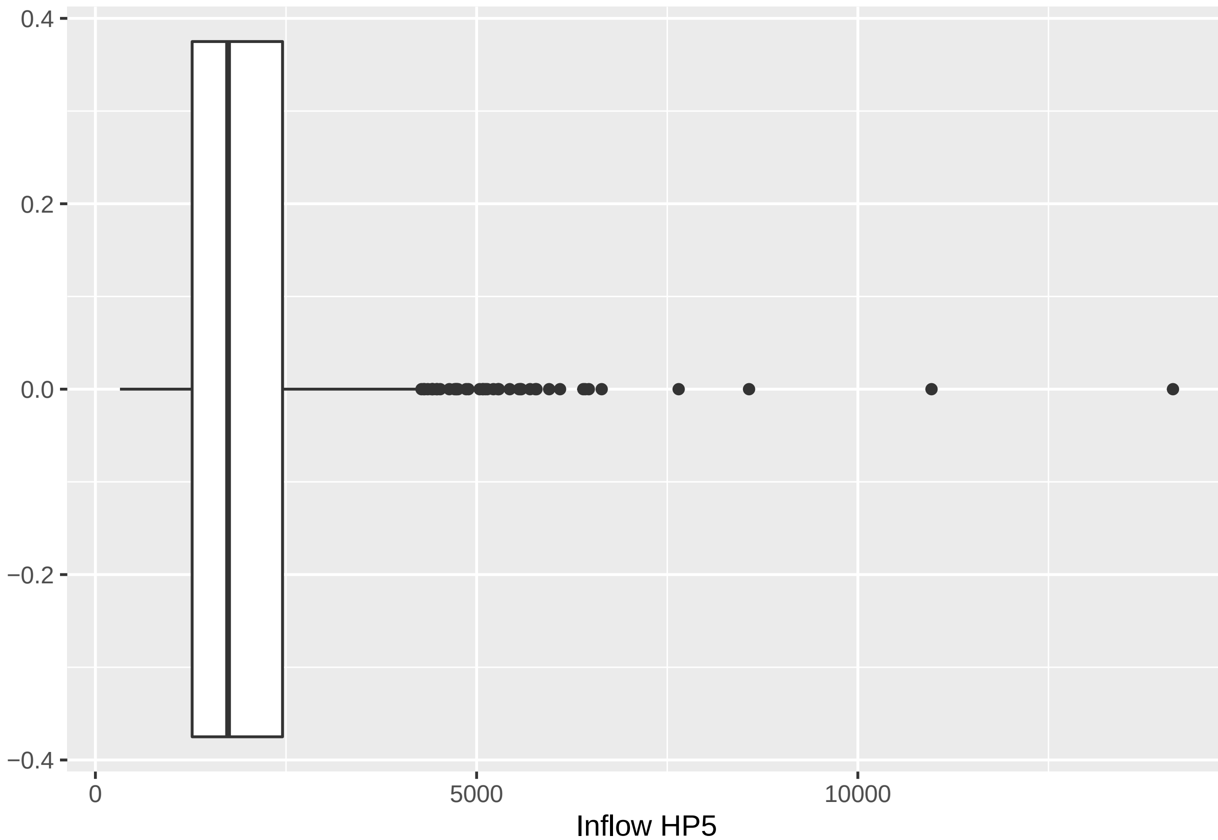

```
#using package ggplot2 to make boxplots
for(i in 1:nhydro){
  print(ggplot(inflow_data, aes(inflow_data[,(1+i)])) +
          geom_boxplot() +
          xlab(paste0("Inflow ",colnames(inflow_data)[(1+i)],sep=""))
        )
}
```


