Examining the Hydrologic Properties of the Missouri River Basin

https://github.com/cwatson1013/Hydrologic_Data_Analysis_Final_Proj

Rachel Bash, Keqi He, Caroline Watson, and Haoyu Zhang

Abstract

The Missouri River provides critical water resources that drives the region's agriculture, industry, and ecosystems. This is a region that experiences surface water variability, characterized by damaging floods and severe droughts, greatly impacting the agricultural production of the area. It is reported that a serious flood disaster occurred in the lower Missouri River in the spring of 2019 and the Missouri River experienced severe drought in 2012-2013. This project highlights the changes in stream flow and water quality over time, and identifies key characteristics of the river. Twenty two sites across the lower Missouri River Basin were examined in order to get a fuller picture of the Missouri River and its tributaries over time. By analyzing the trend of the Missouri River discharge, we can predict future changes in the Missouri River flow to provide a reference for water resources management. In addition, we focus on the stream flow and water quality of Missouri River during March to July, 2019 to see how discharge influence the water quality and what can be done to keep the water in the Missouri River in good quality.

Contents

List of Tables

List of Figures

1 Research Question and Rationale

The Missouri River is the largest river in North America (2,540 miles) and has the second largest watershed (529,000 mi^2/339 acres, U.S.-Canada). Its watershed covers portions of ten states, which account for approximately one-sixth of the continental United States, as well as a small part of Canada. The headwater is located in the Bitterroot Mountains River of northwestern Wyoming and southwestern Montana. The watershed is home to around 12 million people in 1990, and has been inhabited by indigenous people for millennia. Demands for managing the river for the benefit of human livelihood has resulted in drastic modification in the river and the floodplains. Numerous reservoirs and dams have been constructed, of which six major dams were built on the mainstream, following the Pick-Sloan Plan in 1944. Now, the river is used intensively in multiple ways, including municipal, agricultural, hydropower, recreation, flood control etc.

Within the 328 million acres of the basin's total area in the United States, 95% is related to agricultural uses, while the rest dedicated for recreation, fish and wildlife, and urban. More than half of the total is pasture and range grassland primarily for grazing, and cropland consists of almost 104 million acres, which is 32% of the whole basin. Irrigated land comprises 7.4 million acres, and 6.9 million acres are intensively cropped. Water bodies, on the other hand, cover 3.9 million acres. In spite of the low proportion of water areas (1.2%), they are the pivotal foundation for agricultural or other usages, and thus critical to the whole region's economy.

Along with the agricultural, urban, and industrial development in the region is nutrient loading and enrichment in water bodies, especially for nitrogen (N) and phosphorus (P). Unlike other regions, agricultural input through fertilizer is the predominant anthropogenic source for nutrient in water bodies in the whole basin. Regardless of the major anthropogenic source, nutrient enrichment is considered nationally as one of the leading factors for water quality impairment. According to USEPA 303(d) lists, more than 160 stream reaches, lakes, or reservoirs were reported by USEPA to be nutrient-related impaired in 2006.

In addition to change in nutrient concentration, discharge appears to be highly variable in the basin, and both severe drought and flooding events occurred in the basin in the past. For example, in the spring and summer of 2011, an unprecedented flooding event caused over $2 billion damage FEMA disaster declaration was made in all states along the Missouri River. Subsequently, in 2012, a drought even struck the Central Great Plain, including the basin, and inflicted at least $12 billion of loss before July, 2012. Recently, another flooding event occurred in the spring of 2019.

Given all the background information above, we would like to know the current state of Missouri River and its tributaries, with a focus on the changing pattern in discharge and nutrient. We are interested in how the dramatic change in discharge (i.e. water quantity) could potentially interact with nutrient enrichment (i.e. water quality). Also, we examined a few specific flooding events, during which changes in both water quality and quantity were well recorded, so that we could make concrete inference on the interplay between quantity and quality. Finally, based on the pattern in the past and the best model we could fit, we attempted to predict the likely future conditions and trends in the Missouri River Basin.

2 Dataset Information

The data we are analyzing comes from the United States Geological Survey (USGS) database called the National Water Information System interface, or NWIS. We pulled data from the interface using the R package `dataRetrieval`. Because we are interested in the lower Missouri River basin, we pulled sites from each HUC4 subbasin from 1020 to 1030 (see Figure below). We chose these subbasins because they had a variety of tributaries that all flowed into the Missouri River, and we wanted a variety of river sizes and lengths. We filtered these subbasin queries to only show us sites that had discharge, nitrogen, and phosphorus data. Once we found the sites with all of this data, we chose 2 sites from each HUC sub basin as our 22 "best sites". Our best sites had the overall best time period range for all of our "must have" variables. We retrieved data on water quantity, water quality (N, P concentrations), pH, and coliform concentrations. @ref{tab:table} illustrates the variables we examined and the number of sites in our area of interest that had quality data for each.

Only seven sites within our HUC subbasin boundary contained any high frequency discharge and nitrogen data. Therefore, we also looked at these 7 sites in order to do analyses and answer our research question about flooding.

After doing initial data wrangling and analysis on our 22 "best sites", we decided to pare it down further and only do subsequent analyses on **10** sites. While we initially wanted to look at many sites that were varied in size and location, we determined that it was too many to look at and draw relevant conclusions from.

We have three main datasets:

- The daily values dataset with our 22 "best sites"
- The water quality dataset with our 22 "best sites", with only six sites that had total coliform data.
- The high freqency dataset with 7 sites that contain both high frequency discharge and high frequency nitrogen data.

Variables	Units	NumOfSites
discharge	cfs or m3/s	22
time	UTC	22
nitrogen	mg/L	22 daily values, 7 high frequency
pH	1	22
total coliform	cfu/100mL	6
phosphorus	mg/L	22

3 Exploratory Data Analysis and Wrangling

3.1 Water Quality Data

There were 6 sites in the Missouri River Basin that had total coliform data in addition to total nitrogen, total phosphorus, pH, and discharge data. Out of the 6 sites, only *3* of the sites were chosen for water quality analysis because they had the most data for all of the parameters for water quality. The water quality data was wrangled to include certain columns necessary for the analysis. The sites that were looked at in depth for water quality analysis are:

- 06810000
- 06856600
- 06934500

Water quality parameters were analyzed to determine the quality of river water in the Missouri River Basin. TN and TP were analyzed because the area in the Southern Missouri River Basin we were looking at is largely agricultural land. Total coliform was analyzed because it is known to indicate poor water quality standards in rivers. TC concentrations can be elevated, especially if near agricultural land. However, our data does not supply enough information to draw conclusions on total coliform levels in the rivers of our chosen sites.

```
#filtering water quality dataset to include only 3 sites
bestsites.wq.skinny <- bestsites.wq %>%
    select(Site = site_no,
              Date = Date,
              Parameter = parm_cd,
              Value = result_va,
              Discharge = X_00060_00003) %>%
    group_by(Date, Parameter, Site) %>%
    summarize(Value = mean(Value),
              Discharge = mean(Discharge)) %>%
    spread(key = Parameter, value = Value) %>%
    rename(pH = '00400', total.coliform = '31501',
           Discharge2 = '00060', total.nitrogen = '00600',
           total.phosphorus = '00665') %>%
    mutate(Year = year(Date)) %>%
    select(-Discharge2) %>%
    filter(Site == "06810000" | Site == "06856600" |
           Site == "06934500")
```

Graphs were made to look at total phosphorus, total nitrogen, pH, and total coliform over time. These figures were also faceted by site to see whether there were trends in specific sites.

As seen by Figure 1, total phosphorus values have a slight positive trend. From Figure 1, site 06934500 has the most total phosphorus data.

From Figure 2, total nitrogen looks as though there is a slight positive trend from 1980 to

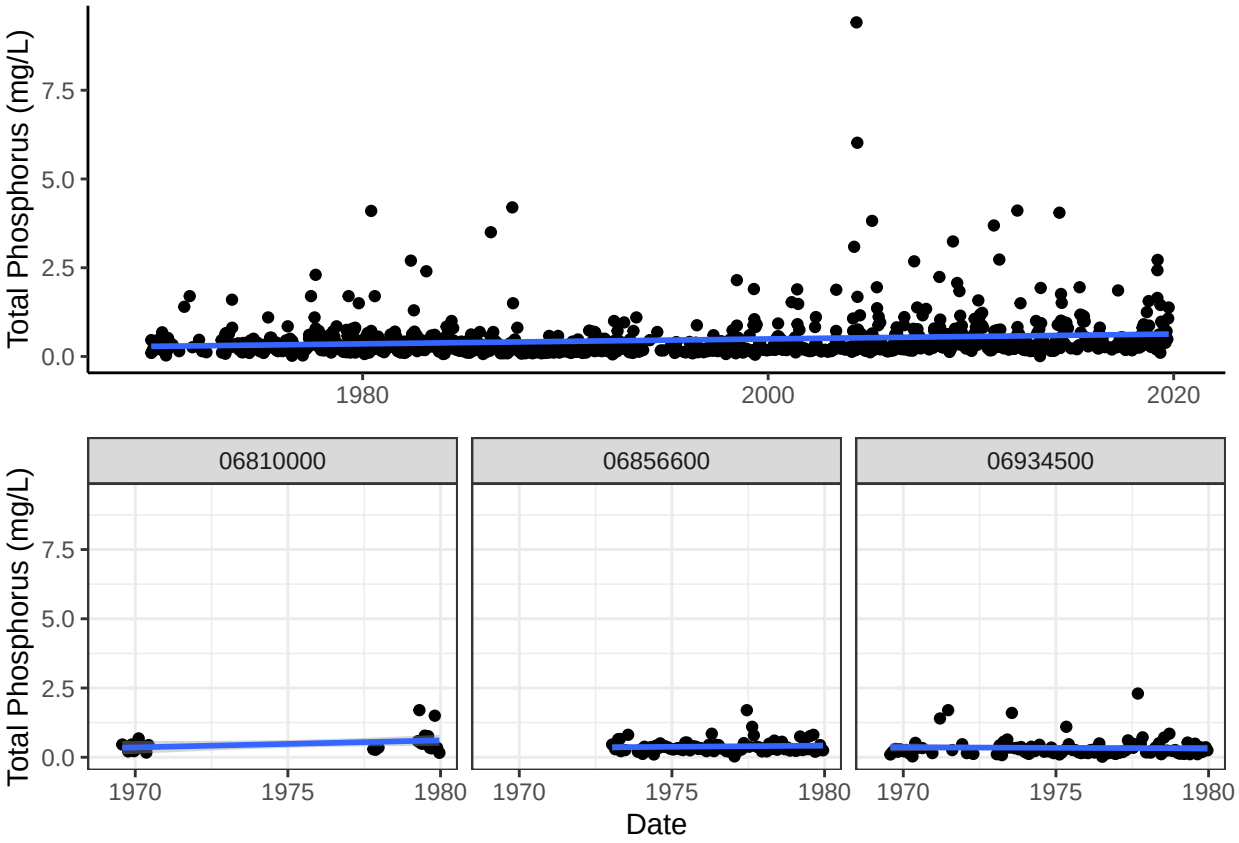


Figure 1: Total Phosphorus Over Time

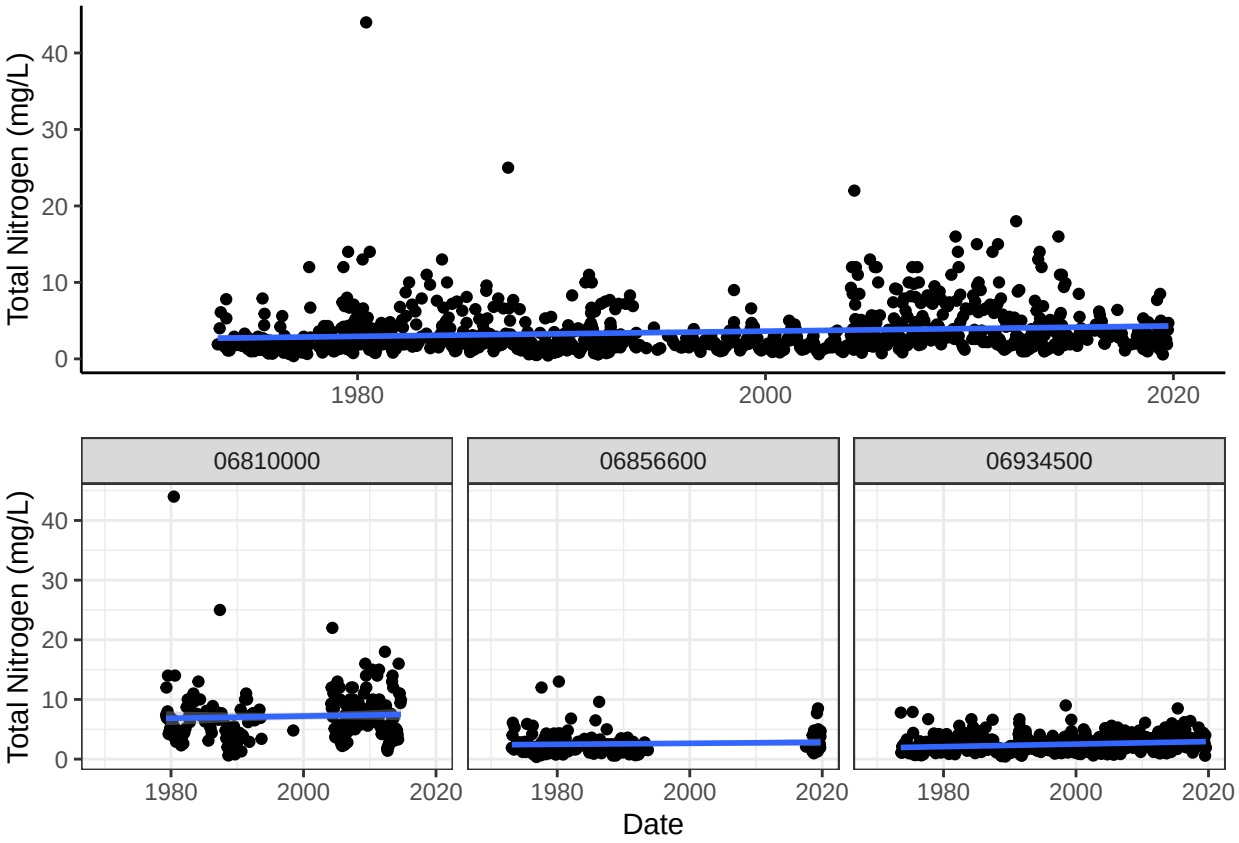


Figure 2: Total Nitrogen Over Time

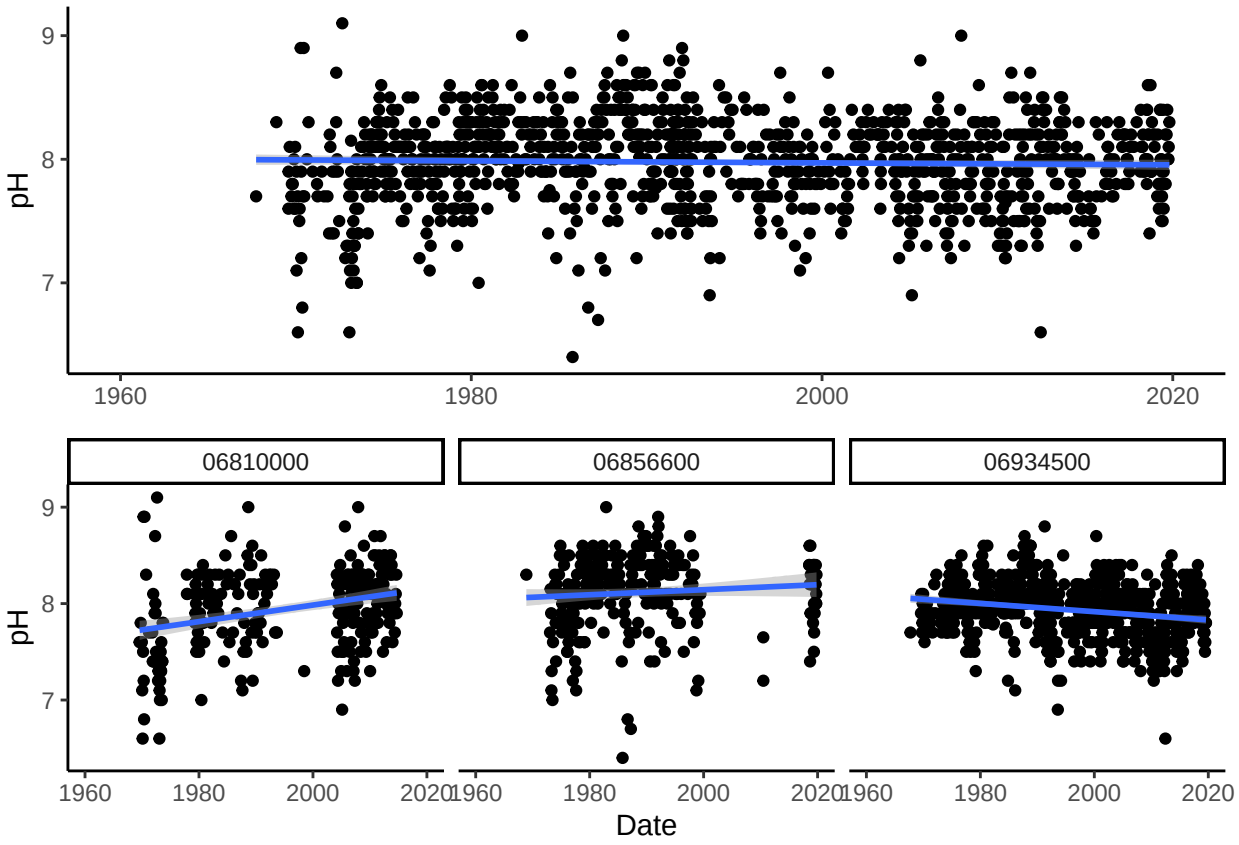


Figure 3: pH Over Time

2019. Again, site 06934500 has the most total nitrogen data, as evidenced in Figure 2.

As seen in Figure 3, pH values range from below pH of 7 to above pH of 9 from 1970 - 2019. From Figure 3, site 06810000 has a positive increase in pH over time, whereas site 06934500 has a decreasing trend in pH over time.