0.4
0.2
0.0
−0.2
−0.4
3000
6000
9000
Inflow HP1

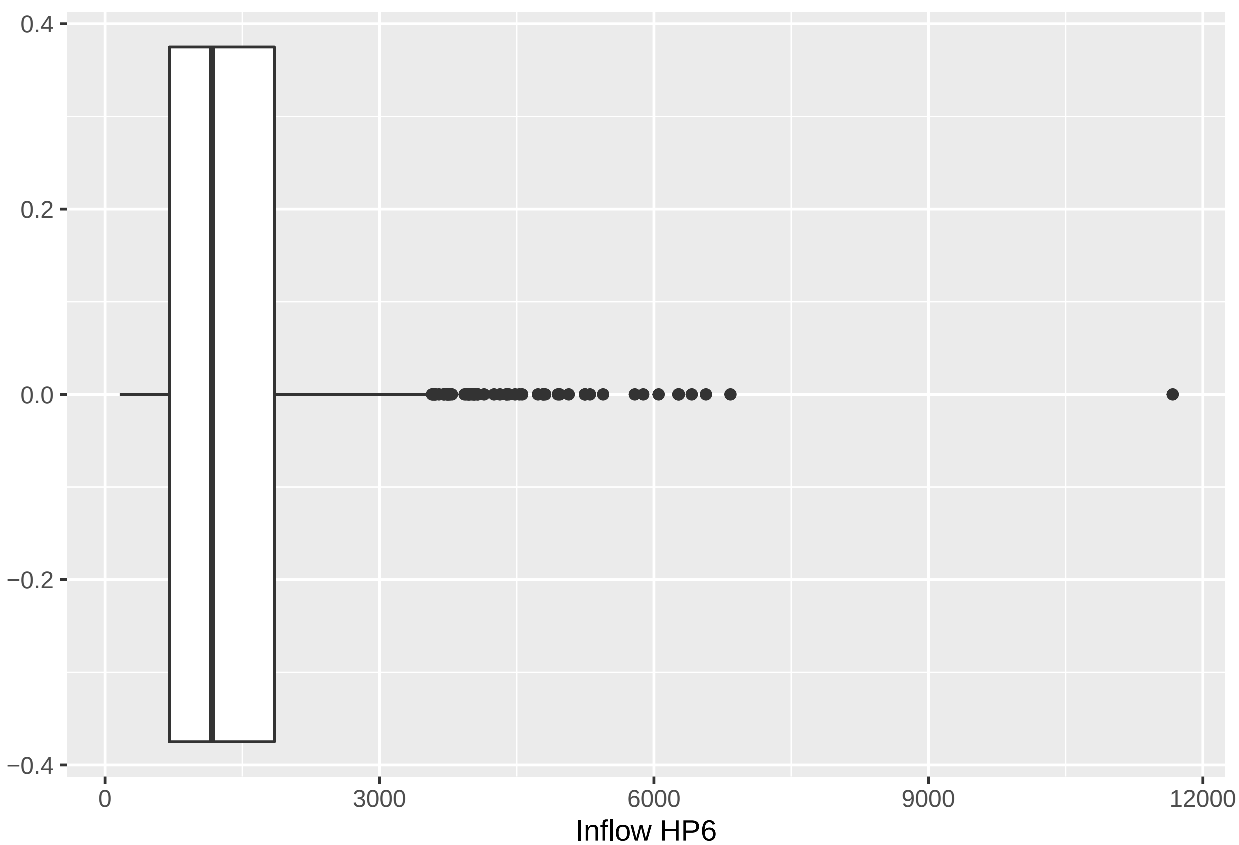

0.4
0.2
0.0
−0.2
−0.4
2000
4000
6000
8000
Inflow HP2


0.4
0.2
0.0
−0.2
−0.4
1000
2000
3000
4000
5000
6000
Inflow HP3

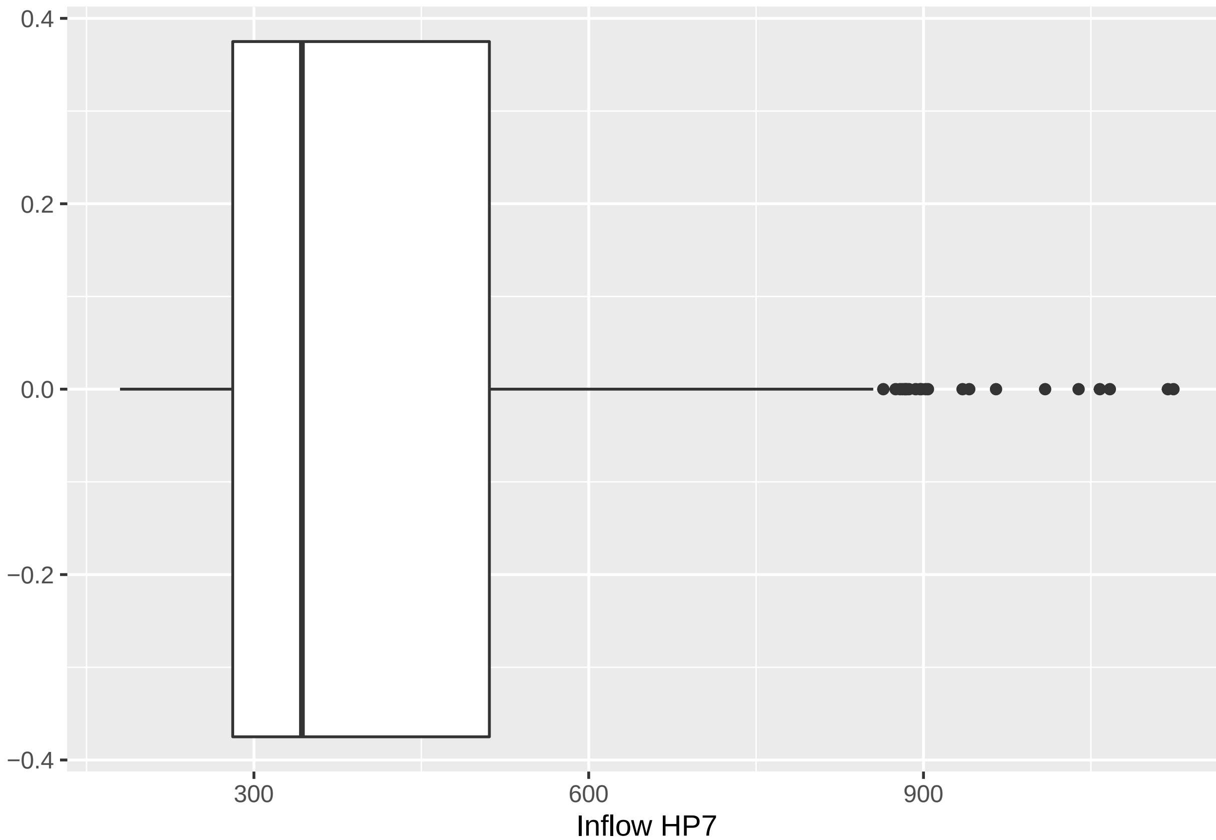

0.4
0.2
0.0
−0.2
−0.4
0
2000
4000
6000
8000
Inflow HP4

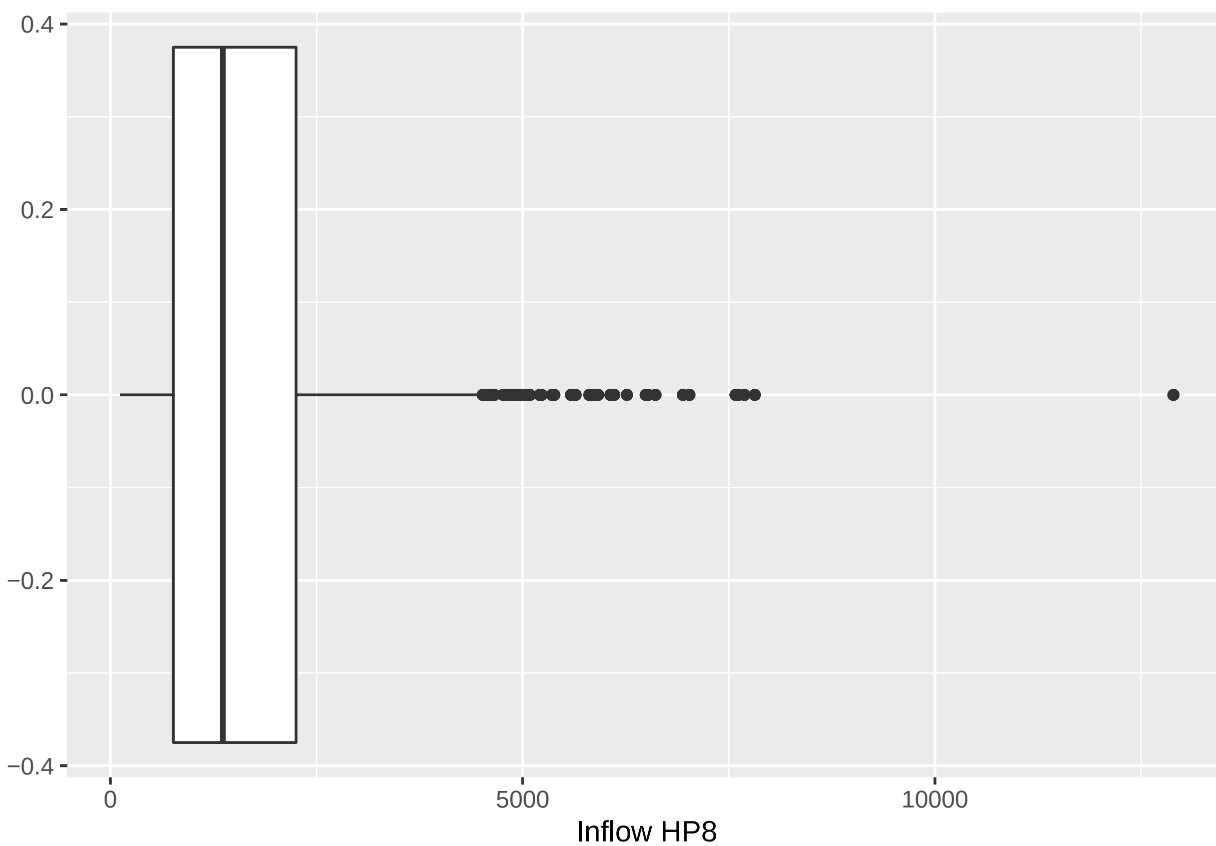


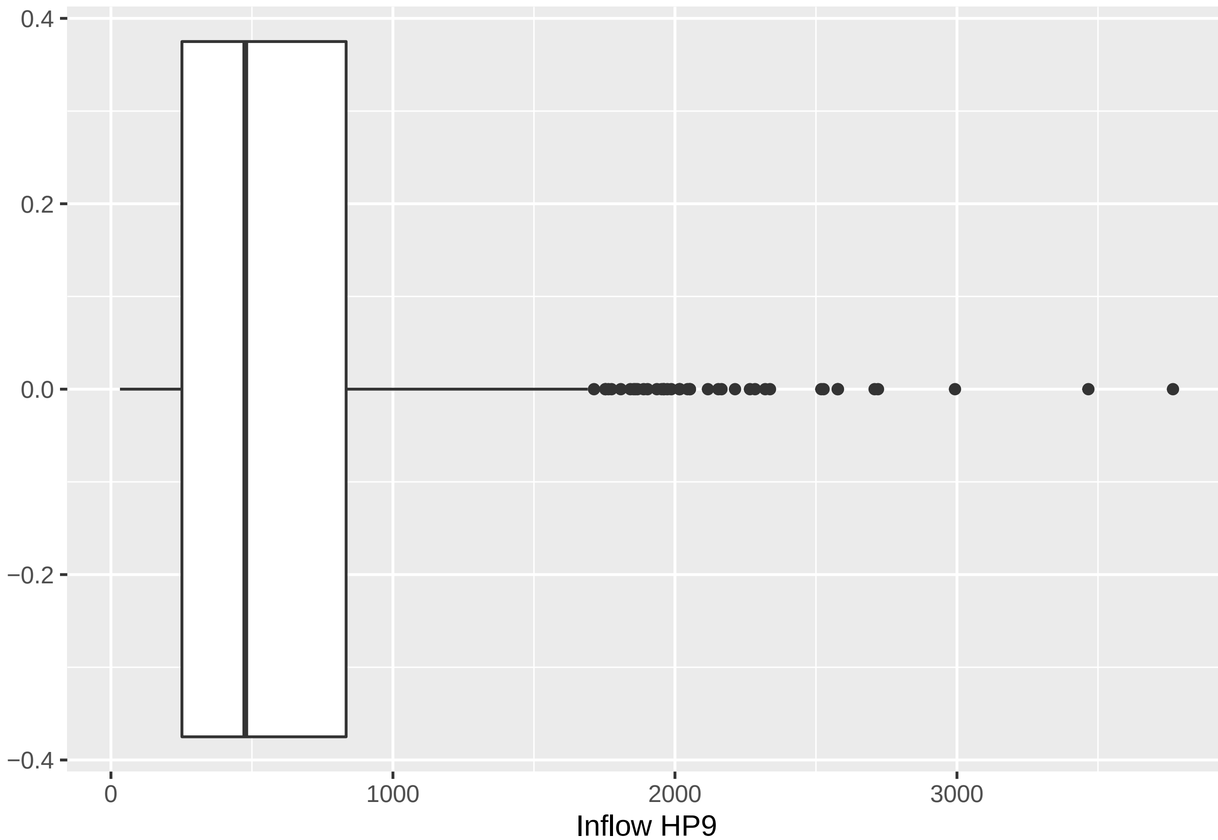
0.4
0.2
0.0
−0.2
−0.4
0
5000
10000
Inflow HP5

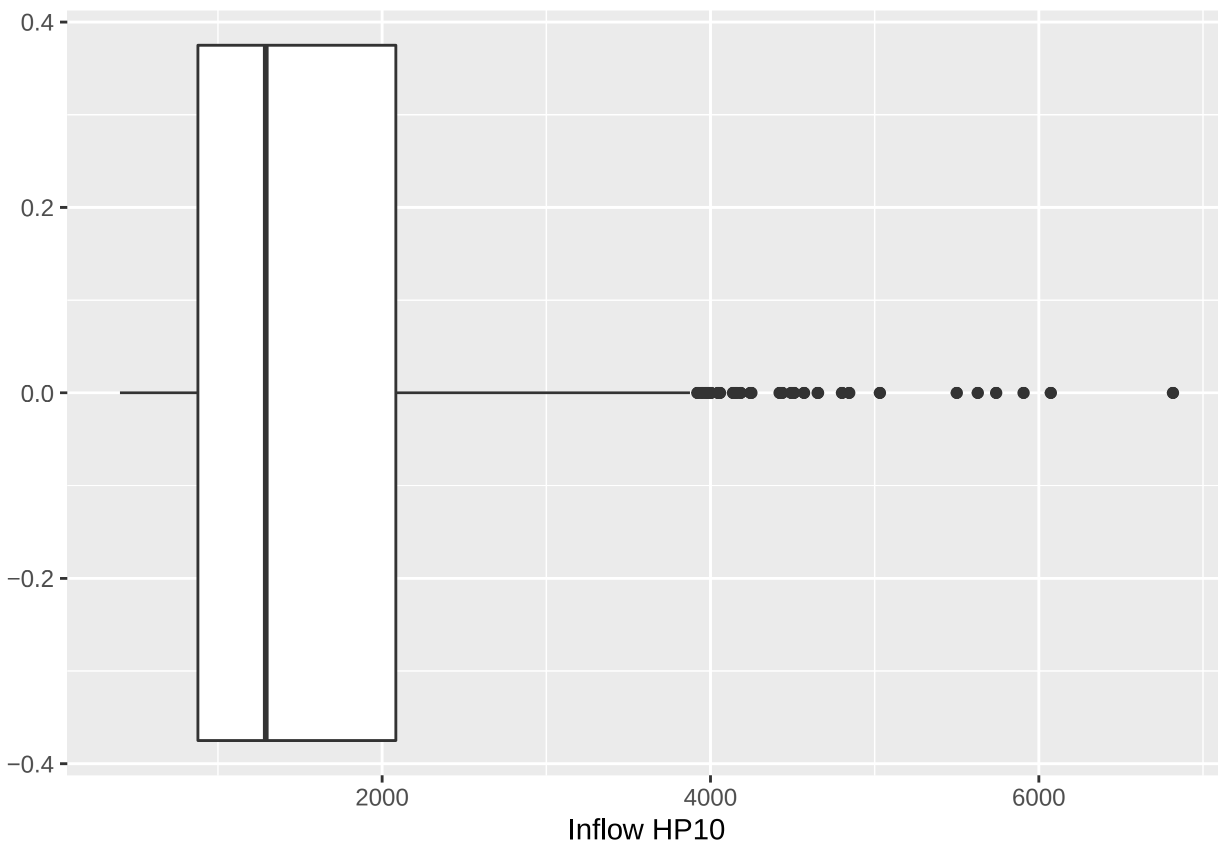


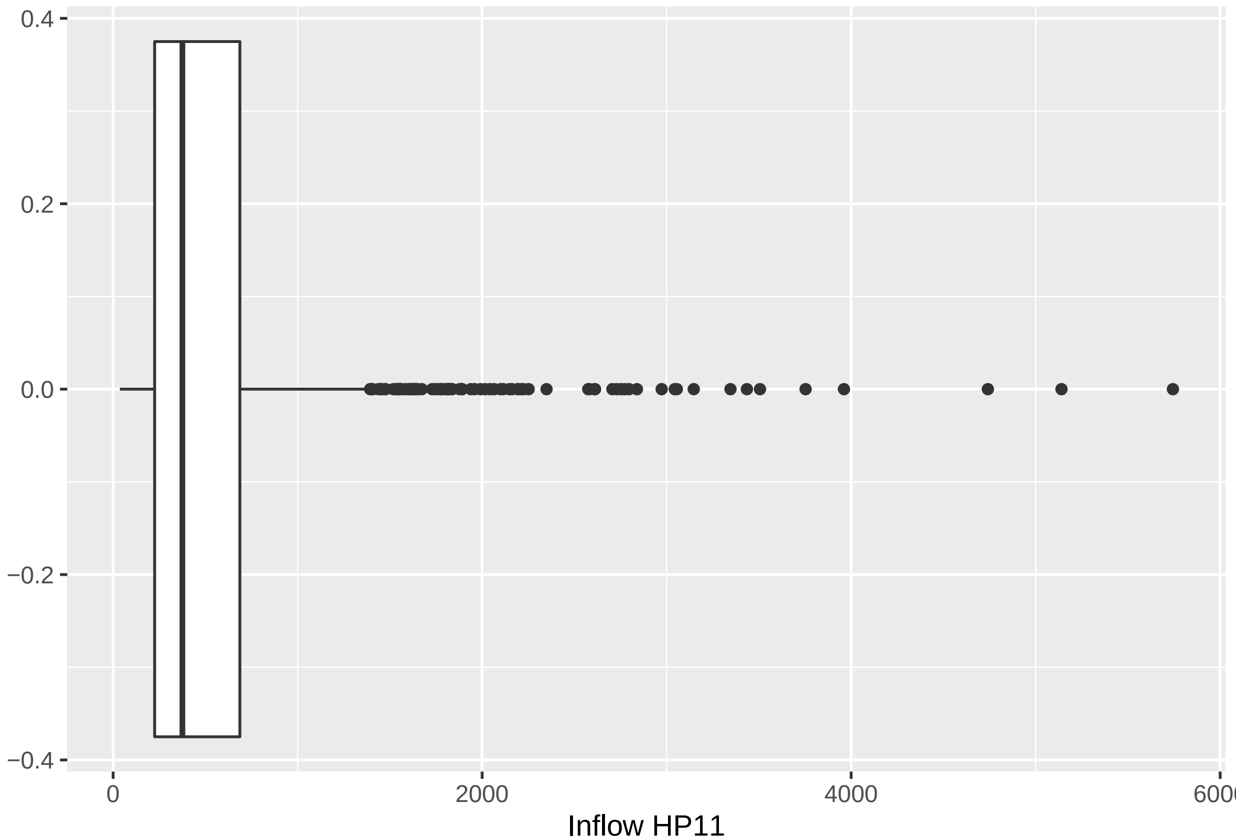
0.4
0.2
0.0
−0.2
−0.4
0
3000
6000
9000
12000
Inflow HP6