Figure 4 shows the total coliform measurements in the 3 chosen sites over time. From Figure 4, it is evident that there is not much data on total coliform in the Missouri River Basin and monitoring for total coliform occured at these 3 sites from late 1960s through 1975.

Figure 5 was created to determine whether high amounts of total coliform coincided with an increase in discharge. Because there were limited total coliform measurements taken, as evidenced in Figure 4, there is not a great conclusion from this data. However, Figure 5 shows a spike in discharge events between 1972 - 1974, which also happens to be a time when total coliform was sampled. Figure 5 also shows increases in total coliform between 1972 - 1975.

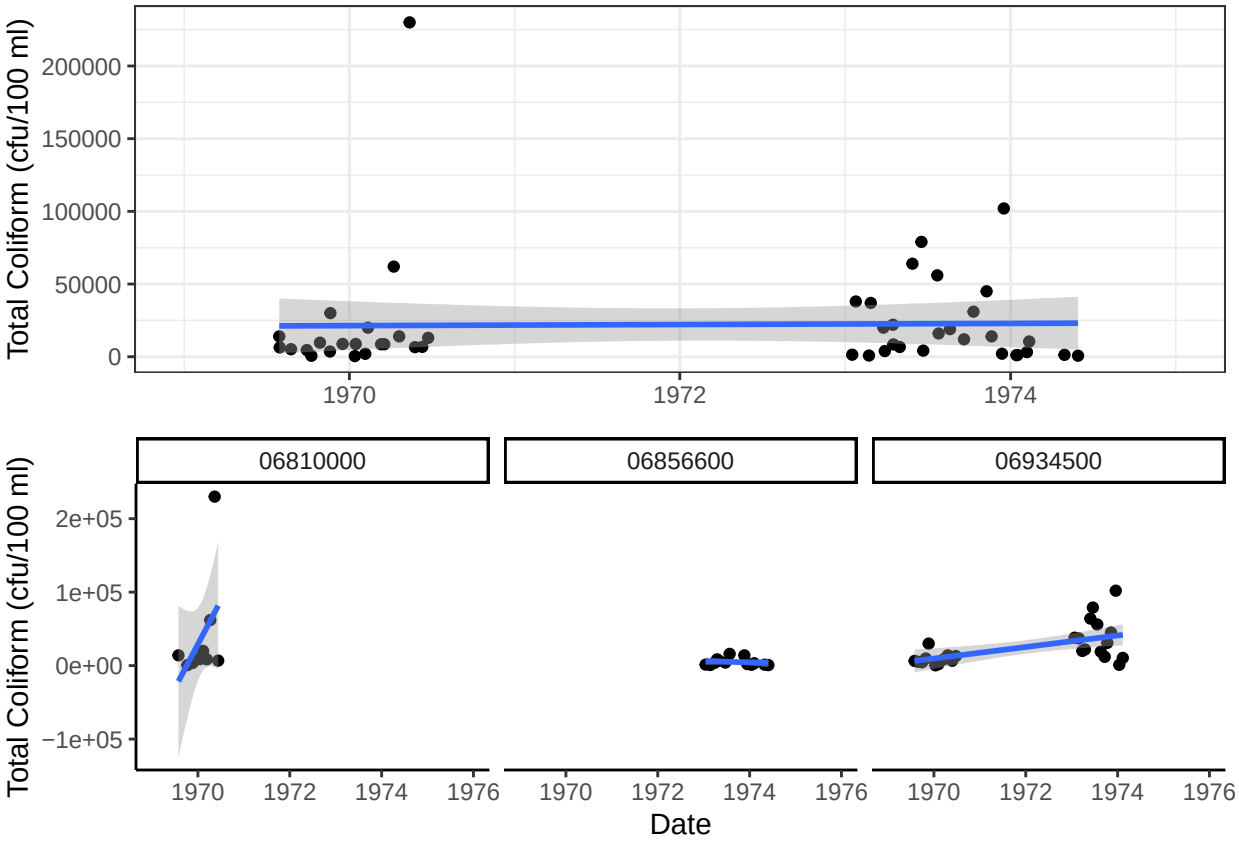


Figure 4: Total Coliform Over Time

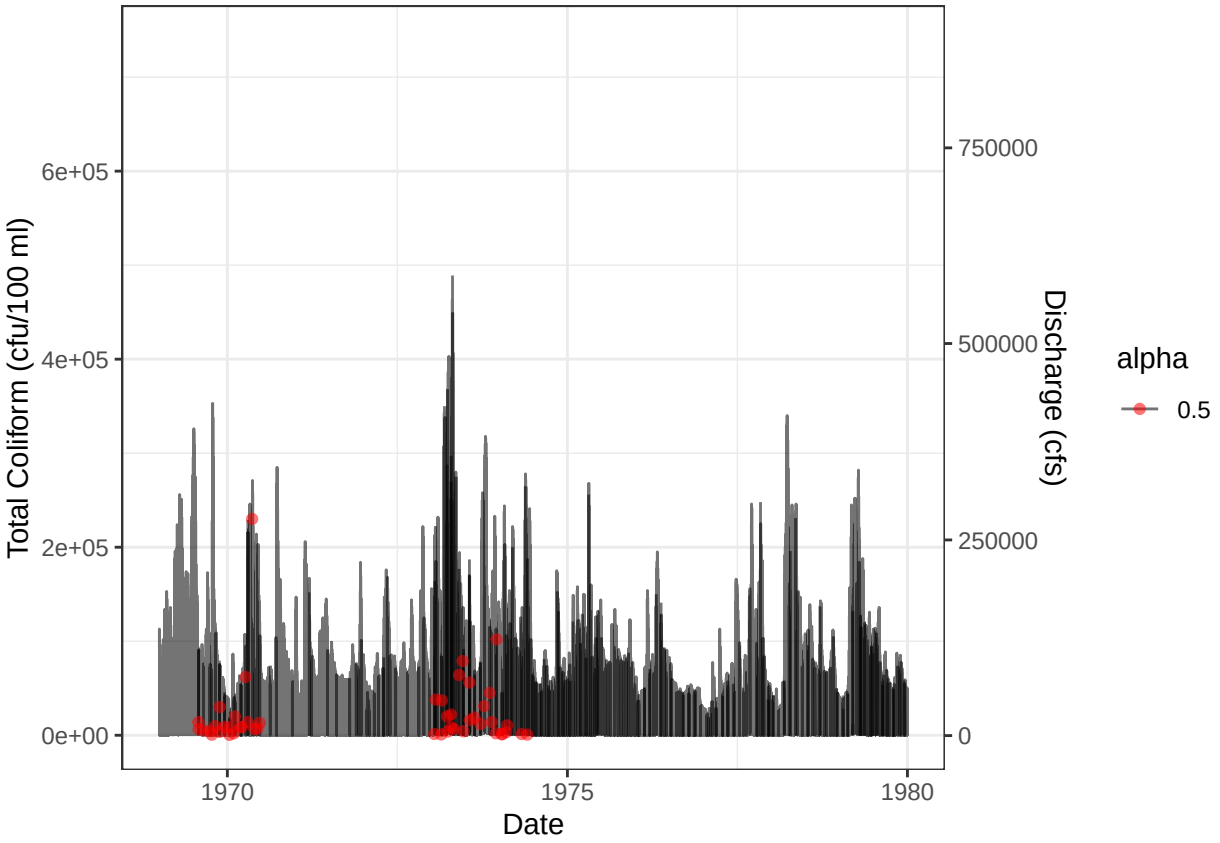


Figure 5: Total Coliform over time and Discharge over time

3.1.1 Time Series Analysis

In time series analysis, we focus on the same 3 sites chosen in the water quality analysis.

```
#Discharge Data
# 06810000 Nishnabotna River above Hamburg, IA
NRHDischarge <- readNWISdv(siteNumbers = "06810000",
                           parameterCd = "00060", # discharge (ft3/s)
                           startDate = "",
                           endDate = "")
names(NRHDischarge)[4:5] <-c("Discharge", "Approval.Code")
# 06934500 Missouri River at Hermann, MO
MRHDischarge <- readNWISdv(siteNumbers = "06934500",
                           parameterCd = "00060", # discharge (ft3/s)
                           startDate = "",
                           endDate = "")
names(MRHDischarge)[4:5] <-c("Discharge", "Approval.Code")
# 06856600 REPUBLICAN R AT CLAY CENTER, KS
RRCCDischarge <- readNWISdv(siteNumbers = "06856600",
                           parameterCd = "00060", # discharge (ft3/s)
                           startDate = "",
                           endDate = "")
names(RRCCDischarge)[4:5] <-c("Discharge", "Approval.Code")

# Nitrogen Data
# 06810000 Nishnabotna River above Hamburg, IA
NRHNitrogen <- readNWISqw(siteNumbers = "06810000",
                          parameterCd = "00600", # nitrogen (mg/L)
                          startDate = "",
                          endDate = "") %>%
  select(sample_dt, result_va)
# 06934500 Missouri River at Hermann, MO
MRHNitrogen <- readNWISqw(siteNumbers = "06934500",
                           parameterCd = "00600", # nitrogen (mg/L)
                           startDate = "",
                           endDate = "") %>%
  select(sample_dt, result_va)
# 06856600 REPUBLICAN R AT CLAY CENTER, KS
RRCCNitrogen <- readNWISqw(siteNumbers = "06856600",
                          parameterCd = "00600", # nitrogen (mg/L)
                          startDate = "",
                          endDate = "") %>%
  select(sample_dt, result_va)

#Phosphorus Data
```

```
# 06810000 Nishnabotna River above Hamburg, IA
NRHPhosphorus <- readNWISqw(siteNumbers = "06810000",
                            parameterCd = "00665", # phosphorus (mg/L)
                            startDate = "",
                            endDate = "") %>%
  select(sample_dt, result_va)
# 06934500 Missouri River at Hermann, MO
MRHPhosphorus <- readNWISqw(siteNumbers = "06934500",
                              parameterCd = "00665", # phosphorus (mg/L)
                              startDate = "",
                              endDate = "") %>%
  select(sample_dt, result_va)
# 06856600 REPUBLICAN R AT CLAY CENTER, KS
RRCCPhosphorus <- readNWISqw(siteNumbers = "06856600",
                            parameterCd = "00665", # phosphorus (mg/L)
                            startDate = "",
                            endDate = "") %>%
  select(sample_dt, result_va)
```

As seen by Figure 6, there is a gap of discharge data in 06810000. Therefore, we filter the discharge data by the date and only use the data available after 1928-01-01 for site 06810000.

Also, we do the same things for nitrogen and phosphorus.

3.2 High Frequency Nitrogen and Discharge

```
#high frequency data wrangling
highfreqsite2019 <- highfreqsiteinfo %>%
  filter(end_date > "2019-03-31"); head(highfreqsite2019)
```

```
## Warning in Ops.factor(end_date, "2019-03-31"): '>' not meaningful for
## factors
```

```
##  [1] X                 agency_cd         site_no
##  [4] station_nm        site_tp_cd        dec_lat_va
##  [7] dec_long_va       coord_acy_cd      dec_coord_datum_cd
## [10] alt_va            alt_acy_va        alt_datum_cd
## [13] huc_cd            data_type_cd      parm_cd
## [16] stat_cd           ts_id             loc_web_ds
## [19] medium_grp_cd     parm_grp_cd       srs_id
## [22] access_cd         begin_date        end_date
## [25] count_nu
## <0 rows> (or 0-length row.names)
```

```
highfreqsites.DN <- readNWISuv(site = c("06808500", "06817000", "06892350", "06934500"),
                               parameterCd = c("00060", "99133"),
```

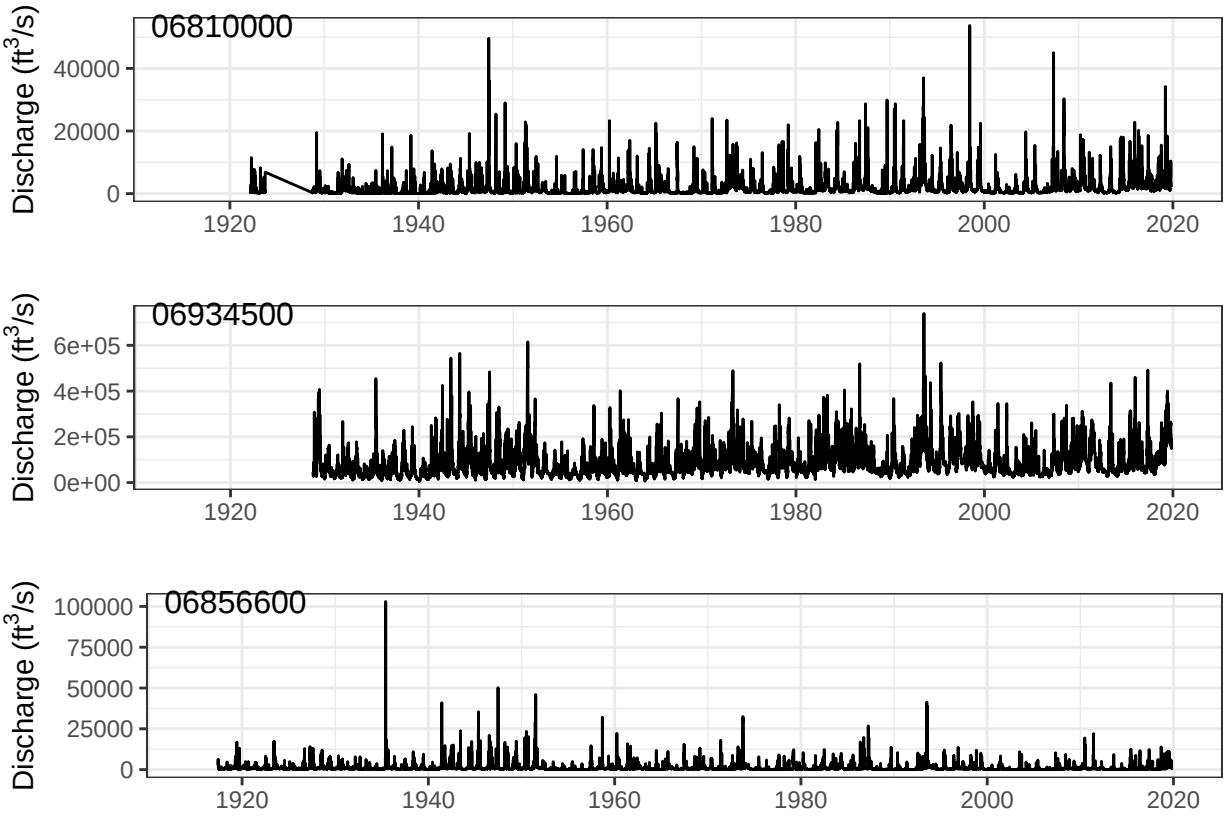


Figure 6: Discharge Over Time

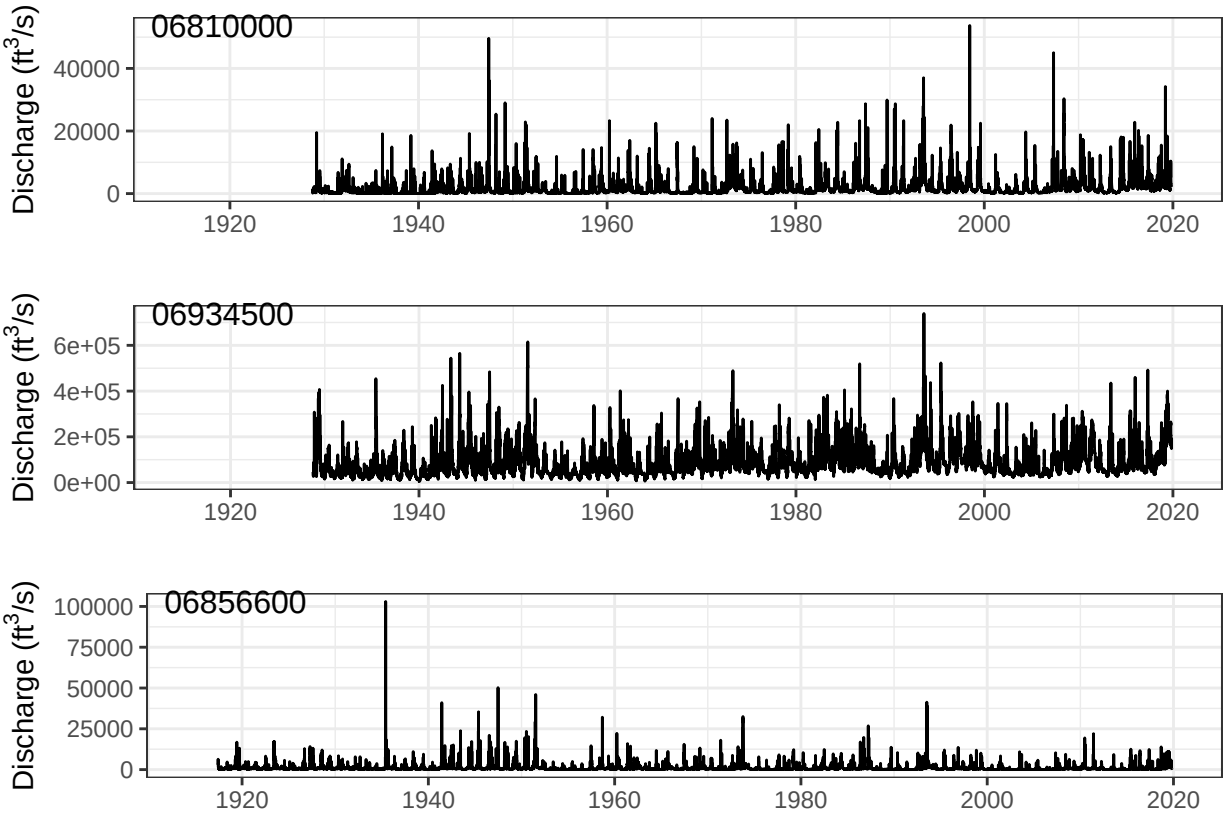


Figure 7: Discharge Over Time for 3 sites

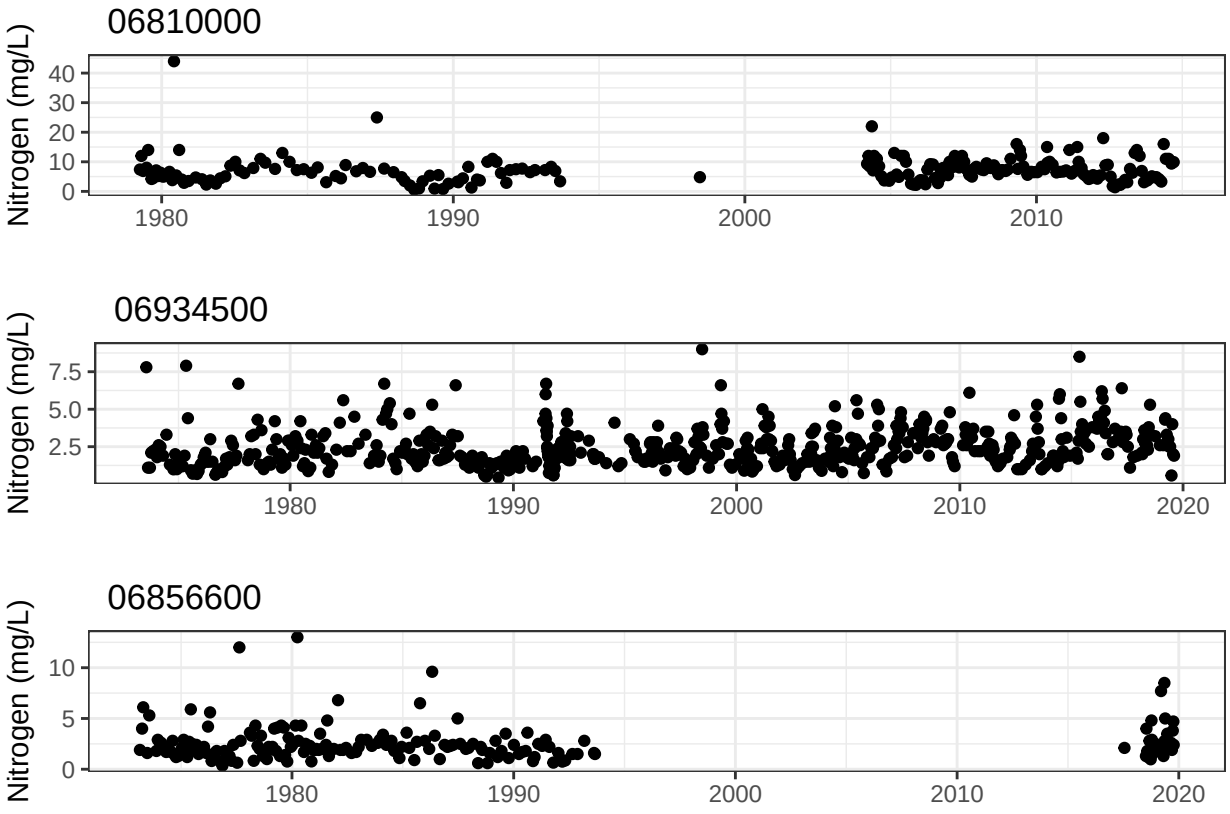


Figure 8: Nitrogen Over Time

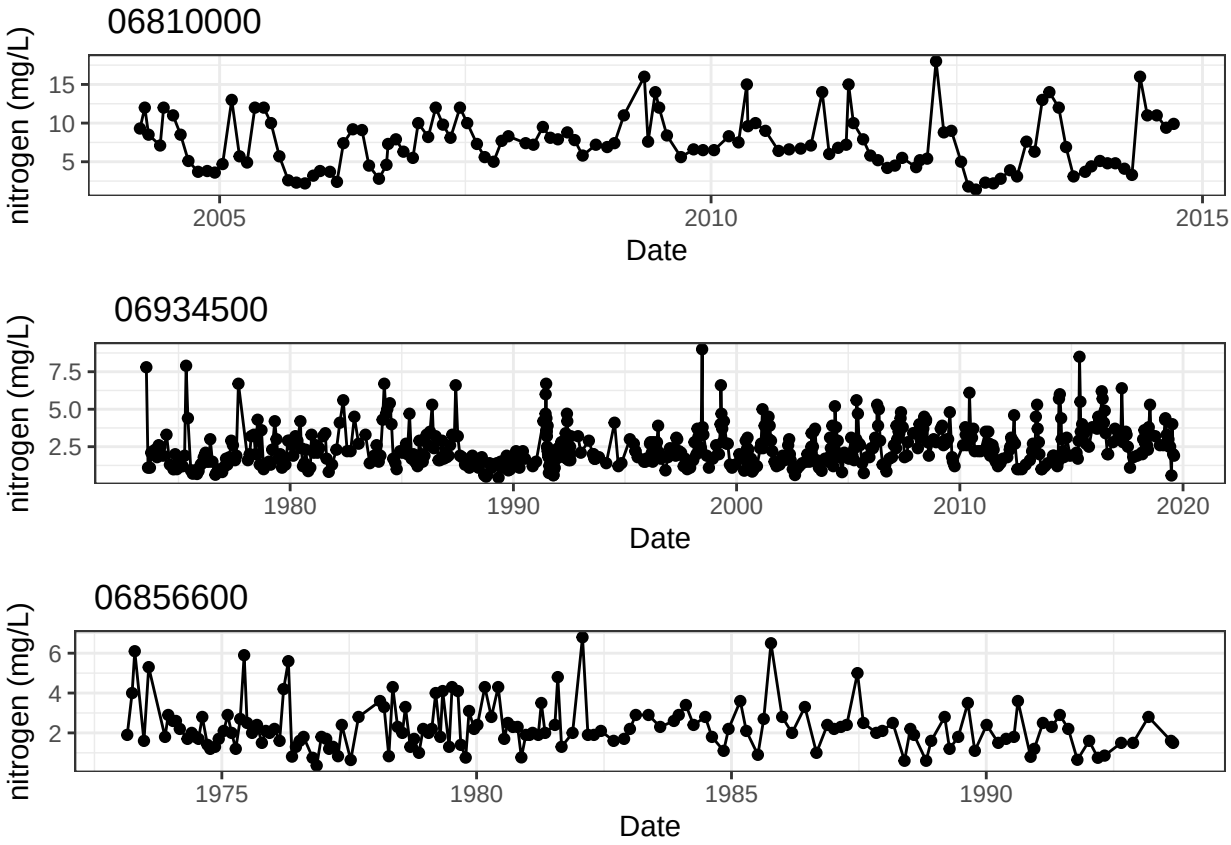


Figure 9: Nitrogen Over Time for 3 sites after filtering

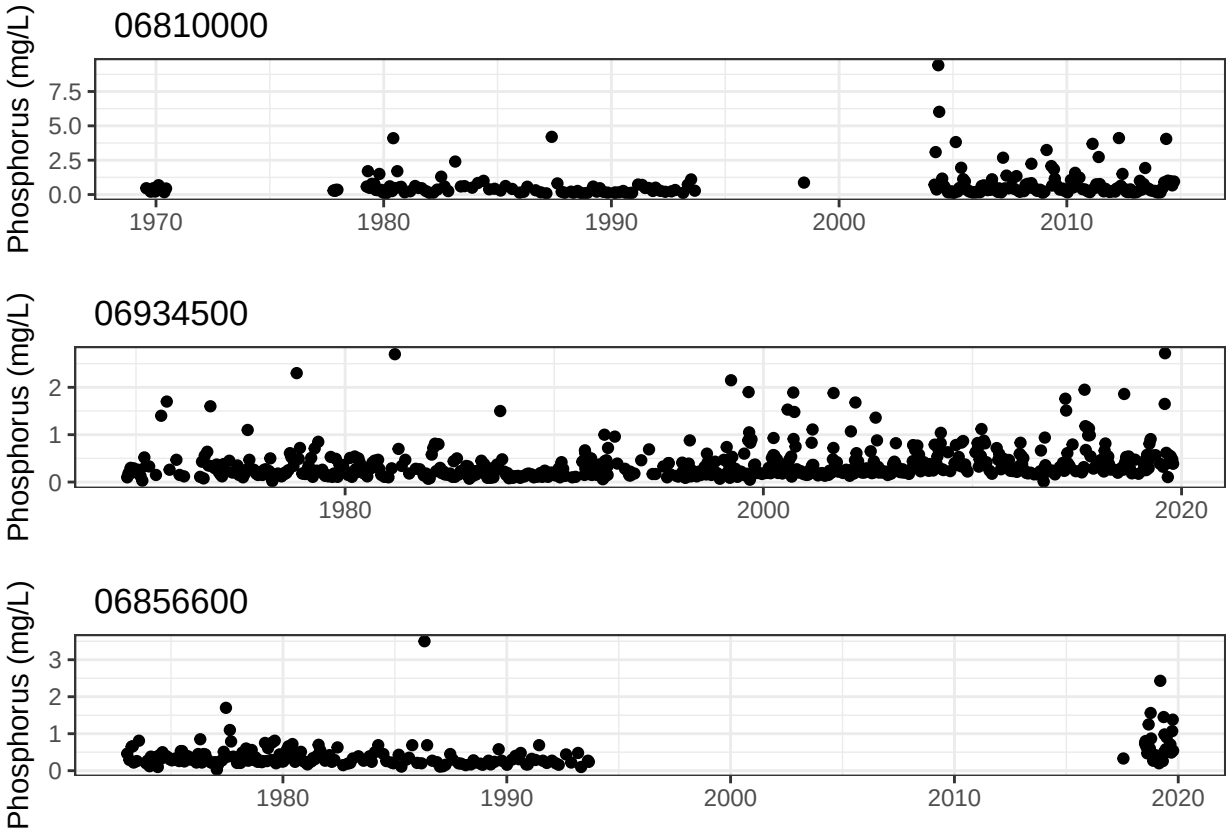


Figure 10: Phosphorus Over Time

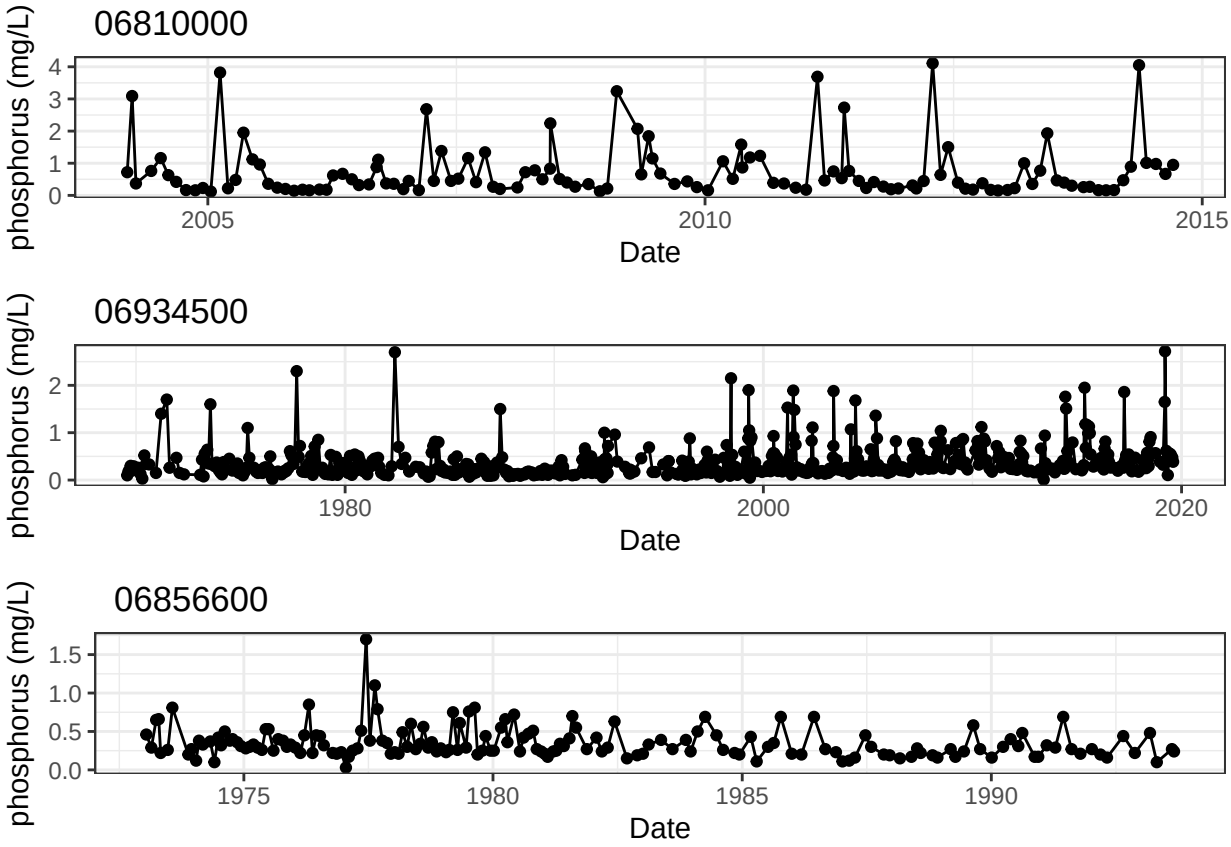


Figure 11: Phosphorus Over Time for 3 sites after filtering

```
                                # Discharge in cfs & Nitrate in mg/l NO3-N
                                startDate = "2019-01-01",
                                endDate = "2019-11-01") %>%
                                renameNWISColumns() %>%
                                rename(Nitrate_mgl = 6)

#individual sites
Hermann <- highfreqsites.DN %>%
           filter(site_no=="06934500")
Desoto <- highfreqsites.DN %>%
          filter(site_no=="06892350")
Clarinda <- highfreqsites.DN %>%
            filter(site_no=="06817000")
Randolph <- highfreqsites.DN %>%
            filter(site_no=="06808500")
```

There were 7 sites in our region of interest that had high freq N data, and only 4 sites had high freq N data during the floods of 2019. The sites looked at in depth are:

```
- West Nishnabotna River in Randolph, IA
- Nodaway River at Clarinda, IA
- Kansas River in Desoto, KS
- Missouri River at Hermann, MO
```

The Missouri River is the biggest river, with an average of 214693 cfs discharge rate during the year 2019, and the Nodaway River is the smallest river, with an average of 1185 cfs discharge rate for 2019.

In March of 2019, a bomb cyclone hit the midwest. Our initial research question, what effect did the March 2019 storm have on water quality, attempted to look into the behavior of nitrogen in the discharge of the rivers. Unfortunately, instantaneous Nitrogen values stopped recording during the peak of the storm events in March, so it was hard to create hysteresis plots that exhibited the type of storm and its effects on nitrogen concentration.

Even though Nitrogen concentrations were not recorded in March, they were recorded in other times of the year. 2019 was a wet year and many large storm events occurred.

```
## Warning: Removed 9 rows containing missing values (geom_point).
```

The Figure 12 shows Hysteresis plots for two storm events in the Missouri River Basin. The storm event on the West Nishnabotna River exhibits an oddly-shaped plot that has a negative slope, indicating it is a diluting storm. The Kansas River experienced a storm in late February that has a counter-clockwise motion and a positive slope, indicating a flushing storm. These two plots illustrate that two rivers near each other can have very different behaviors.

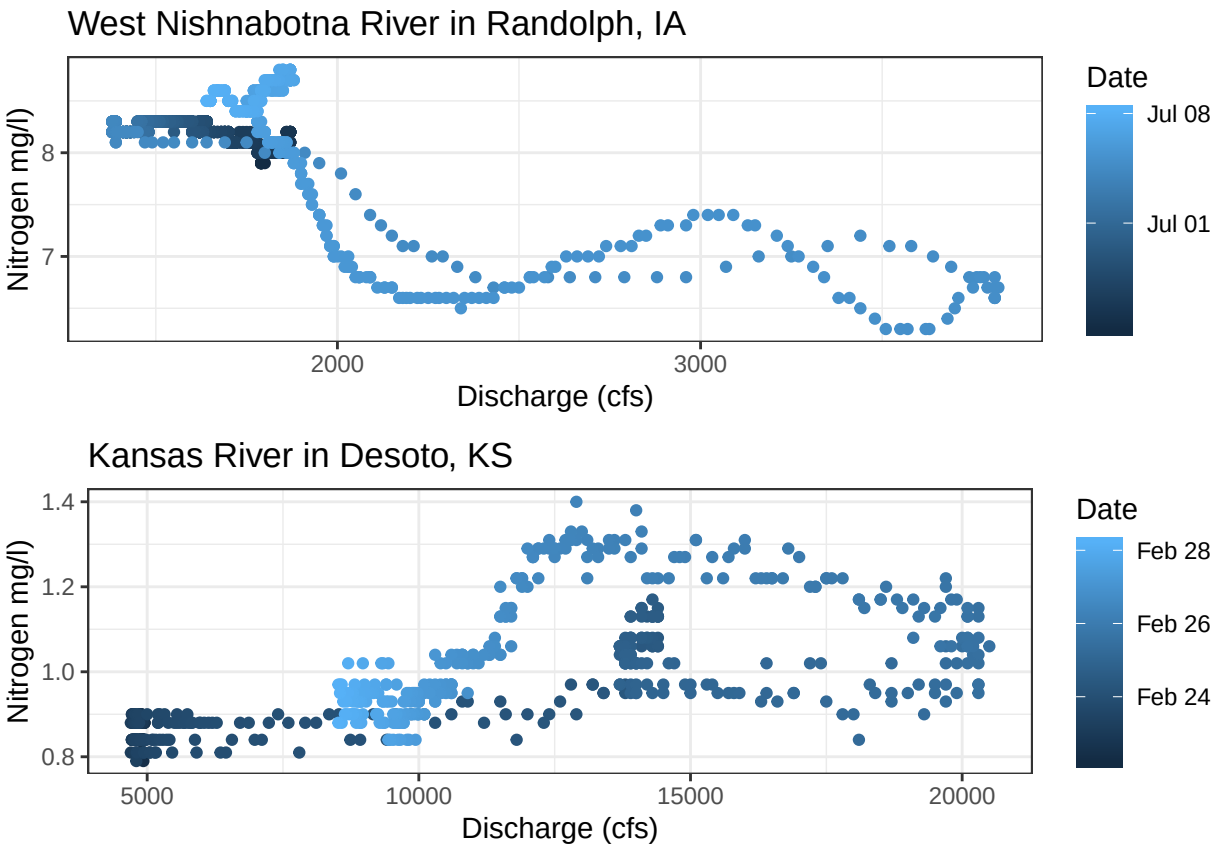


Figure 12: Hysteresis plots

4 Analysis

4.1 Linear Models for Water Quality

Linear models were run for total nitrogen, total phosphorus, total coliform, and pH to determine if there were significant patterns in the data.

```
##
## Call:
## lm(formula = total.nitrogen ~ Date, data = bestsites.wq.skinny)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.708 -1.709 -0.917  0.524 41.040
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.605e+00  2.082e-01  12.517  < 2e-16 ***
## Date        9.310e-05  1.928e-05   4.828  1.6e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.051 on 992 degrees of freedom
##   (104765 observations deleted due to missingness)
## Multiple R-squared:  0.02296,    Adjusted R-squared:  0.02197
## F-statistic: 23.31 on 1 and 992 DF,  p-value: 1.598e-06
```

```
##
## Call:
## lm(formula = total.coliform ~ Date, data = bestsites.wq.skinny)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -22343 -18568 -12589  -1235 208472
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 21389.217   8351.431   2.561   0.0138 *
## Date            1.046      8.329   0.126   0.9006
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 38270 on 46 degrees of freedom
##   (105711 observations deleted due to missingness)
## Multiple R-squared:  0.0003425,  Adjusted R-squared:  -0.02139
## F-statistic: 0.01576 on 1 and 46 DF,  p-value: 0.9006
```

```
##
## Call:
## lm(formula = total.phosphorus ~ Date, data = bestsites.wq.skinny)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.5768 -0.2685 -0.1320  0.0578  8.8856
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.876e-01  3.570e-02   8.055 2.14e-15 ***
## Date        1.887e-05  3.394e-06   5.561 3.40e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5752 on 1048 degrees of freedom
##   (104709 observations deleted due to missingness)
## Multiple R-squared:  0.02866,    Adjusted R-squared:  0.02774
## F-statistic: 30.92 on 1 and 1048 DF,  p-value: 3.404e-08
```

```
##
## Call:
## lm(formula = pH ~ Date, data = bestsites.wq.skinny)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.57958 -0.18708  0.02653  0.23034  1.11246
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  7.989e+00  1.973e-02 404.824   <2e-16 ***
## Date        -1.660e-06  1.938e-06  -0.856    0.392
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.355 on 1199 degrees of freedom
##   (104558 observations deleted due to missingness)
## Multiple R-squared:  0.0006111,  Adjusted R-squared:  -0.0002224
## F-statistic: 0.7331 on 1 and 1199 DF,  p-value: 0.392
```