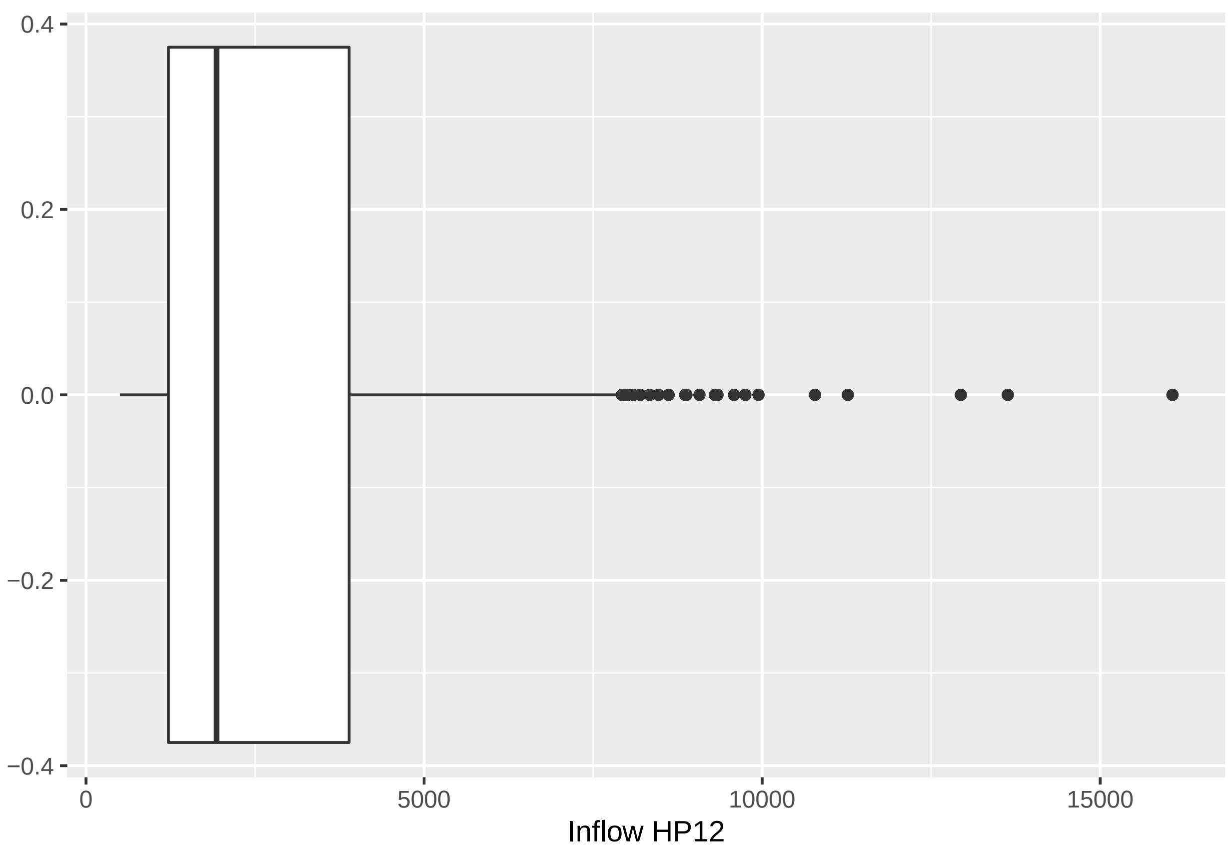

0.4
0.2
0.0
−0.2
−0.4
300
600
900
Inflow HP7

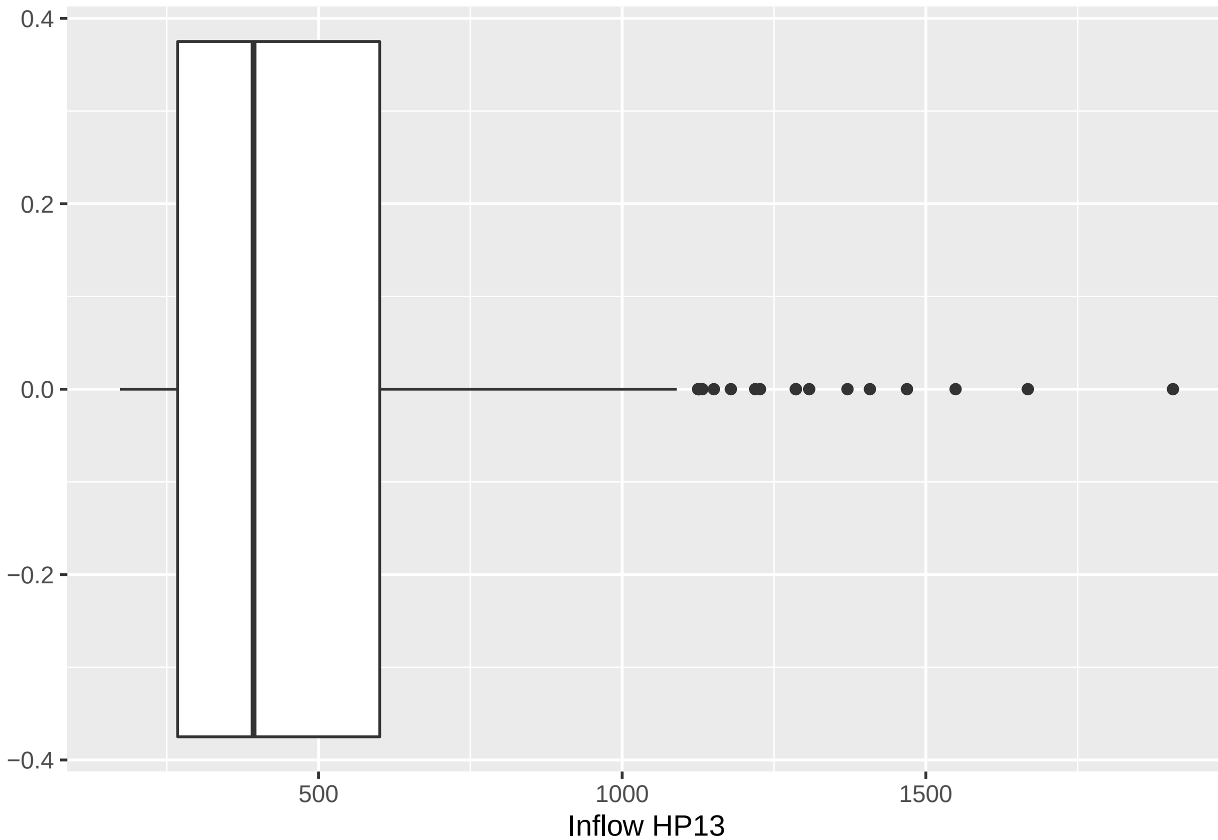

0.4
0.2
0.0
−0.2
−0.4
0
5000
10000
Inflow HP8


0.4
0.2
0.0
−0.2
−0.4
0
1000
2000
3000
Inflow HP9

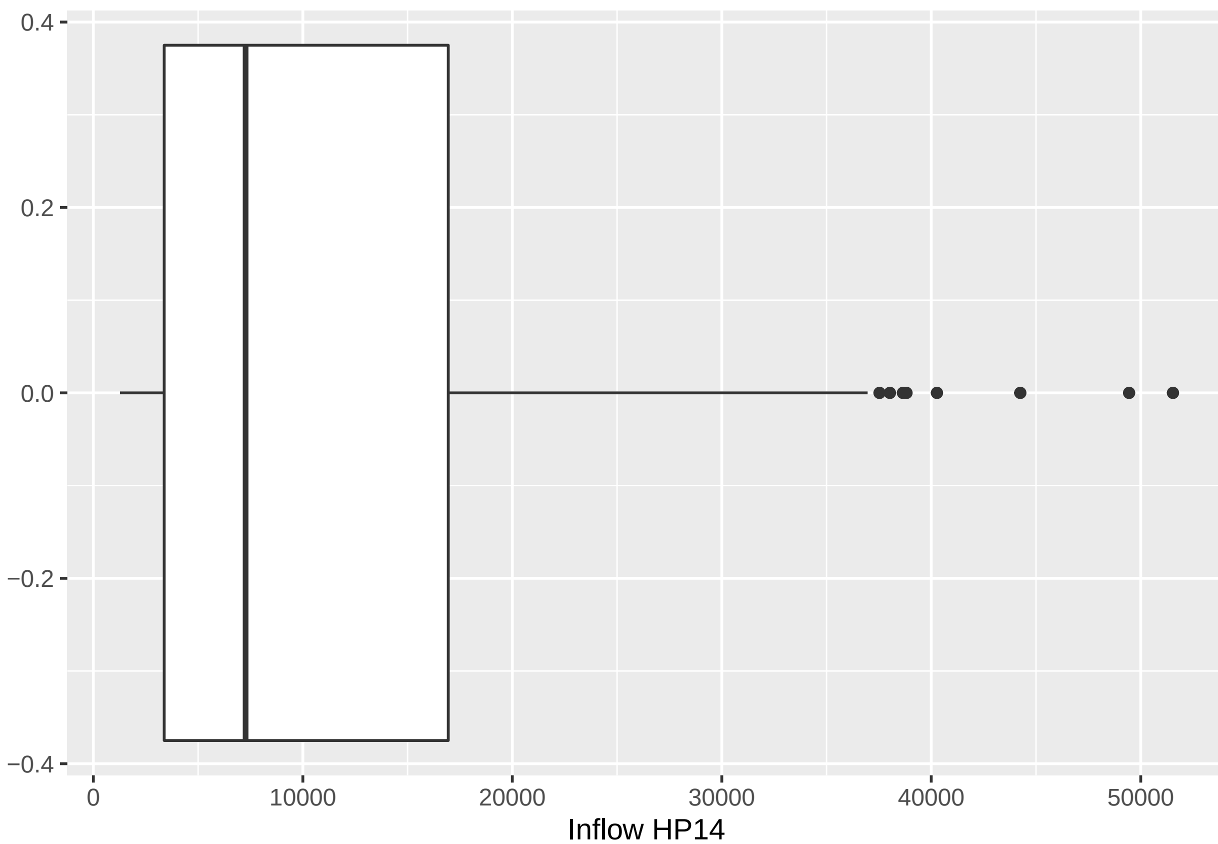

0.4
0.2
0.0
−0.2
−0.4
2000
4000
6000
Inflow HP10


0.4
0.2
0.0
−0.2
−0.4
0
2000
4000
6000
Inflow HP11

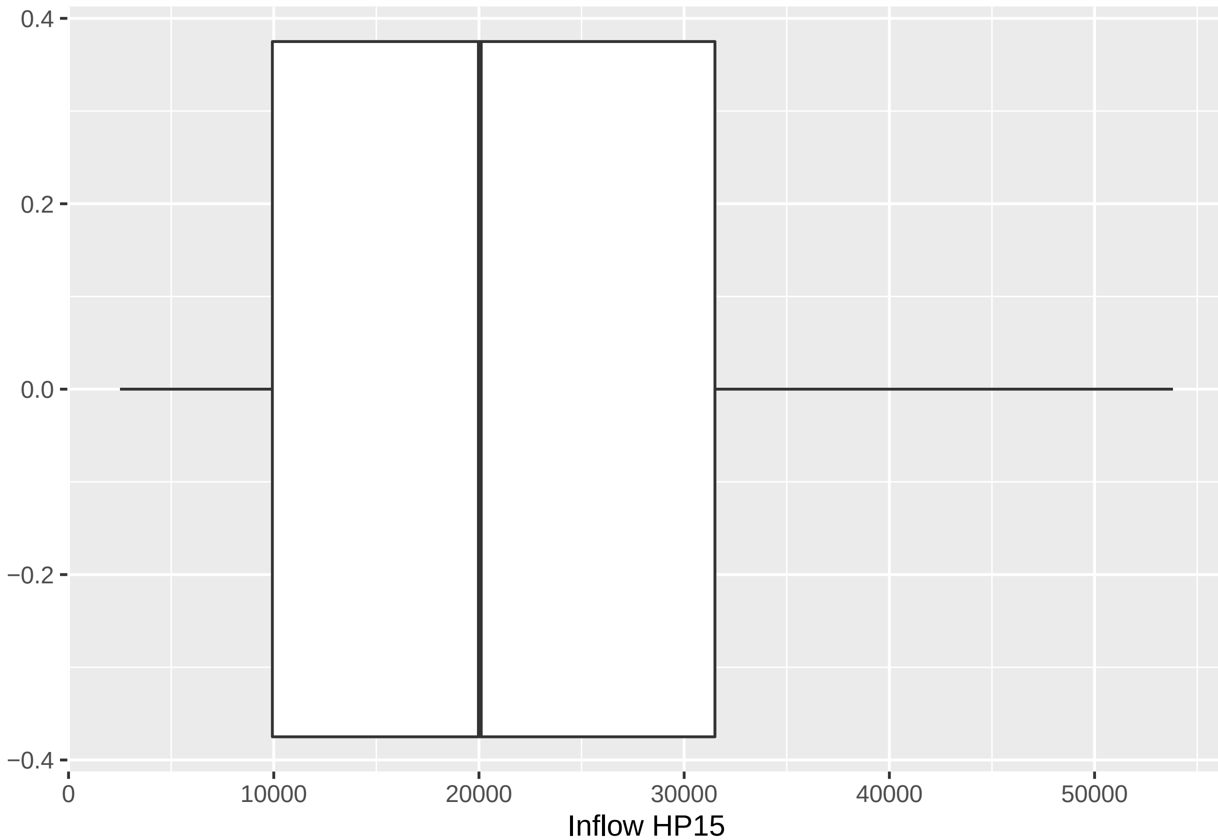

0.4
0.2
0.0
−0.2
−0.4
0
5000
10000
15000
Inflow HP12



0.4
0.2
0.0
−0.2
−0.4
500
1000
1500
Inflow HP13



0.4
0.2
0.0
−0.2
−0.4
0
10000
20000
30000
40000
50000
Inflow HP14



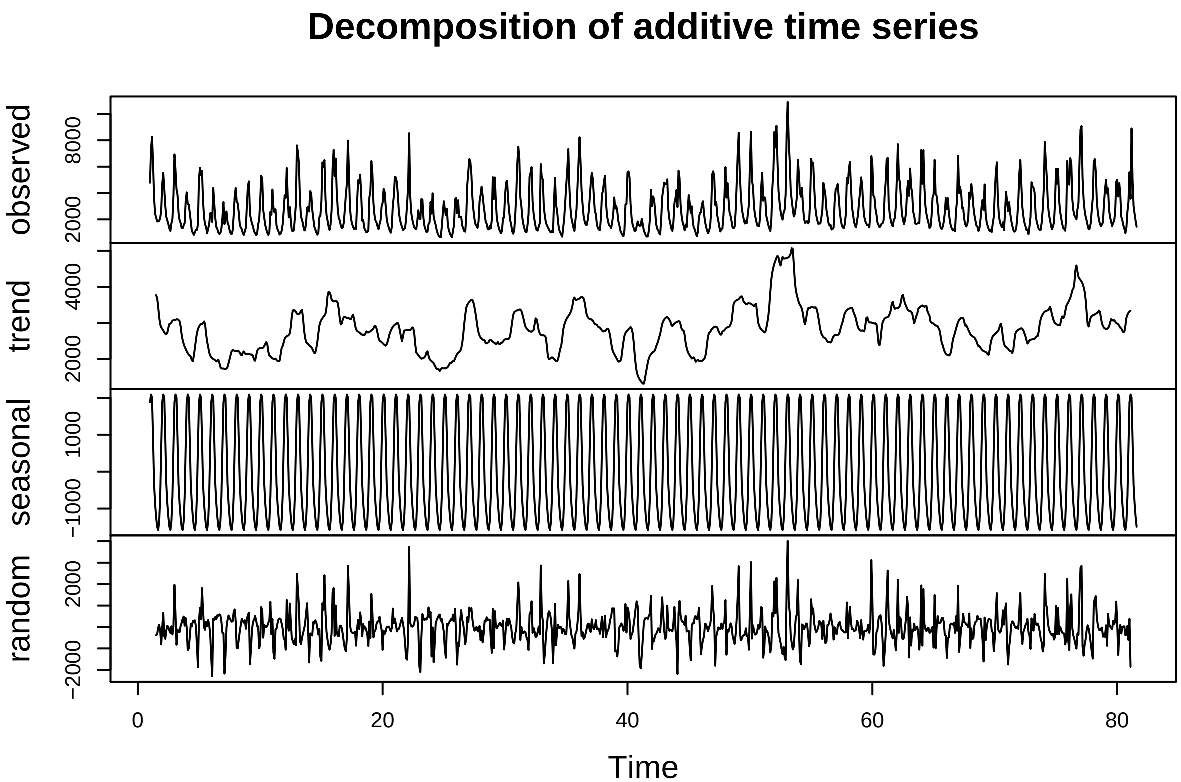
Note from the plots that some reservoirs have many points outside the box. But it's hard to tell if those are outliers or not because we are looking at the time series with all its components.

Decomposing the time series

The stats package has a function called decompose(). This function only take time series object. As the name says the decompose function will decompose your time series into three components: trend, seasonal and random. This is similar to what we did in the previous script, but in a more automated way.

The random component is the time series without seasonal and trend component. Let's try to identify outliers by looking at the random component only.

```
#Using R decompose function
iHP=1
decompose_inflow_data=decompose(ts_inflow_data[,iHP],"additive")
plot(decompose_inflow_data)
```




```
#Inspect random component
inflow_random <- decompose_inflow_data$random
mean_inflow <- mean(inflow_random)
sd_inflow <- sd(inflow_random)

cat(mean_inflow,sd_inflow)
```

```
## NA NA
```

```
#Note random series has some missing values, that is why we got NAs

#Compute mean and standard deviation without missing values
mean_inflow <- mean(na.exclude(inflow_random))  #exclude NA or missing observation to compute mean and
sd_inflow <- sd(na.exclude(inflow_random))

cat(mean_inflow,sd_inflow)
```

```
## -4.839207 764.0217
```

Missing observations

The decompose function introduced NAs in the beginning and end of the data set. Let's just remove them. NAs on the tails can be simply removed.

```
#Create data frame for further use with new random series
inflow_random <- data.frame(date=my_date,month=as.factor(month(my_date)),inflow=as.numeric(inflow_rando

#How many NAs we have, you can get it from summary or using is.na()
sum(is.na(inflow_random$inflow))
```

```
## [1] 12
```

```r
#We have NAs in the beginning and end of data, just remove them
head(inflow_random,10)
```

```
##          date month     inflow
## 1  1931-01-01     1         NA
## 2  1931-02-01     2         NA
## 3  1931-03-01     3         NA
## 4  1931-04-01     4         NA
## 5  1931-05-01     5         NA
## 6  1931-06-01     6         NA
## 7  1931-07-01     7 -382.56004
## 8  1931-08-01     8 -340.76212
## 9  1931-09-01     9  -28.44389
## 10 1931-10-01    10  100.05975
```

```r
tail(inflow_random,10)
```

```
##           date month      inflow
## 959 2010-11-01    11    53.03006
## 960 2010-12-01    12  -607.79650
## 961 2011-01-01     1   380.88423
## 962 2011-02-01     2 -1857.77671
## 963 2011-03-01     3          NA
## 964 2011-04-01     4          NA
## 965 2011-05-01     5          NA
## 966 2011-06-01     6          NA
## 967 2011-07-01     7          NA
## 968 2011-08-01     8          NA
```

```r
#Just remove them
inflow_random <- na.omit(inflow_random)

#Check data again
sum(is.na(inflow_random$inflow))
```

```
## [1] 0
```

```r
head(inflow_random,10)
```

```
##          date month     inflow
## 7  1931-07-01     7 -382.56004
## 8  1931-08-01     8 -340.76212
## 9  1931-09-01     9  -28.44389
## 10 1931-10-01    10  100.05975
## 11 1931-11-01    11 -189.76160
## 12 1931-12-01    12 -813.67150
## 13 1932-01-01     1  -29.36577
## 14 1932-02-01     2  660.34829
## 15 1932-03-01     3 -430.83024
## 16 1932-04-01     4 -552.04754
```

```r
tail(inflow_random,10)
```