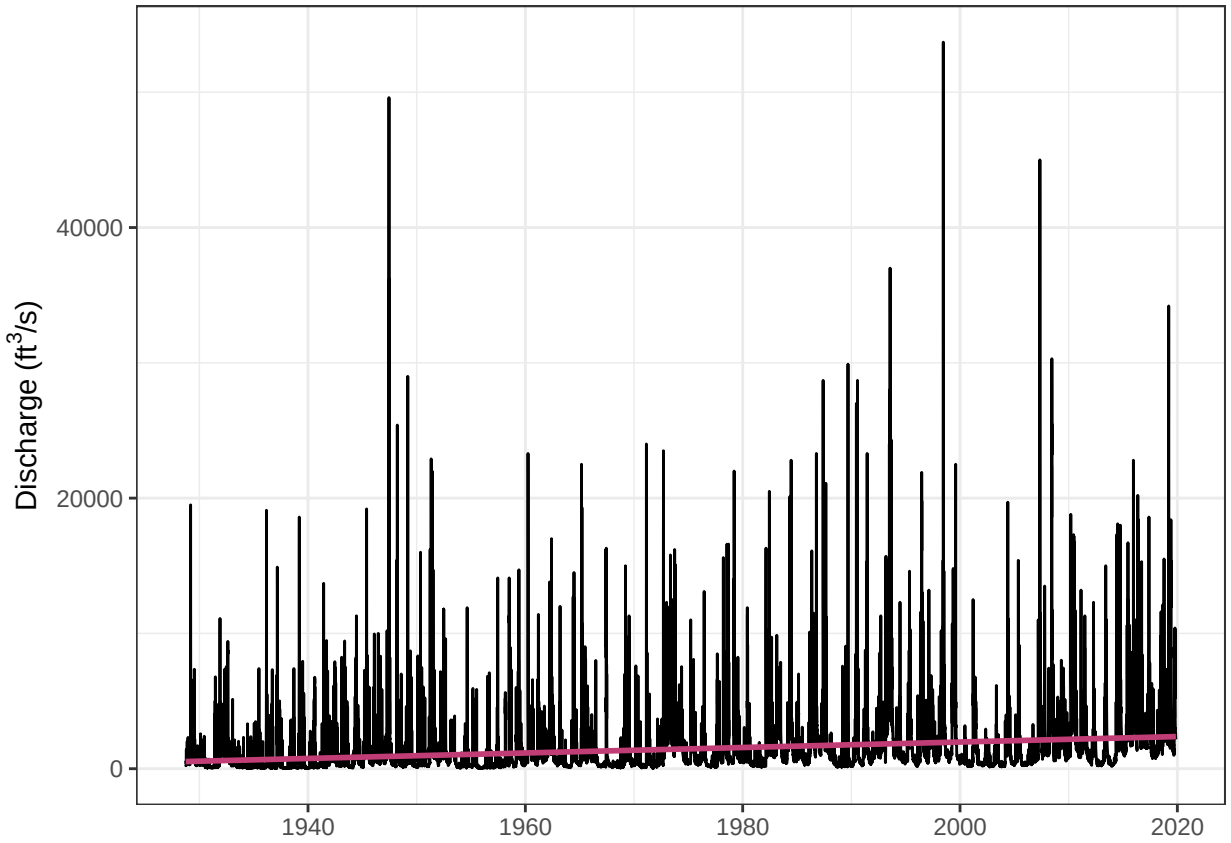
Linear models were created in order to determine whether water quality parameters have changed over time. When nitrogen was evaluated for all sites over the time period, it was found that nitrogen has significantly increased over time (p< 2e-16, F_{1,992} = 23.31). At the six sites that total coliform was recorded, total coliform significantly increased over the time period (p < 0.0138, F_{1,46} = 0.01576). However, this result should be critically analyzed, because total coliform has not been measured since the 1980s. Total coliform levels

should be measured more closely in this region, given the lack of data. A linear model was also performed on total phosphorus over time, and it was found that phosphorus levels have also increased over the time period for all sites of interest ($p < 2.14e\text{-}15$, $F\_\{1,1048\} = 30.92$). Lastly, pH was analyzed in a linear model and it was found that pH had no significant increase or decrease over time in all of the sites of interest ($p = 0.392$).

A time series analysis will be conducted on the discharge from the chosen sies. This will include an analysis of baseflow and quickflow. A dygraph of discharge over time will be created to be able to better examine large increases in discharge.

For high frequency data, a hystersis plot will be created to determine hydrologic flashiness.

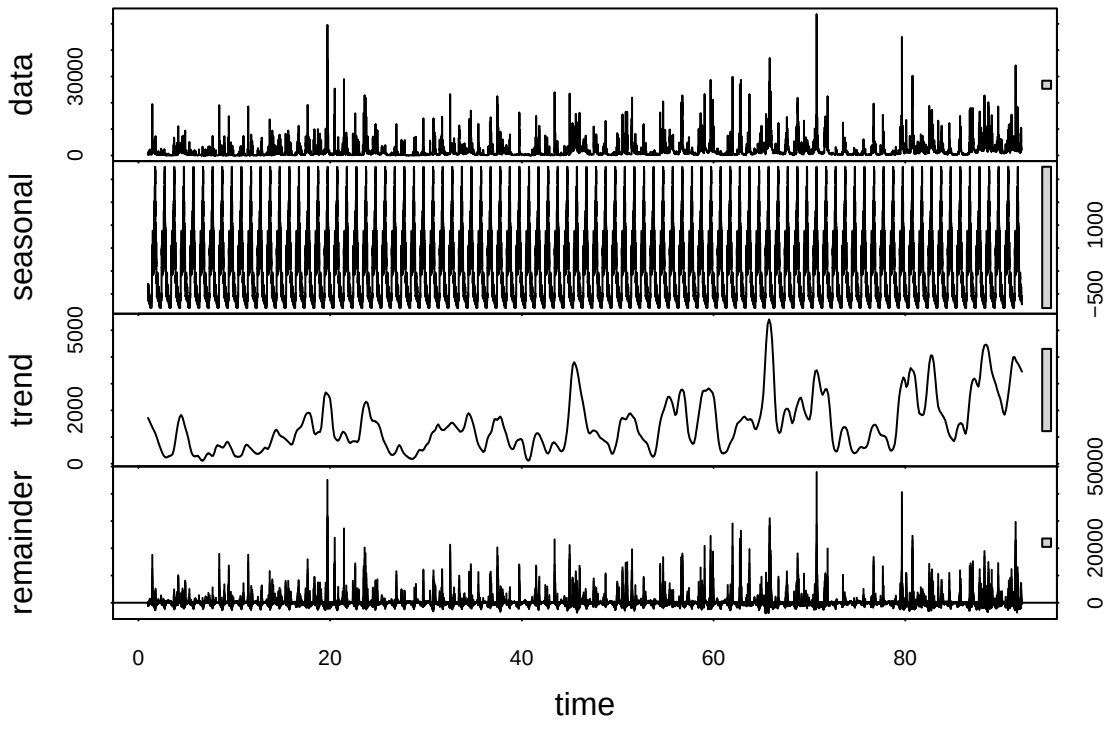
```
##Time Series

# 06810000 Nishnabotna River above Hamburg, IA

############################ Discharge Analysis ##############################
NRHRegressionPlot <-
  ggplot(NRHDischarge, aes(x = Date, y = Discharge)) +
  geom_line() +
  geom_smooth(method = "lm", se = FALSE, color = "#c13d75ff") +
  labs(x = "", y = expression("Discharge (ft"^3*"/s)")) +
  theme(plot.title = element_text(margin = margin(b = -10), size = 12),
        axis.title.x = element_blank())
print(NRHRegressionPlot)
```



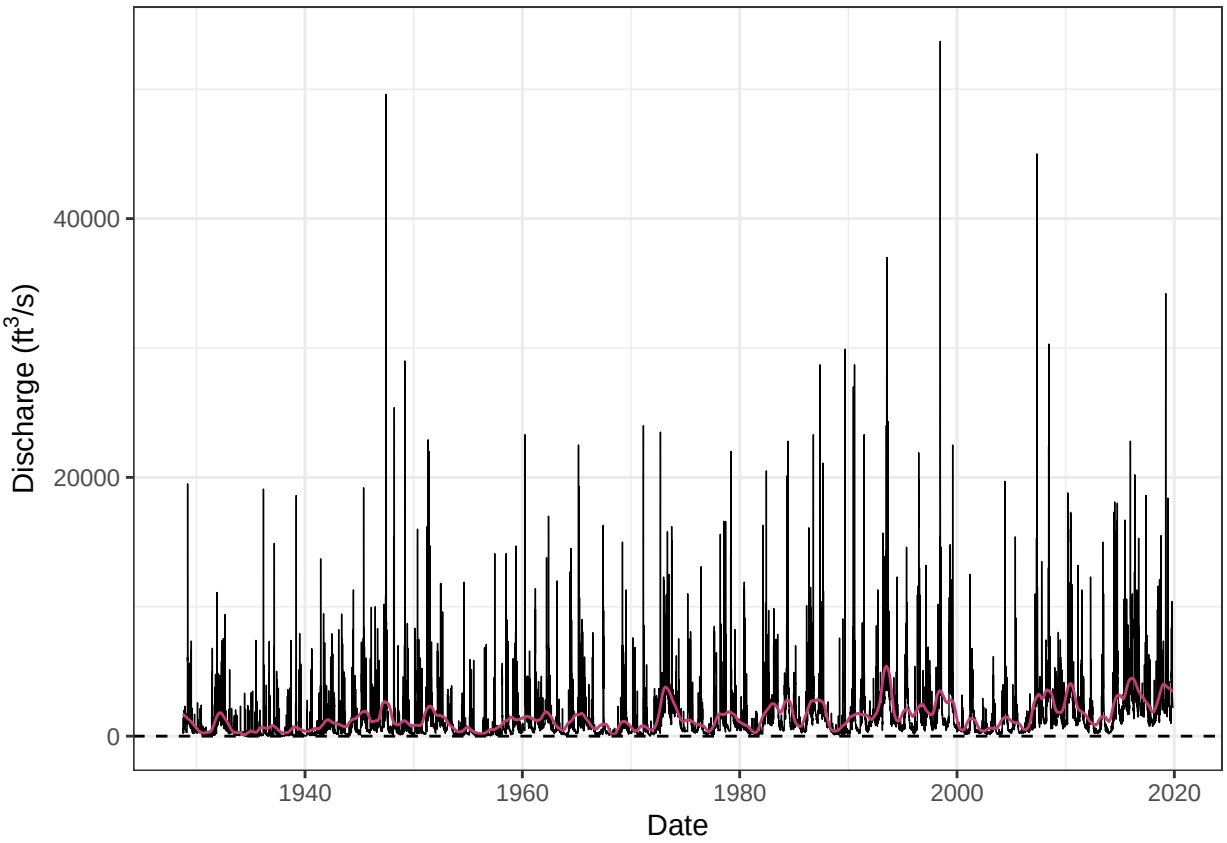

```
NRHDischarge <- na.omit(NRHDischarge)
NRH_ts <- ts(NRHDischarge[[4]], frequency = 365)
table(diff(NRHDischarge$Date))
```

```
##
##     1
## 33279
```

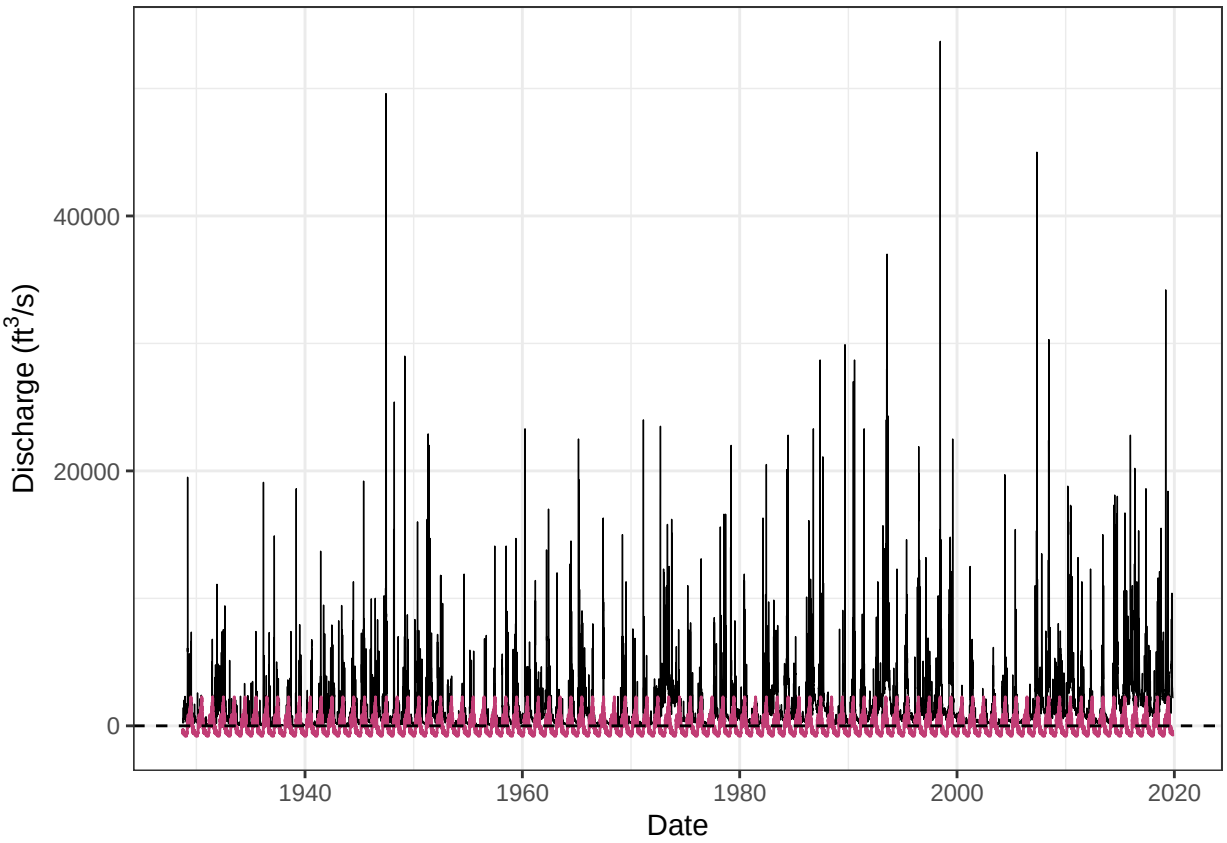
```r
# Generate the decomposition
NRH_Decomposed <- stl(NRH_ts, s.window = "periodic")

# Visualize the decomposed series.
plot(NRH_Decomposed)
```




```r
# We can extract the components and turn them into data frames
NRH_Components <- as.data.frame(NRH_Decomposed$time.series[,1:3])
NRH_Components <- mutate(NRH_Components,
                         Observed = NRHDischarge$Discharge,
                         Date = NRHDischarge$Date)

# Visualize how the trend maps onto the data
ggplot(NRH_Components) +
  geom_line(aes(y = Observed, x = Date),  size = 0.25) +
  geom_line(aes(y = trend, x = Date), color = "#c13d75ff") +
  geom_hline(yintercept = 0, lty = 2) +
  ylab(expression("Discharge (ft"^3*"/s)"))
```




```
# Visualize how the seasonal cycle maps onto the data
ggplot(NRH_Components) +
  geom_line(aes(y = Observed, x = Date),  size = 0.25) +
  geom_line(aes(y = seasonal, x = Date), color = "#c13d75ff") +
  geom_hline(yintercept = 0, lty = 2) +
  ylab(expression("Discharge (ft"^3*"/s)"))
```

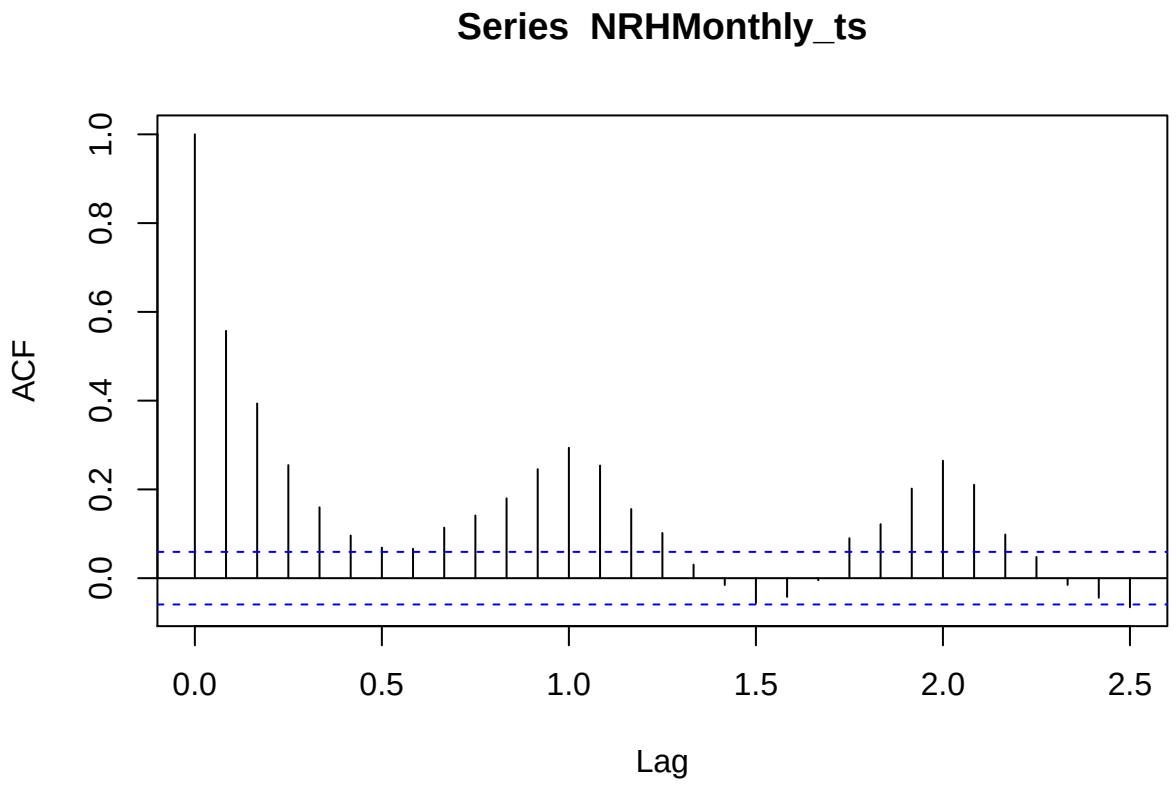



```
NRHDischarge.Monthly <- NRHDischarge %>%
  mutate(Year = year(Date),
         Month = month(Date)) %>%
  group_by(Year, Month) %>%
  summarise(Discharge = mean(Discharge))
NRHMonthly_ts <- ts(NRHDischarge.Monthly[[3]], frequency = 12)
adf.test(NRHMonthly_ts, alternative = "stationary")
```

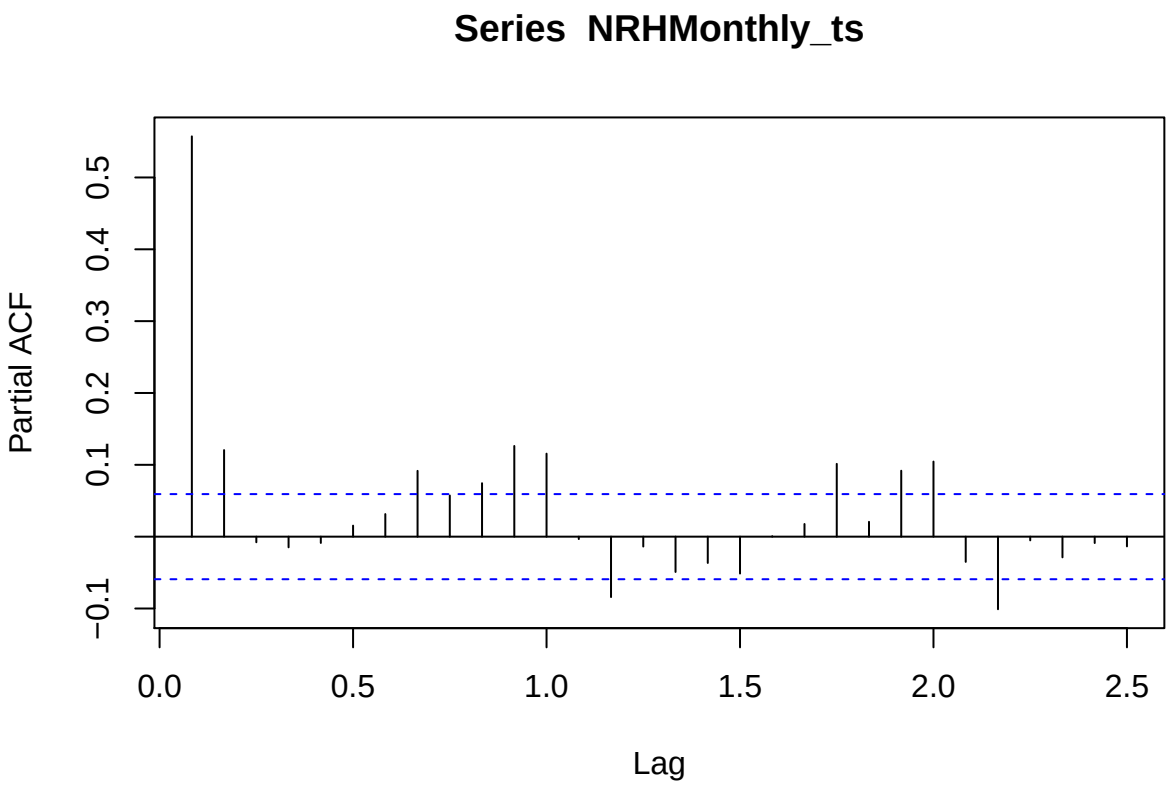
```
## Warning in adf.test(NRHMonthly_ts, alternative = "stationary"): p-value
## smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  NRHMonthly_ts
## Dickey-Fuller = -7.0535, Lag order = 10, p-value = 0.01
## alternative hypothesis: stationary
```

```
acf(NRHMonthly_ts)
```




```
pacf(NRHMonthly_ts)
```



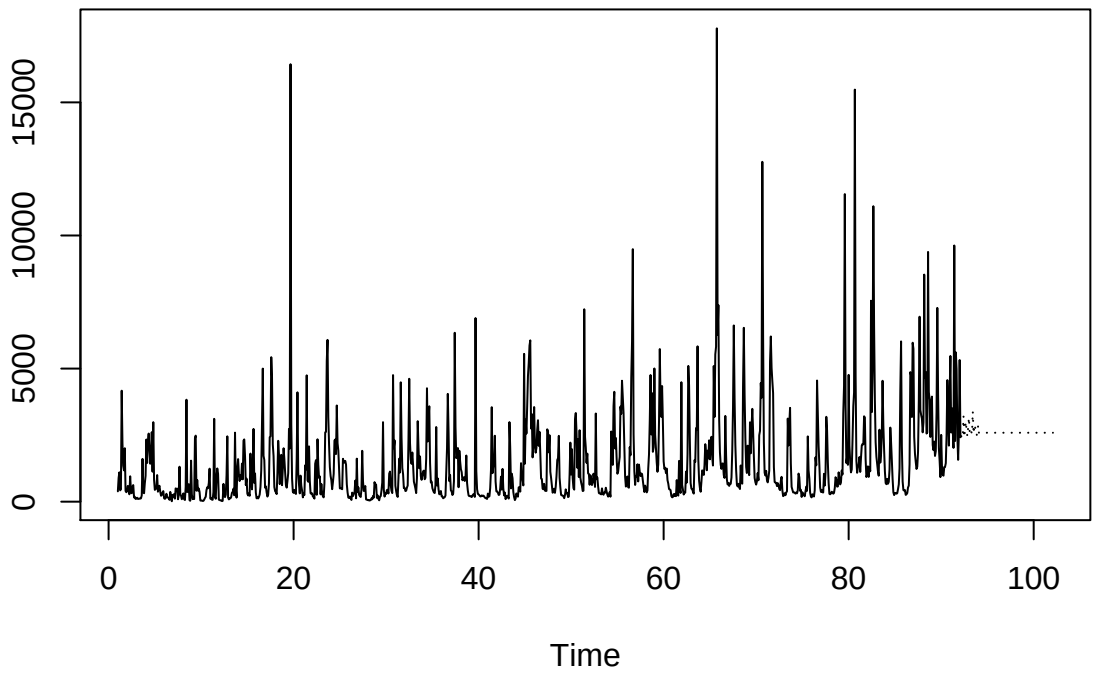

```
# run the arima function and search for best fit
auto.arima(NRHMonthly_ts, trace = TRUE)
```

```
##
##  Fitting models using approximations to speed things up...
##
##  ARIMA(2,1,2)(1,0,1)[12] with drift         : Inf
##  ARIMA(0,1,0)            with drift         : 19302.7
##  ARIMA(1,1,0)(1,0,0)[12] with drift         : 19168.62
##  ARIMA(0,1,1)(0,0,1)[12] with drift         : 19120.96
##  ARIMA(0,1,0)                               : 19300.7
##  ARIMA(0,1,1)            with drift         : 19146.2
##  ARIMA(0,1,1)(1,0,1)[12] with drift         : Inf
##  ARIMA(0,1,1)(0,0,2)[12] with drift         : 19094.06
##  ARIMA(0,1,1)(1,0,2)[12] with drift         : Inf
##  ARIMA(0,1,0)(0,0,2)[12] with drift         : 19284.84
##  ARIMA(1,1,1)(0,0,2)[12] with drift         : Inf
##  ARIMA(0,1,2)(0,0,2)[12] with drift         : 19052.68
##  ARIMA(0,1,2)(0,0,1)[12] with drift         : 19068.11
##  ARIMA(0,1,2)(1,0,2)[12] with drift         : Inf
##  ARIMA(0,1,2)(1,0,1)[12] with drift         : Inf
##  ARIMA(1,1,2)(0,0,2)[12] with drift         : Inf
```

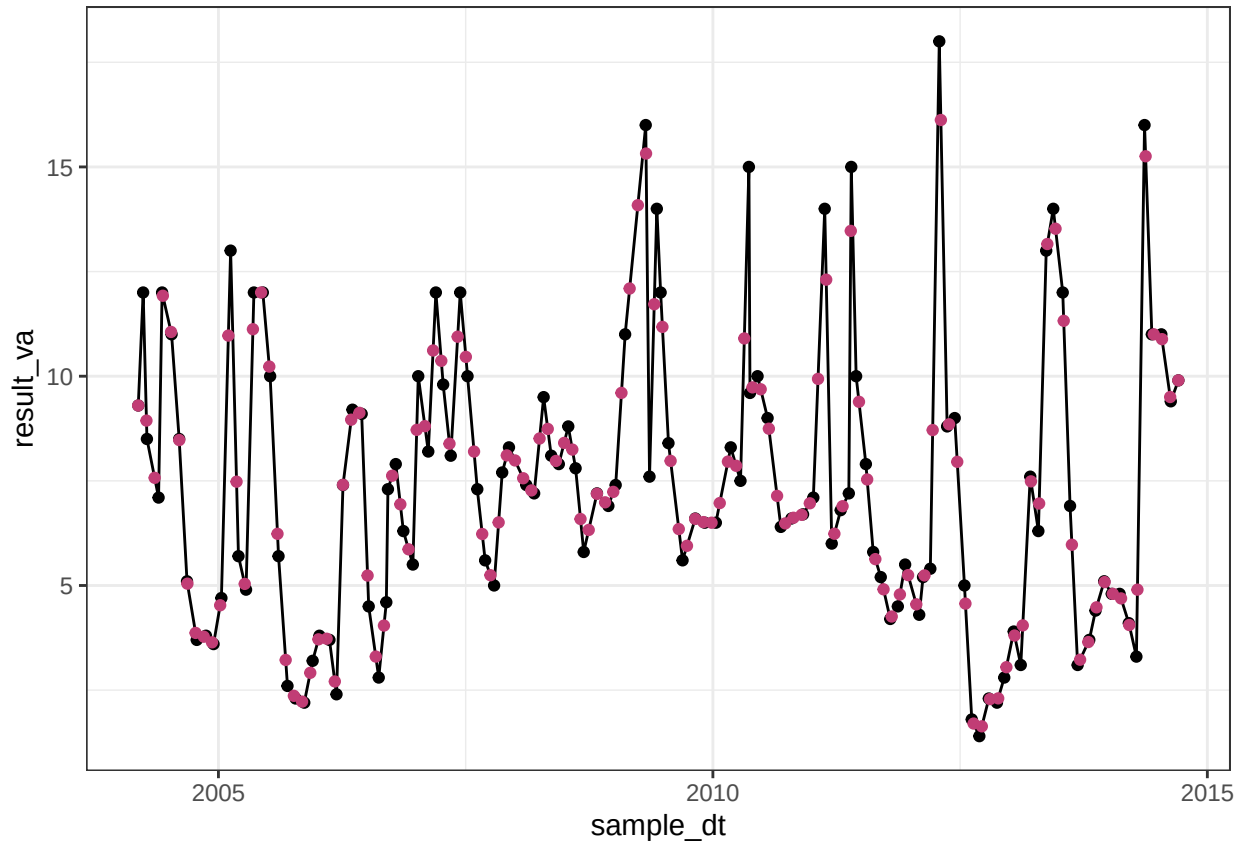
```
##  ARIMA(0,1,3)(0,0,2)[12] with drift         : Inf
##  ARIMA(1,1,3)(0,0,2)[12] with drift         : Inf
##  ARIMA(0,1,2)(0,0,2)[12]                    : 19050.72
##  ARIMA(0,1,2)(0,0,1)[12]                    : 19066.22
##  ARIMA(0,1,2)(1,0,2)[12]                    : Inf
##  ARIMA(0,1,2)(1,0,1)[12]                    : Inf
##  ARIMA(0,1,1)(0,0,2)[12]                    : 19092.05
##  ARIMA(1,1,2)(0,0,2)[12]                    : 18972.02
##  ARIMA(1,1,2)(0,0,1)[12]                    : 18988.33
##  ARIMA(1,1,2)(1,0,2)[12]                    : Inf
##  ARIMA(1,1,2)(1,0,1)[12]                    : Inf
##  ARIMA(1,1,1)(0,0,2)[12]                    : 18982.4
##  ARIMA(2,1,2)(0,0,2)[12]                    : Inf
##  ARIMA(1,1,3)(0,0,2)[12]                    : Inf
##  ARIMA(0,1,3)(0,0,2)[12]                    : 18994.1
##  ARIMA(2,1,1)(0,0,2)[12]                    : Inf
##  ARIMA(2,1,3)(0,0,2)[12]                    : Inf
##
##  Now re-fitting the best model(s) without approximations...
##
##  ARIMA(1,1,2)(0,0,2)[12]                    : Inf
##  ARIMA(1,1,1)(0,0,2)[12]                    : 18997.57
##
##  Best model: ARIMA(1,1,1)(0,0,2)[12]
```

```
## Series: NRHMonthly_ts
## ARIMA(1,1,1)(0,0,2)[12]
##
## Coefficients:
##          ar1      ma1    sma1    sma2
##       0.4935  -0.9892  0.1054  0.1108
## s.e.  0.0274   0.0053  0.0318  0.0289
##
## sigma^2 estimated as 2053595:  log likelihood=-9493.76
## AIC=18997.51   AICc=18997.57   BIC=19022.5
```

```
# create an object that defines the best fit model
NRHfit <- arima(NRHMonthly_ts, c(1, 1, 1),seasonal = list(order = c(0, 0, 2), period = 1

# make a prediction into the future
NRHprediction <- predict(NRHfit, n.ahead = 10*12)

# plot future predictions
ts.plot(NRHMonthly_ts, NRHprediction$pred, lty = c(1, 3))
```




```
############################ Nitrogen Analysis ############################
# Generate monthly values from March 2004 to September 2014
NRHNitrogenlp <- as.data.frame(approx(NRHNitrogen, n = 128, method = "linear"))
NRHNitrogenlp$x <- as.Date(NRHNitrogenlp$x, origin = "1970-01-01")
names(NRHNitrogenlp) <- c("Date", "N")

# Inspect interpolated values
NRHNinterpolated <-
  ggplot(NRHNitrogen, aes(x = sample_dt, y = result_va)) +
  geom_point() +
  geom_line() +
  geom_point(data = NRHNitrogenlp, aes(x = Date, y = N), color = "#c13d75ff")
print(NRHNinterpolated)
```




```
# Generate time series (smk.test needs ts, not data.frame)
NRHNtimeseries <- ts(NRHNitrogenlp$N, frequency = 12,
                     start = c(2004, 3, 11), end = c(2014, 9, 16))
# Run SMK test
NRHNtrend <- smk.test(NRHNtimeseries)

# Inspect results
NRHNtrend
```