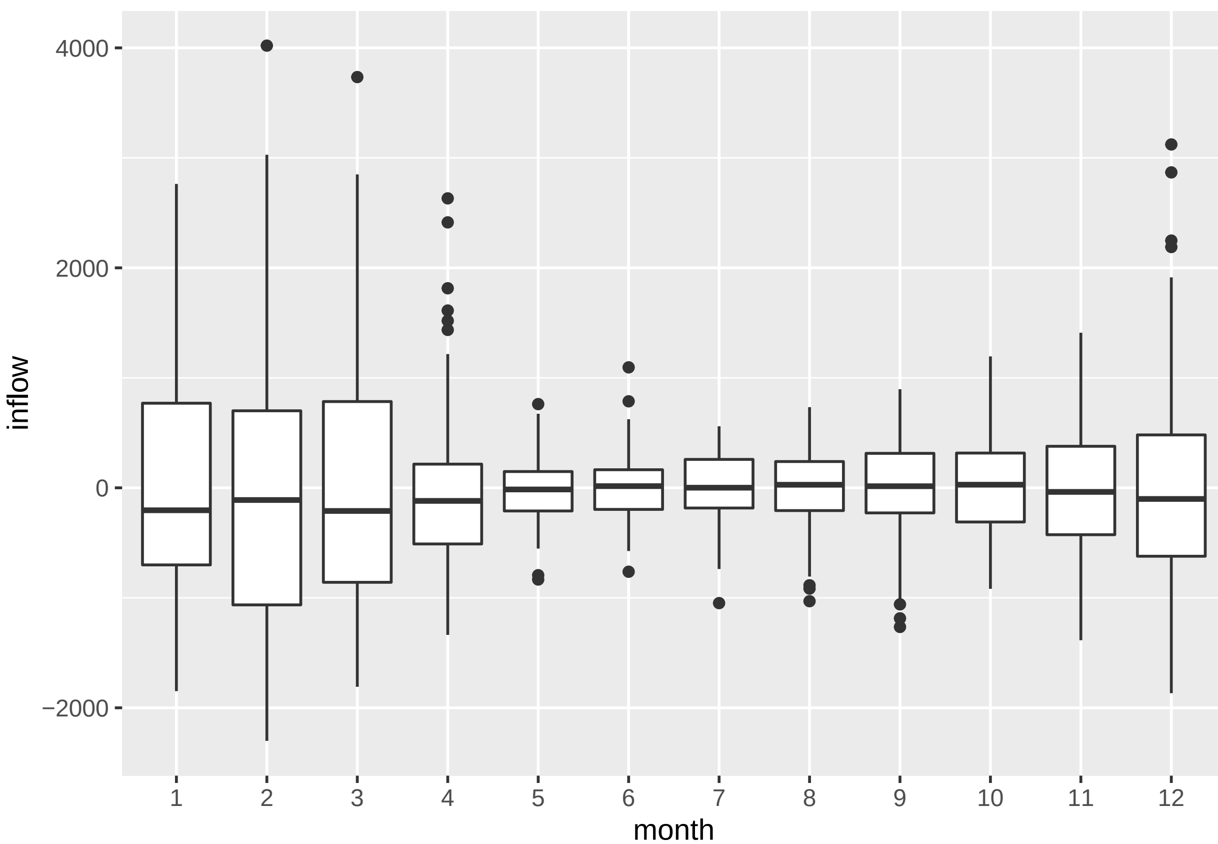
```
##           date month       inflow
## 953 2010-05-01     5  -304.991106
## 954 2010-06-01     6     3.317439
## 955 2010-07-01     7    25.773293
```

```
## 956 2010-08-01     8    26.029543
## 957 2010-09-01     9  -387.860561
## 958 2010-10-01    10  -378.023582
## 959 2010-11-01    11    53.030064
## 960 2010-12-01    12  -607.796499
## 961 2011-01-01     1   380.884231
## 962 2011-02-01     2 -1857.776707
```

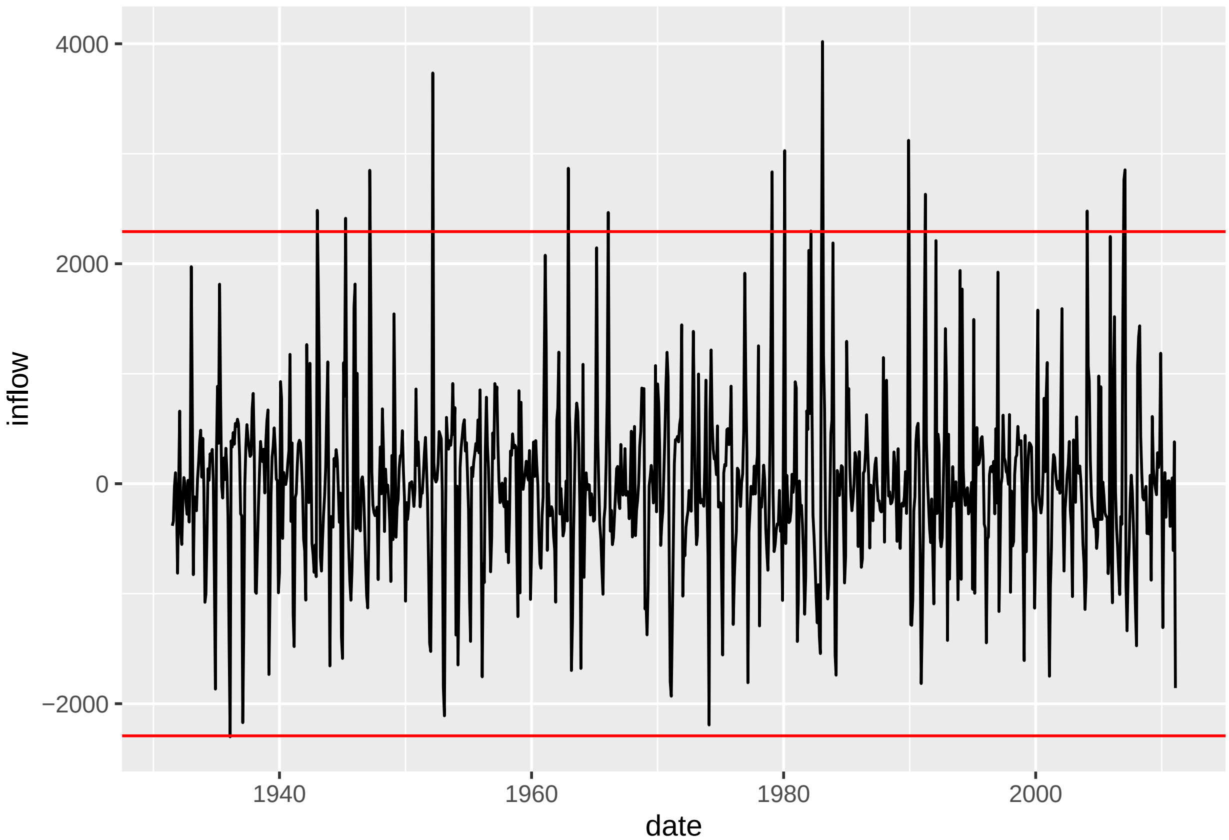
Data is ready!