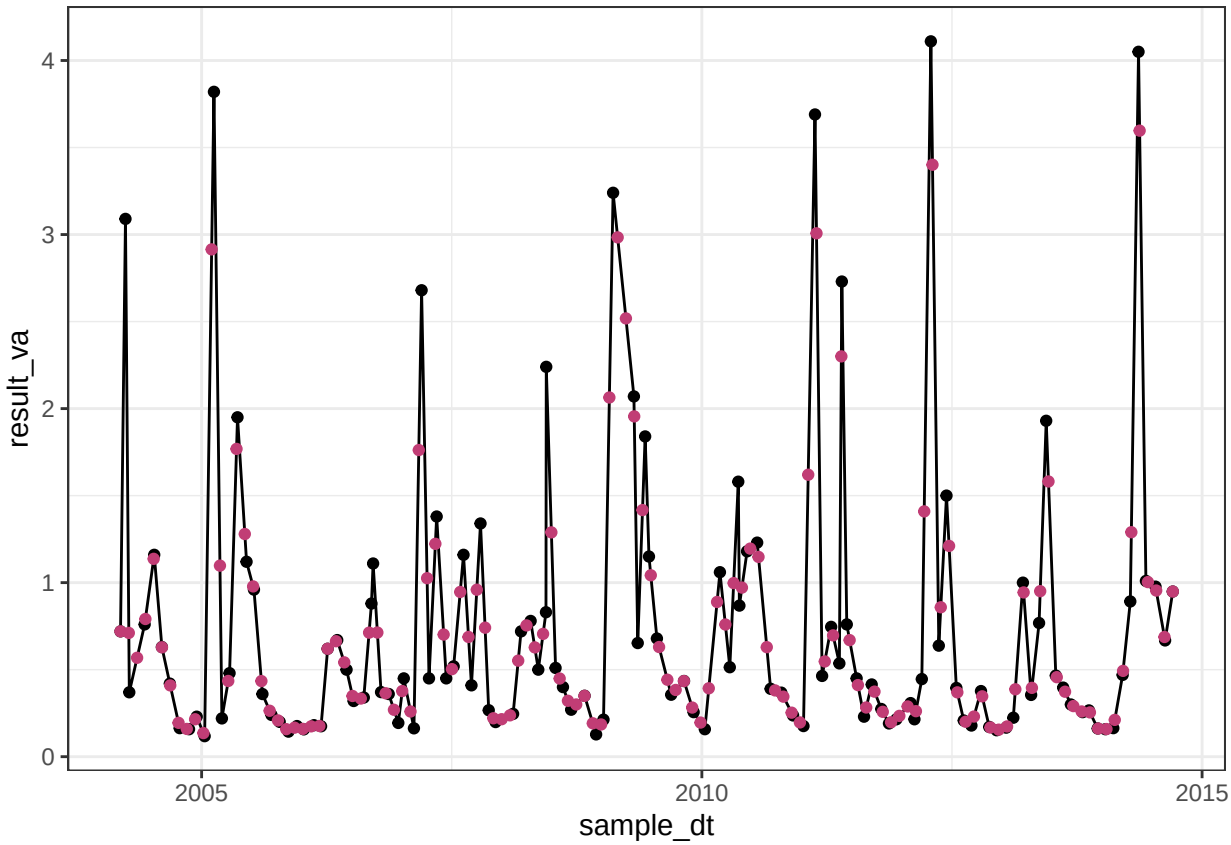
```
##
##  Seasonal Mann-Kendall trend test (Hirsch-Slack test)
##
## data:  NRHNtimeseries
## z = -0.63996, p-value = 0.5222
## alternative hypothesis: true S is not equal to 0
## sample estimates:
##    S varS
##  -28 1780
```

```
summary(NRHNtrend)
```

```
##
##  Seasonal Mann-Kendall trend test (Hirsch-Slack test)
```

```
##
## data: NRHNtimeseries
## alternative hypothesis: two.sided
##
## Statistics for individual seasons
##
## H0
##                       S varS    tau      z Pr(>|z|)
## Season 1:   S = 0  -13  125 -0.289 -1.073  0.28313
## Season 2:   S = 0  -13  125 -0.289 -1.073  0.28313
## Season 3:   S = 0   -9  165 -0.164 -0.623  0.53342
## Season 4:   S = 0  -11  165 -0.200 -0.778  0.43627
## Season 5:   S = 0   -9  165 -0.164 -0.623  0.53342
## Season 6:   S = 0   11  165  0.200  0.778  0.43627
## Season 7:   S = 0    3  165  0.055  0.156  0.87627
## Season 8:   S = 0   13  165  0.236  0.934  0.35020
## Season 9:   S = 0   17  165  0.309  1.246  0.21291
## Season 10:   S = 0  -7  125 -0.156 -0.537  0.59151
## Season 11:   S = 0  -5  125 -0.111 -0.358  0.72051
## Season 12:   S = 0  -5  125 -0.111 -0.358  0.72051
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
############################ Phosphorus Analysis ##############################
# Generate monthly values from March 2004 to September 2014
NRHPhosphoruslp <- as.data.frame(approx(NRHPhosphorus, n = 128, method = "linear"))
NRHPhosphoruslp$x <- as.Date(NRHPhosphoruslp$x, origin = "1970-01-01")
names(NRHPhosphoruslp) <- c("Date", "P")

# Inspect interpolated values
NRHPinterpolated <-
  ggplot(NRHPhosphorus, aes(x = sample_dt, y = result_va)) +
  geom_point() +
  geom_line() +
  geom_point(data = NRHPhosphoruslp, aes(x = Date, y = P), color = "#c13d75ff")
print(NRHPinterpolated)
```




```
# Generate time series (smk.test needs ts, not data.frame)
NRHPtimeseries <- ts(NRHPhosphoruslp$P, frequency = 12,
                     start = c(2004, 3, 11), end = c(2014, 9, 16))
# Run SMK test
NRHPtrend <- smk.test(NRHPtimeseries)

# Inspect results
NRHPtrend
```

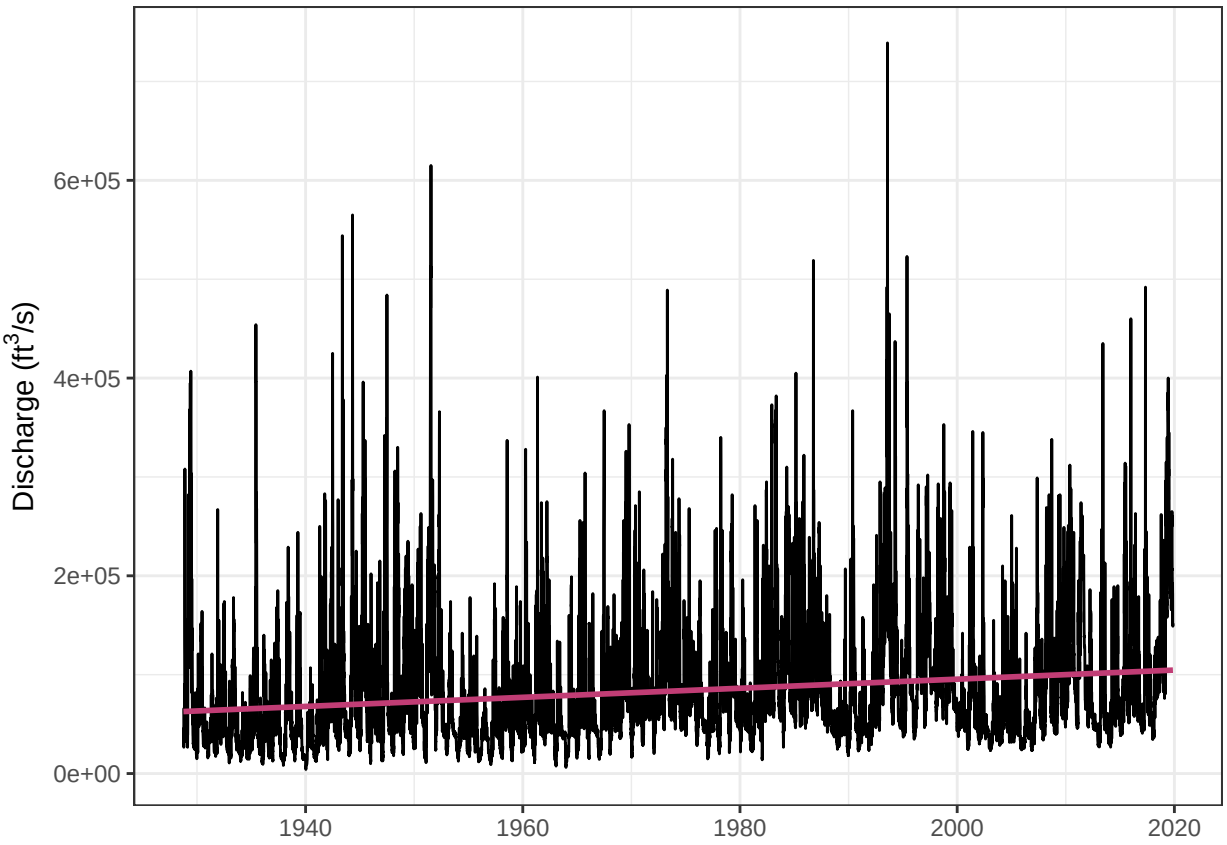
```
##
##  Seasonal Mann-Kendall trend test (Hirsch-Slack test)
##
## data:  NRHPtimeseries
## z = 0.82958, p-value = 0.4068
## alternative hypothesis: true S is not equal to 0
## sample estimates:
##    S varS
##   36 1780
```

```
summary(NRHPtrend)
```

```
##
##  Seasonal Mann-Kendall trend test (Hirsch-Slack test)
```

```
##
## data: NRHPtimeseries
## alternative hypothesis: two.sided
##
## Statistics for individual seasons
##
## H0
##                       S varS    tau      z Pr(>|z|)
## Season 1:   S = 0     3  125  0.067  0.179 0.858028
## Season 2:   S = 0   -15  125 -0.333 -1.252 0.210498
## Season 3:   S = 0    -7  165 -0.127 -0.467 0.640429
## Season 4:   S = 0     7  165  0.127  0.467 0.640429
## Season 5:   S = 0     7  165  0.127  0.467 0.640429
## Season 6:   S = 0    23  165  0.418  1.713 0.086768 .
## Season 7:   S = 0    15  165  0.273  1.090 0.275758
## Season 8:   S = 0     7  165  0.127  0.467 0.640429
## Season 9:   S = 0    -5  165 -0.091 -0.311 0.755497
## Season 10:   S = 0   -1  125 -0.022  0.000 1.000000
## Season 11:   S = 0   -1  125 -0.022  0.000 1.000000
## Season 12:   S = 0    3  125  0.067  0.179 0.858028
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

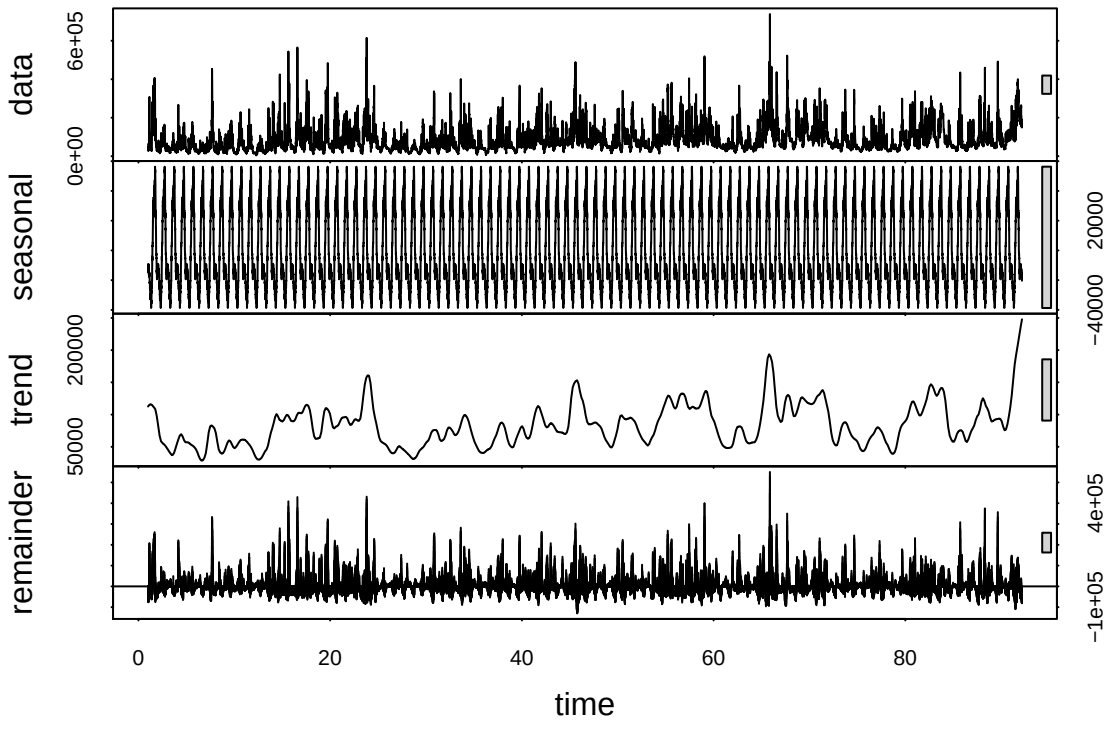
```r
# 06934500 Missouri River at Hermann, MO

############################ Discharge Analysis ##############################

MRHRegressionPlot <-
  ggplot(MRHDischarge, aes(x = Date, y = Discharge)) +
  geom_line() +
  geom_smooth(method = "lm", se = FALSE, color = "#c13d75ff") +
  labs(x = "", y = expression("Discharge (ft"^3*"/s)")) +
  theme(plot.title = element_text(margin = margin(b = -10), size = 12),
        axis.title.x = element_blank())
print(MRHRegressionPlot)
```



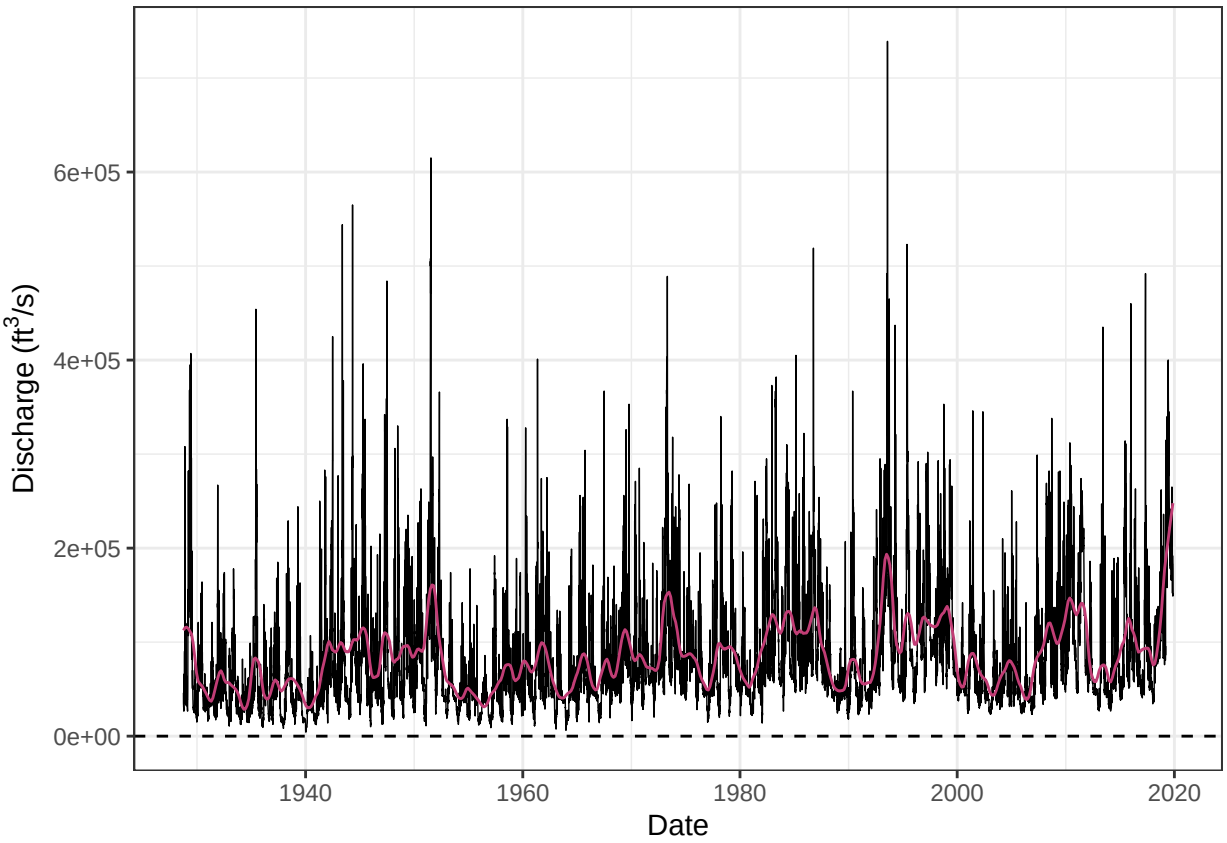

```
MRHDischarge <- na.omit(MRHDischarge)
MRH_ts <- ts(MRHDischarge[[4]], frequency = 365)
table(diff(MRHDischarge$Date))
```

```
##
##     1     3
## 33276     1
```

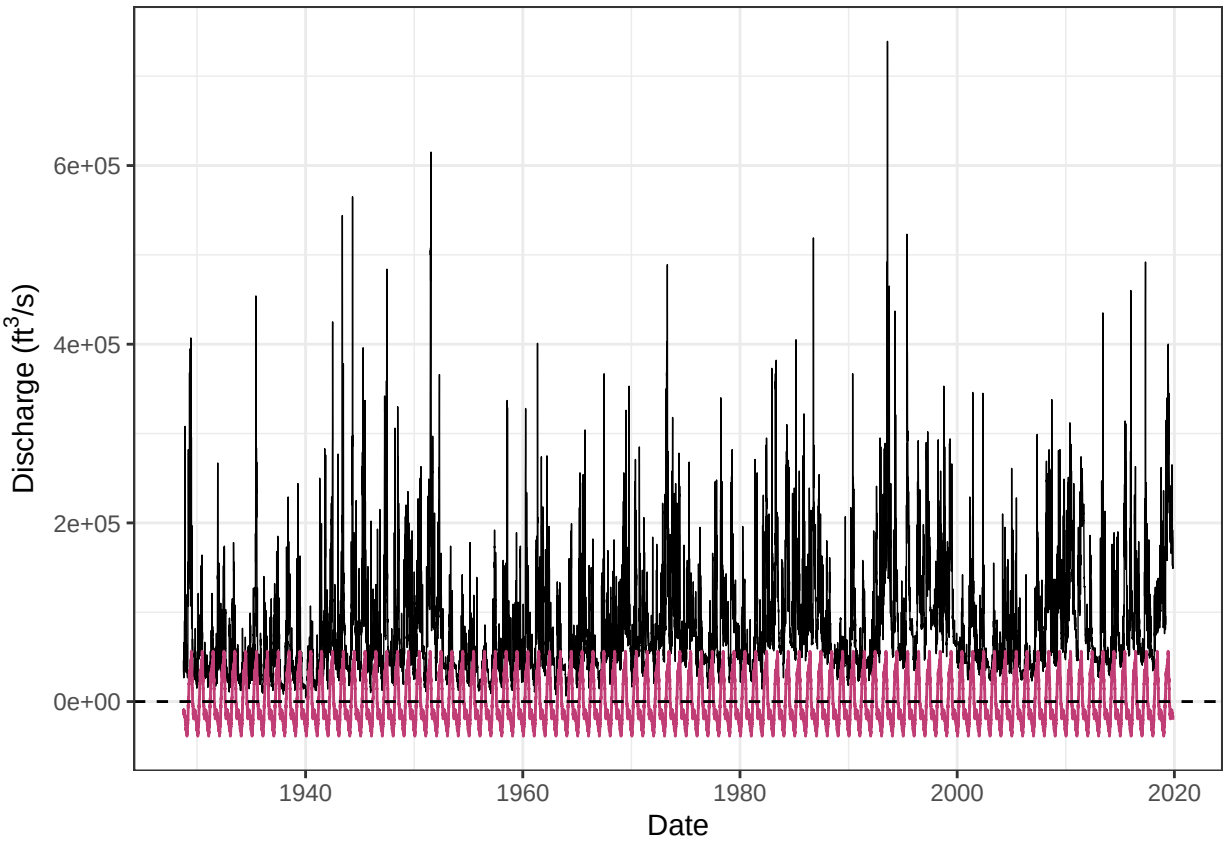
```
# Generate the decomposition
MRH_Decomposed <- stl(MRH_ts, s.window = "periodic")

# Visualize the decomposed series.
plot(MRH_Decomposed)
```




```
# We can extract the components and turn them into data frames
MRH_Components <- as.data.frame(MRH_Decomposed$time.series[,1:3])
MRH_Components <- mutate(MRH_Components,
                         Observed = MRHDischarge$Discharge,
                         Date = MRHDischarge$Date)

# Visualize how the trend maps onto the data
ggplot(MRH_Components) +
  geom_line(aes(y = Observed, x = Date),  size = 0.25) +
  geom_line(aes(y = trend, x = Date), color = "#c13d75ff") +
  geom_hline(yintercept = 0, lty = 2) +
  ylab(expression("Discharge (ft"^3*"/s)"))
```




```
# Visualize how the seasonal cycle maps onto the data
ggplot(MRH_Components) +
  geom_line(aes(y = Observed, x = Date),  size = 0.25) +
  geom_line(aes(y = seasonal, x = Date), color = "#c13d75ff") +
  geom_hline(yintercept = 0, lty = 2) +
  ylab(expression("Discharge (ft"^3*"/s)"))
```

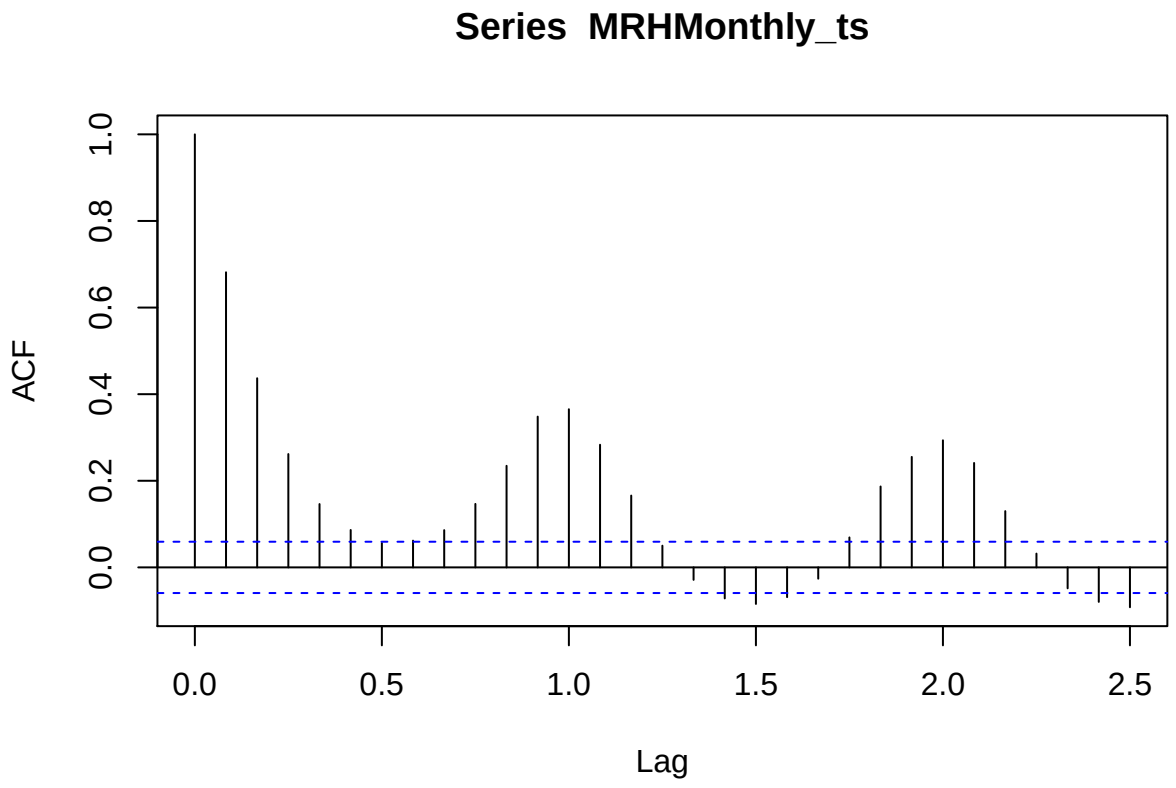



```
MRHDischarge.Monthly <- MRHDischarge %>%
  mutate(Year = year(Date),
         Month = month(Date)) %>%
  group_by(Year, Month) %>%
  summarise(Discharge = mean(Discharge))
MRHMonthly_ts <- ts(MRHDischarge.Monthly[[3]], frequency = 12)
adf.test(MRHMonthly_ts, alternative = "stationary")
```

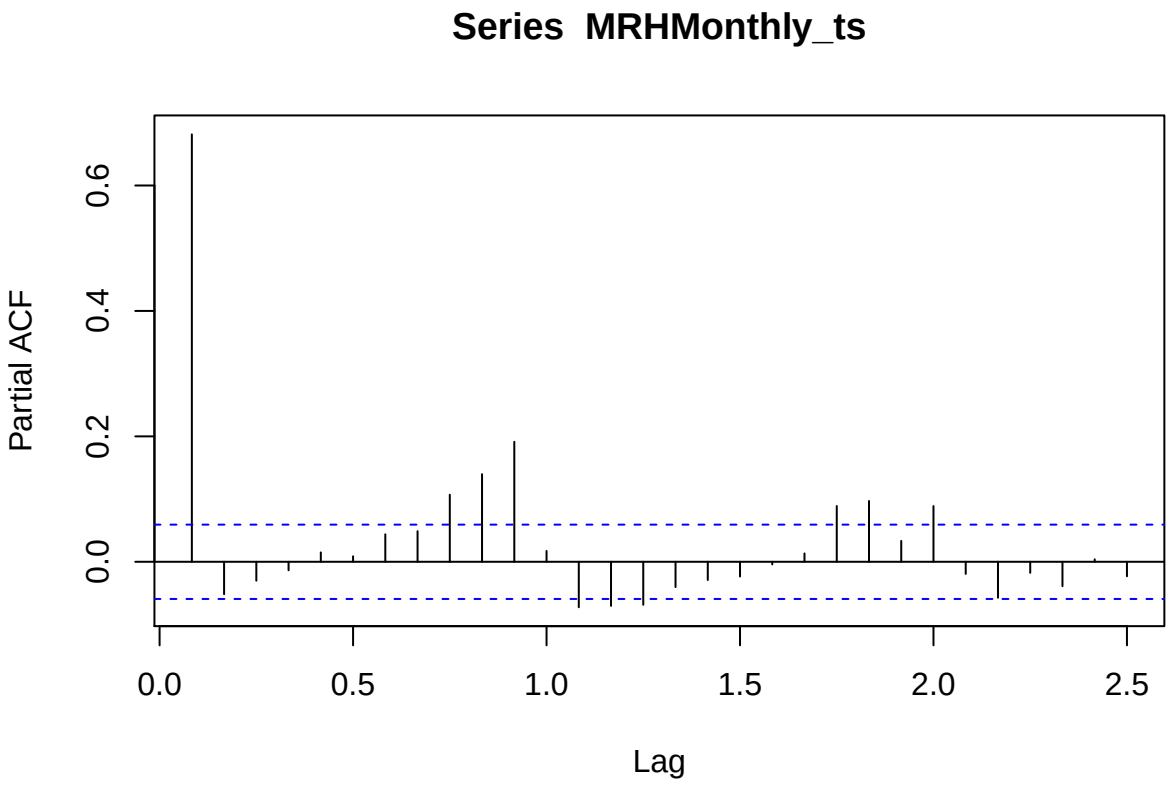
```
## Warning in adf.test(MRHMonthly_ts, alternative = "stationary"): p-value
## smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  MRHMonthly_ts
## Dickey-Fuller = -5.0751, Lag order = 10, p-value = 0.01
## alternative hypothesis: stationary
```

```
acf(MRHMonthly_ts)
```




```
pacf(MRHMonthly_ts)
```




```
# run the arima function and search for best fit
auto.arima(MRHMonthly_ts, trace = TRUE)
```

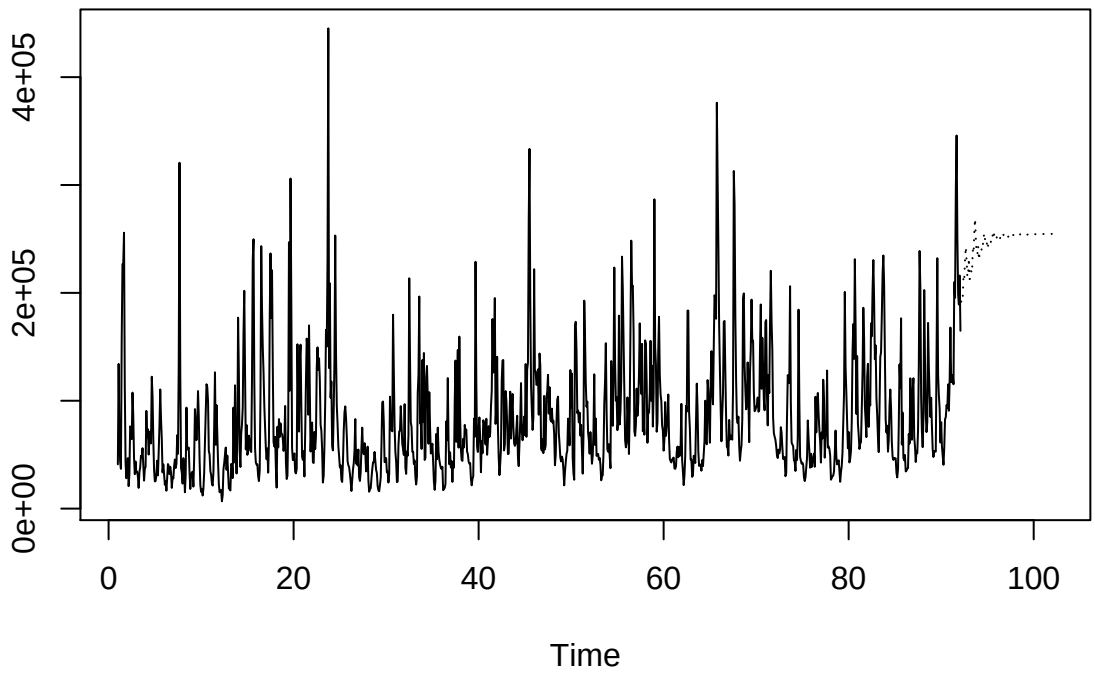
```
##
##  Fitting models using approximations to speed things up...
##
##  ARIMA(2,1,2)(1,0,1)[12] with drift         : Inf
##  ARIMA(0,1,0)            with drift         : 26461.72
##  ARIMA(1,1,0)(1,0,0)[12] with drift         : 26386.98
##  ARIMA(0,1,1)(0,0,1)[12] with drift         : 26407.7
##  ARIMA(0,1,0)                               : 26459.72
##  ARIMA(1,1,0)            with drift         : 26445.3
##  ARIMA(1,1,0)(2,0,0)[12] with drift         : 26374.86
##  ARIMA(1,1,0)(2,0,1)[12] with drift         : Inf
##  ARIMA(1,1,0)(1,0,1)[12] with drift         : Inf
##  ARIMA(0,1,0)(2,0,0)[12] with drift         : 26410.1
##  ARIMA(2,1,0)(2,0,0)[12] with drift         : 26346.77
##  ARIMA(2,1,0)(1,0,0)[12] with drift         : 26366.23
##  ARIMA(2,1,0)(2,0,1)[12] with drift         : Inf
##  ARIMA(2,1,0)(1,0,1)[12] with drift         : Inf
##  ARIMA(3,1,0)(2,0,0)[12] with drift         : 26326.8
##  ARIMA(3,1,0)(1,0,0)[12] with drift         : 26350.81
```

```
##  ARIMA(3,1,0)(2,0,1)[12] with drift         : Inf
##  ARIMA(3,1,0)(1,0,1)[12] with drift         : Inf
##  ARIMA(4,1,0)(2,0,0)[12] with drift         : 26310.75
##  ARIMA(4,1,0)(1,0,0)[12] with drift         : 26332.4
##  ARIMA(4,1,0)(2,0,1)[12] with drift         : Inf
##  ARIMA(4,1,0)(1,0,1)[12] with drift         : Inf
##  ARIMA(5,1,0)(2,0,0)[12] with drift         : 26298.17
##  ARIMA(5,1,0)(1,0,0)[12] with drift         : 26319.67
##  ARIMA(5,1,0)(2,0,1)[12] with drift         : Inf
##  ARIMA(5,1,0)(1,0,1)[12] with drift         : Inf
##  ARIMA(5,1,1)(2,0,0)[12] with drift         : 26219.71
##  ARIMA(5,1,1)(1,0,0)[12] with drift         : 26224.97
##  ARIMA(5,1,1)(2,0,1)[12] with drift         : Inf
##  ARIMA(5,1,1)(1,0,1)[12] with drift         : Inf
##  ARIMA(4,1,1)(2,0,0)[12] with drift         : Inf
##  ARIMA(5,1,2)(2,0,0)[12] with drift         : Inf
##  ARIMA(4,1,2)(2,0,0)[12] with drift         : Inf
##  ARIMA(5,1,1)(2,0,0)[12]                    : 26219.03
##  ARIMA(5,1,1)(1,0,0)[12]                    : 26224.22
##  ARIMA(5,1,1)(2,0,1)[12]                    : Inf
##  ARIMA(5,1,1)(1,0,1)[12]                    : Inf
##  ARIMA(4,1,1)(2,0,0)[12]                    : Inf
##  ARIMA(5,1,0)(2,0,0)[12]                    : 26296.19
##  ARIMA(5,1,2)(2,0,0)[12]                    : Inf
##  ARIMA(4,1,0)(2,0,0)[12]                    : 26308.76
##  ARIMA(4,1,2)(2,0,0)[12]                    : 26212.54
##  ARIMA(4,1,2)(1,0,0)[12]                    : 26225.14
##  ARIMA(4,1,2)(2,0,1)[12]                    : Inf
##  ARIMA(4,1,2)(1,0,1)[12]                    : Inf
##  ARIMA(3,1,2)(2,0,0)[12]                    : Inf
##  ARIMA(4,1,3)(2,0,0)[12]                    : 26214.31
##  ARIMA(3,1,1)(2,0,0)[12]                    : 26212.12
##  ARIMA(3,1,1)(1,0,0)[12]                    : Inf
##  ARIMA(3,1,1)(2,0,1)[12]                    : Inf
##  ARIMA(3,1,1)(1,0,1)[12]                    : Inf
##  ARIMA(2,1,1)(2,0,0)[12]                    : Inf
##  ARIMA(3,1,0)(2,0,0)[12]                    : 26324.8
##  ARIMA(2,1,0)(2,0,0)[12]                    : 26344.76
##  ARIMA(2,1,2)(2,0,0)[12]                    : Inf
##  ARIMA(3,1,1)(2,0,0)[12] with drift         : 26212.76
##
##  Now re-fitting the best model(s) without approximations...
##
##  ARIMA(3,1,1)(2,0,0)[12]                    : Inf
##  ARIMA(4,1,2)(2,0,0)[12]                    : Inf
```

```
##  ARIMA(3,1,1)(2,0,0)[12] with drift         : Inf
##  ARIMA(4,1,3)(2,0,0)[12]                    : Inf
##  ARIMA(5,1,1)(2,0,0)[12]                    : Inf
##  ARIMA(5,1,1)(2,0,0)[12] with drift         : Inf
##  ARIMA(5,1,1)(1,0,0)[12]                    : Inf
##  ARIMA(5,1,1)(1,0,0)[12] with drift         : Inf
##  ARIMA(4,1,2)(1,0,0)[12]                    : Inf
##  ARIMA(5,1,0)(2,0,0)[12]                    : 26328.41
##
##  Best model: ARIMA(5,1,0)(2,0,0)[12]
```