Visualizing outliers in R

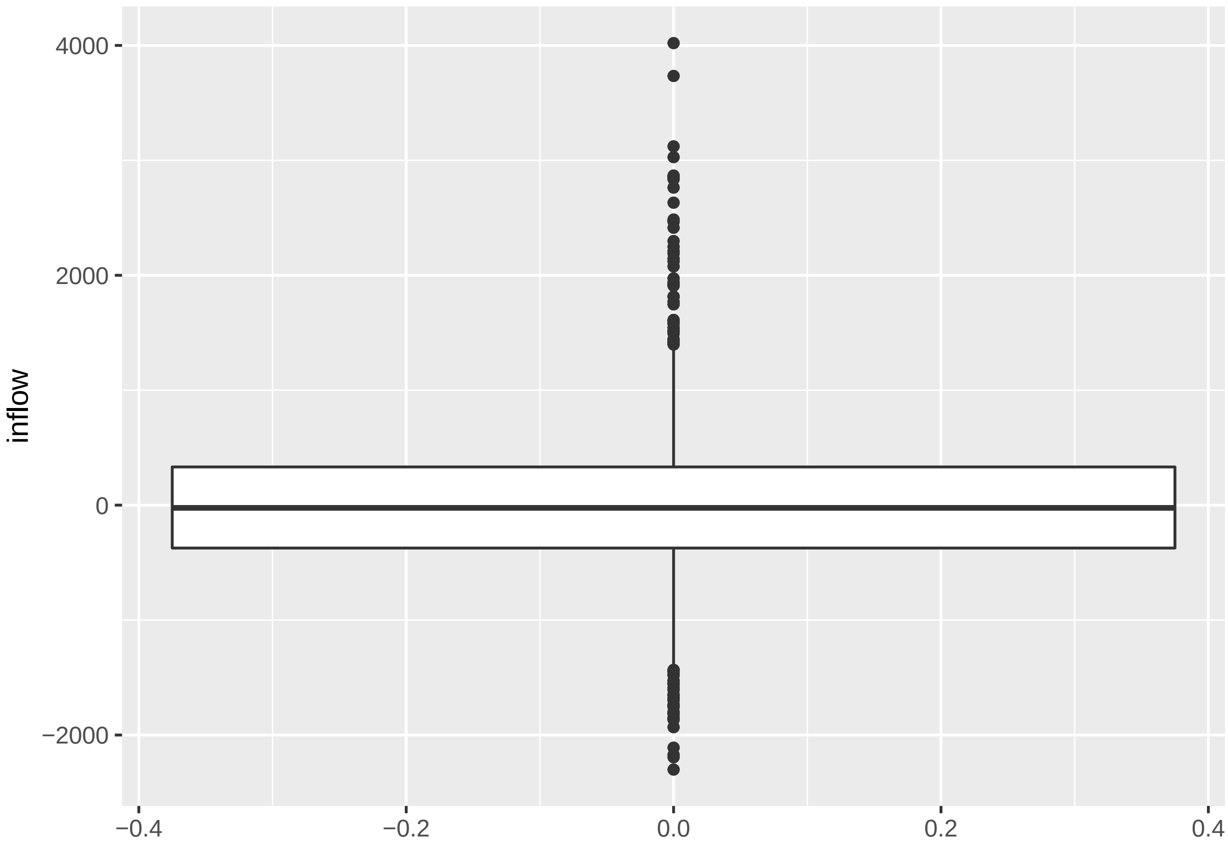
```r
#Generating a box plot by factor where factor is month of the year
ggplot(inflow_random, aes(x=month, y=inflow)) +
            geom_boxplot()
```




```r
ggplot(inflow_random, aes(x=date, y=inflow)) +
            geom_line() +
            geom_abline(slope=0,intercept=3*sd_inflow,color="red") +
            geom_abline(slope=0,intercept=-3*sd_inflow,color="red")
```




```r
ggplot(inflow_random, aes(y=inflow)) +
           geom_boxplot()
```



Since we removed the seasonal and trend component, the mean of the random series should be close to zero. Note that from the line plot with the red lines we see that we do have some outliers. The outliers could be due to error collecting the data or an extreme event. Either way, we may want to remove/replace them before fitting a model to our data set to avoid the effect of outliers on our model coefficients.

The box plots are showing more detailed information about the probability distribution for each month of the year. Note that the same months have larger standard deviations.

Using pre-built functions for outlier detection

We will explore a few function for outlier detection in R.

outlier(): this function identifies the value that deviates the most from the mean, but does not run any statistical test to check if most deviating value is an outlier

chisq.out.test(): this function will check if extreme value is an outlier using hypothesis testing. The null hypothesis for the test is "H0: extreme value not an outlier". Remember to look at p-value to make the decision whether to reject H0 or not.

grubbs.test(): this function will also check if extreme value is an outlier using hypothesis testing. The null hypothesis for the test is "H0: extreme value not an outlier". Remember to look at p-value to make the decision whether to reject H0 or not.

rm.outlier(): if the result from the chi test tells you the extreme value is an outlier, then you can use this function to remove it or replace by sample mean or median.

When working with time series you cannot simply remove an outlier. Remember that in TSA we care about the time dependence structure, therefore eliminating observations is not an option. Instead we replace it with another value - preferably the local mean.

```
#Just find extreme value
outlier(inflow_random$inflow)
```

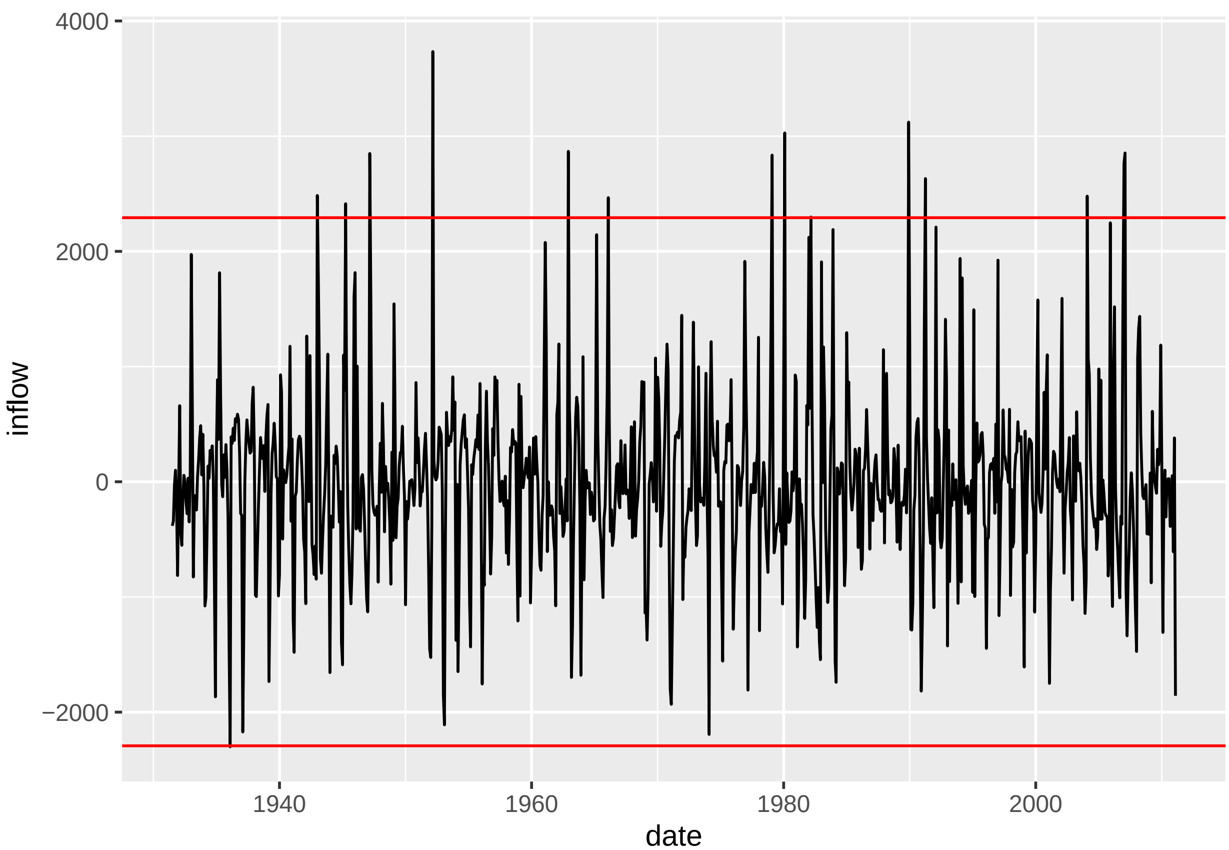
```
## [1] 4020.223
```

```
#Function chisq.out.test check if extreme value is outlier
chi_test <- chisq.out.test(inflow_random$inflow,var(inflow_random$inflow))
print(chi_test) #look at the p-value to find the decision
```

```
##
##  chi-squared test for outlier
##
## data:  inflow_random$inflow
## X-squared = 27.755, p-value = 1.377e-07
## alternative hypothesis: highest value 4020.22329300721 is an outlier
```

```
#If you need to remove outlier use rm.outlier()
inflow_random$inflow <- rm.outlier(inflow_random$inflow,fill=TRUE) #using fill equal true the value wil
#Since we removed seasonality replacing with overall mean instead of local mean is acceptable

#Plot series again and look for more outliers
ggplot(inflow_random, aes(x=date, y=inflow)) +
            geom_line() +
            geom_abline(slope=0,intercept=3*sd_inflow,color="red") +
            geom_abline(slope=0,intercept=-3*sd_inflow,color="red")
```



Note we sill have some outliers.

You can repeat the produce until the next extreme value is not an outlier or write a loop as below.

```r
summary(inflow_random$inflow)
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -2300.860  -373.265   -23.874    -9.054   331.148  3734.378
```

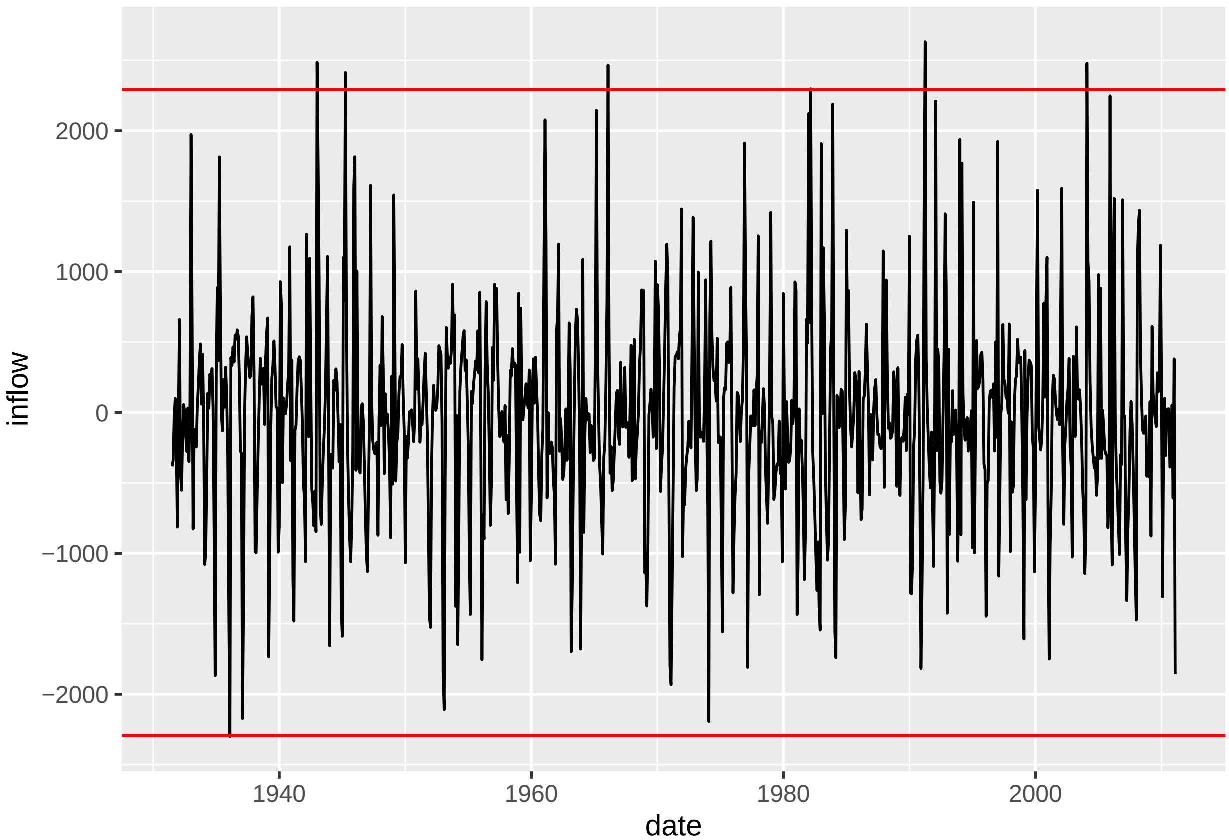
```r
#Writing a loop to remove all outliers
#Loop while to remove all outliers
pvalue <- 0 #just making sure we enter the while loop
aux_inflow <- inflow_random$inflow  #Create a new vector for inflow_random just to make sure we don't l
nout <- 0 #keep track of number of outliers removed
while(pvalue < 0.05){ #the algorithm only enter the loop if the p-value
                 #of first chi_test is less than 0.05 i.e. if there
                 #is an outlier that needs to be removed
  out_test <- grubbs.test(aux_inflow,type=10)
  pvalue <- out_test$p.value   #Update p-value every time we run the test for a new Aux_Y

  if(pvalue < 0.05){
    aux_inflow <- rm.outlier(aux_inflow,fill=TRUE) #replacing outliers
    nout <- nout+1
  }
}
cat("Number of outliers removed: ",nout,"\n")
```

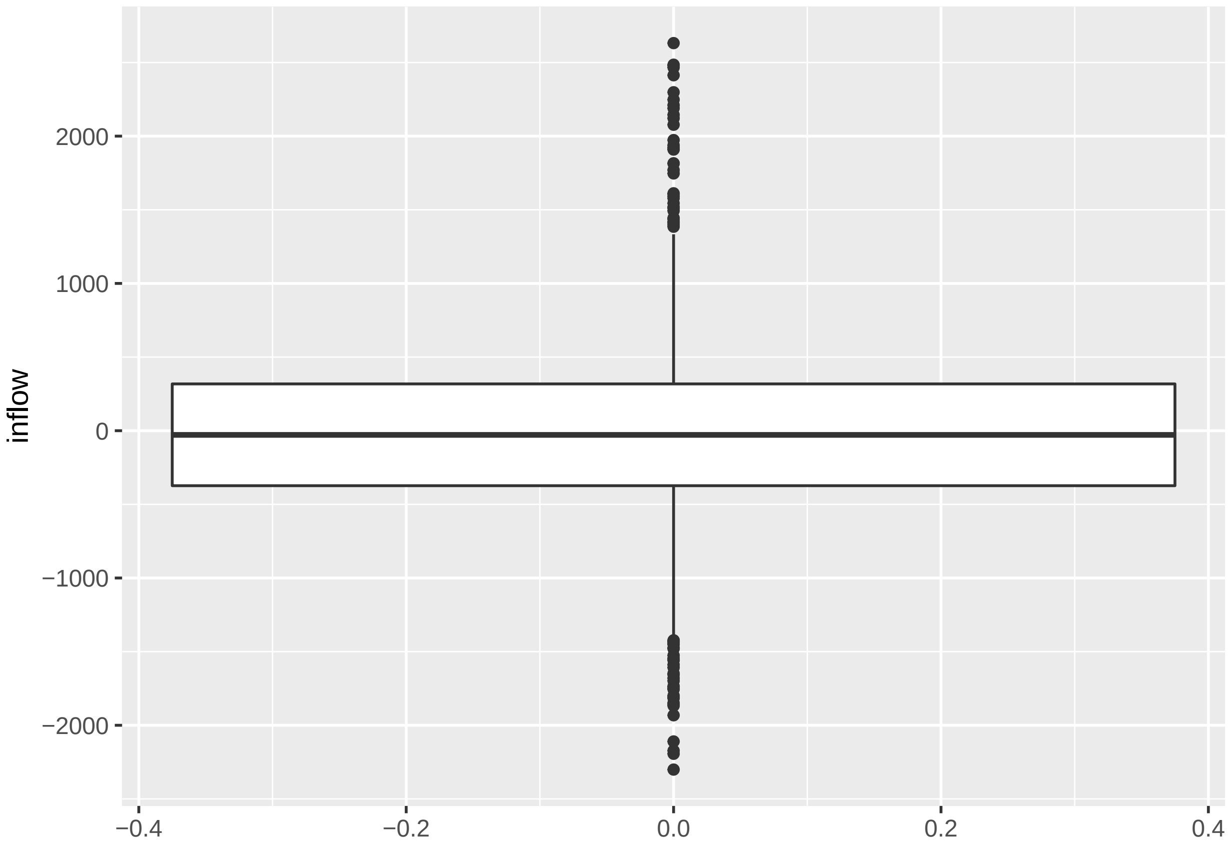
```
## Number of outliers removed:  8
```

```r
#Replaced original data with data without outliers
inflow_random$inflow <- aux_inflow

#Do the plots again
ggplot(inflow_random, aes(x=date, y=inflow)) +
            geom_line() +
            geom_abline(slope=0,intercept=3*sd_inflow,color="red") +
            geom_abline(slope=0,intercept=-3*sd_inflow,color="red")
```




```
ggplot(inflow_random, aes(y=inflow)) +
            geom_boxplot()
```




```
#Check the data
summary(inflow_random$inflow)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2300.86  -373.26   -28.45   -34.42   317.85  2631.74
```