```
## Series: MRHMonthly_ts
## ARIMA(5,1,0)(2,0,0)[12]
##
## Coefficients:
##           ar1      ar2      ar3      ar4      ar5    sar1    sar2
##       -0.2814  -0.2466  -0.2048  -0.1659  -0.1196  0.1938  0.1783
## s.e.   0.0318   0.0319   0.0315   0.0309   0.0303  0.0312  0.0313
##
## sigma^2 estimated as 1.677e+09:  log likelihood=-13156.14
## AIC=26328.28   AICc=26328.41   BIC=26368.26
```

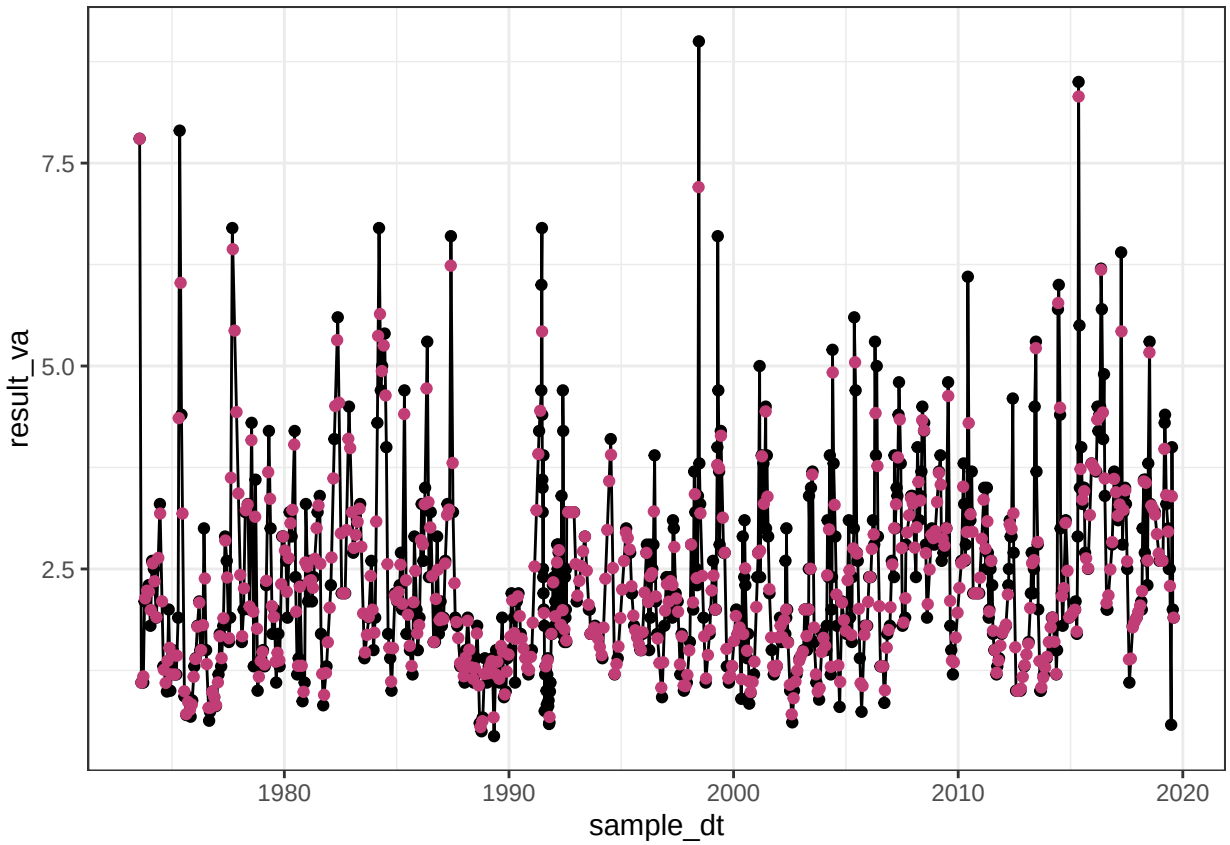
```
# create an object that defines the best fit model
MRHfit <- arima(MRHMonthly_ts, c(5, 1, 0), seasonal = list(order = c(2, 0, 0), period =

# make a prediction into the future
MRHprediction <- predict(MRHfit, n.ahead = 10*12)

# plot future predictions
ts.plot(MRHMonthly_ts, MRHprediction$pred, lty = c(1, 3))
```




```
############################ Nitrogen Analysis ##############################
# Generate monthly values from July 1973 to August 2019
MRHNitrogenlp <- as.data.frame(approx(MRHNitrogen, n = 556, method = "linear"))
```

```
## Warning in regularize.values(x, y, ties, missing(ties)): collapsing to
## unique 'x' values
```

```
MRHNitrogenlp$x <- as.Date(MRHNitrogenlp$x, origin = "1970-01-01")
names(MRHNitrogenlp) <- c("Date", "N")

# Inspect interpolated values
MRHNinterpolated <-
  ggplot(MRHNitrogen, aes(x = sample_dt, y = result_va)) +
  geom_point() +
  geom_line() +
  geom_point(data = MRHNitrogenlp, aes(x = Date, y = N), color = "#c13d75ff")
print(MRHNinterpolated)
```




```
# Generate time series (smk.test needs ts, not data.frame)
MRHNtimeseries <- ts(MRHNitrogenlp$N, frequency = 12,
                     start = c(1973, 7, 23), end = c(2019, 8, 6))
# Run SMK test
MRHNtrend <- smk.test(MRHNtimeseries)

# Inspect results
MRHNtrend
```

```
##
##  Seasonal Mann-Kendall trend test (Hirsch-Slack test)
##
## data:  MRHNtimeseries
## z = 3.8356, p-value = 0.0001253
## alternative hypothesis: true S is not equal to 0
## sample estimates:
##      S   varS
##   1412 135330
```

```
summary(MRHNtrend)
```

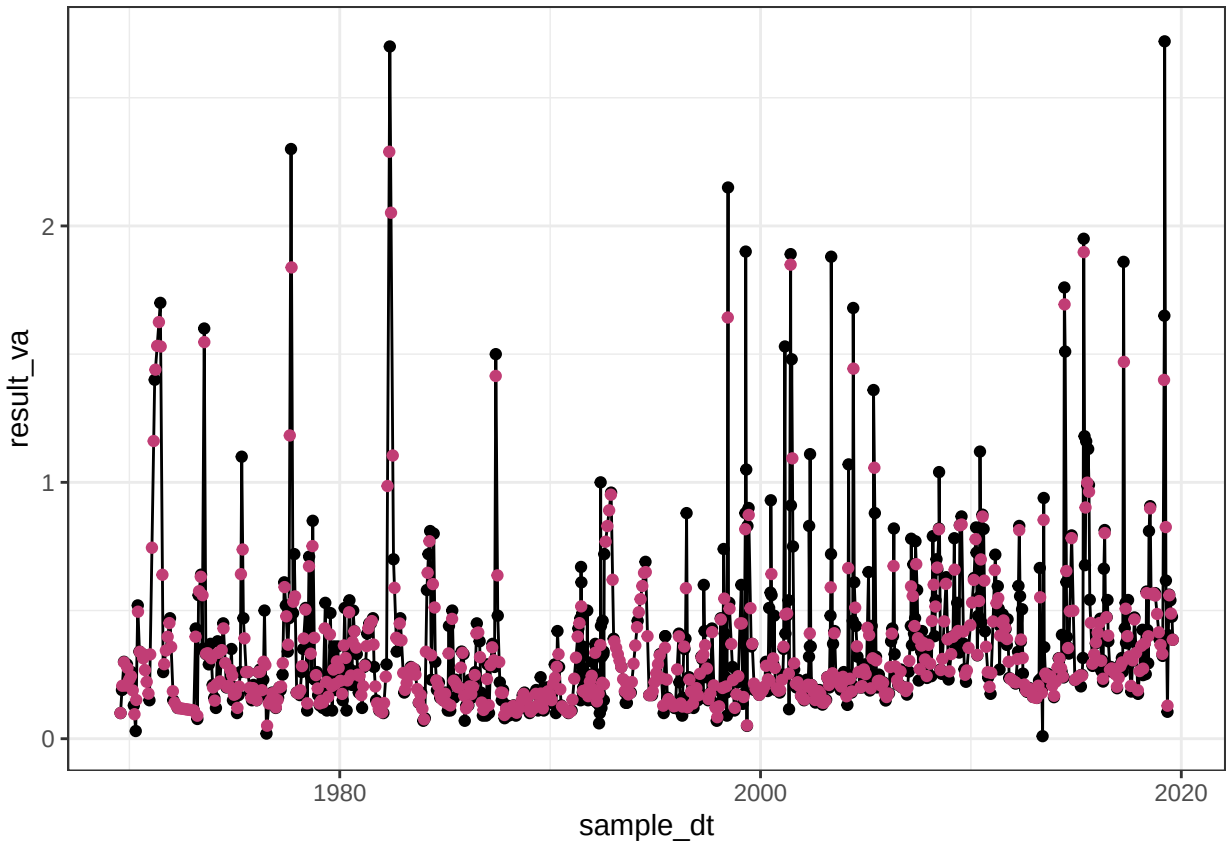
```
##
##  Seasonal Mann-Kendall trend test (Hirsch-Slack test)
```

```
##
## data: MRHNtimeseries
## alternative hypothesis: two.sided
##
## Statistics for individual seasons
##
## H0
##                       S  varS    tau      z   Pr(>|z|)
## Season 1:   S = 0    75 11155  0.072  0.701 0.48352569
## Season 2:   S = 0    77 11155  0.074  0.720 0.47178391
## Season 3:   S = 0    53 11155  0.051  0.492 0.62247625
## Season 4:   S = 0   -54 11154 -0.052 -0.502 0.61578391
## Season 5:   S = 0   -67 11155 -0.065 -0.625 0.53203799
## Season 6:   S = 0    43 11155  0.042  0.398 0.69087907
## Season 7:   S = 0   217 11891  0.201  1.981 0.04761169   *
## Season 8:   S = 0   539 11891  0.499  4.934 8.0685e-07 ***
## Season 9:   S = 0   378 11154  0.365  3.570 0.00035745 ***
## Season 10:   S = 0 103 11155  0.100  0.966 0.33416855
## Season 11:   S = 0  37 11155  0.036  0.341 0.73321390
## Season 12:   S = 0  11 11155  0.011  0.095 0.92456780
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
############################ Phosphorus Analysis ##############################
# Generate monthly values from July 1969 to August 2019
MRHPhosphoruslp <- as.data.frame(approx(MRHPhosphorus, n = 604, method = "linear"))
```

```
## Warning in regularize.values(x, y, ties, missing(ties)): collapsing to
## unique 'x' values
```

```
MRHPhosphoruslp$x <- as.Date(MRHPhosphoruslp$x, origin = "1970-01-01")
names(MRHPhosphoruslp) <- c("Date", "P")

# Inspect interpolated values
MRHPinterpolated <-
  ggplot(MRHPhosphorus, aes(x = sample_dt, y = result_va)) +
  geom_point() +
  geom_line() +
  geom_point(data = MRHPhosphoruslp, aes(x = Date, y = P), color = "#c13d75ff")
print(MRHPinterpolated)
```




```
# Generate time series (smk.test needs ts, not data.frame)
MRHPtimeseries <- ts(MRHPhosphoruslp$P, frequency = 12,
                     start = c(1969, 7, 31), end = c(2019, 8, 6))
# Run SMK test
MRHPtrend <- smk.test(MRHPtimeseries)

# Inspect results
MRHPtrend
```

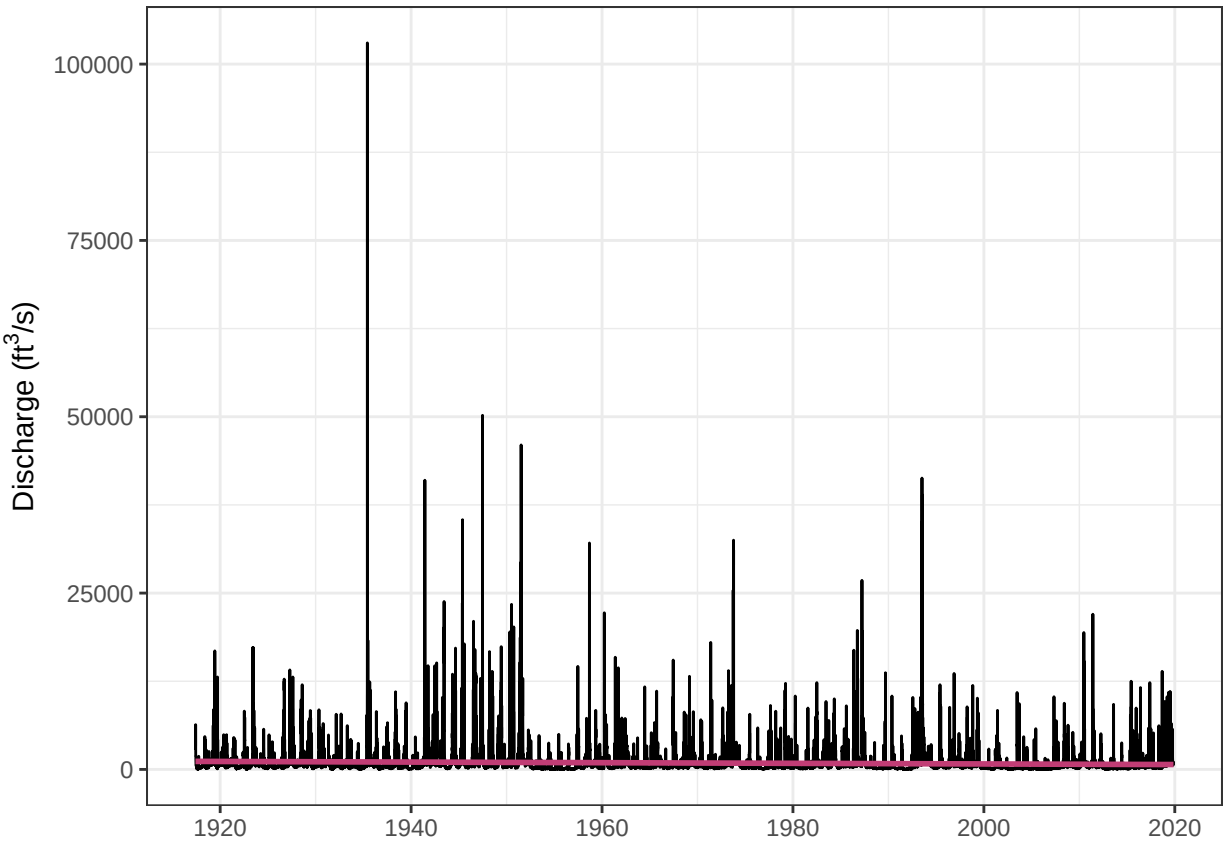
```
##
##  Seasonal Mann-Kendall trend test (Hirsch-Slack test)
##
## data:  MRHPtimeseries
## z = 6.391, p-value = 1.648e-10
## alternative hypothesis: true S is not equal to 0
## sample estimates:
##        S     varS
##   2661.0 173232.3
```

```
summary(MRHPtrend)
```

```
##
##  Seasonal Mann-Kendall trend test (Hirsch-Slack test)
```

```
##
## data: MRHPtimeseries
## alternative hypothesis: two.sided
##
## Statistics for individual seasons
##
## H0
##                       S    varS    tau      z  Pr(>|z|)
## Season 1:   S = 0  371 14291.7  0.303  3.095 0.0019681 **
## Season 2:   S = 0  189 14291.7  0.154  1.573 0.1158130
## Season 3:   S = 0  193 14291.7  0.158  1.606 0.1082623
## Season 4:   S = 0   85 14291.7  0.069  0.703 0.4822751
## Season 5:   S = 0  -35 14291.7 -0.029 -0.284 0.7760999
## Season 6:   S = 0   87 14291.7  0.071  0.719 0.4719082
## Season 7:   S = 0  181 15158.3  0.142  1.462 0.1437418
## Season 8:   S = 0  405 15158.3  0.318  3.281 0.0010330 **
## Season 9:   S = 0  249 14291.7  0.203  2.074 0.0380343  *
## Season 10:  S = 0 283 14291.7  0.231  2.359 0.0183297  *
## Season 11:  S = 0 270 14290.7  0.220  2.250 0.0244346  *
## Season 12:  S = 0 383 14291.7  0.313  3.195 0.0013965 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

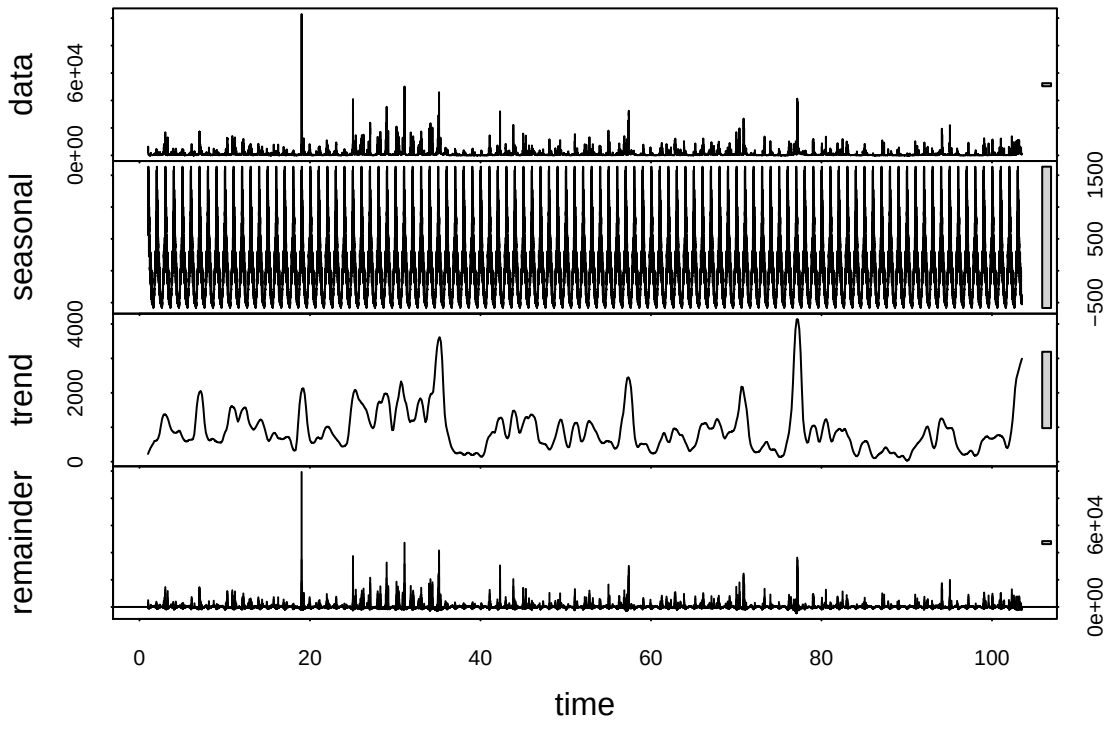
```r
# 06856600 REPUBLICAN R AT CLAY CENTER, KS

############################ Discharge Analysis ##############################
RRCCRegressionPlot <-
  ggplot(RRCCDischarge, aes(x = Date, y = Discharge)) +
  geom_line() +
  geom_smooth(method = "lm", se = FALSE, color = "#c13d75ff") +
  labs(x = "", y = expression("Discharge (ft"^3*"/s)")) +
  theme(plot.title = element_text(margin = margin(b = -10), size = 12),
        axis.title.x = element_blank())
print(RRCCRegressionPlot)
```



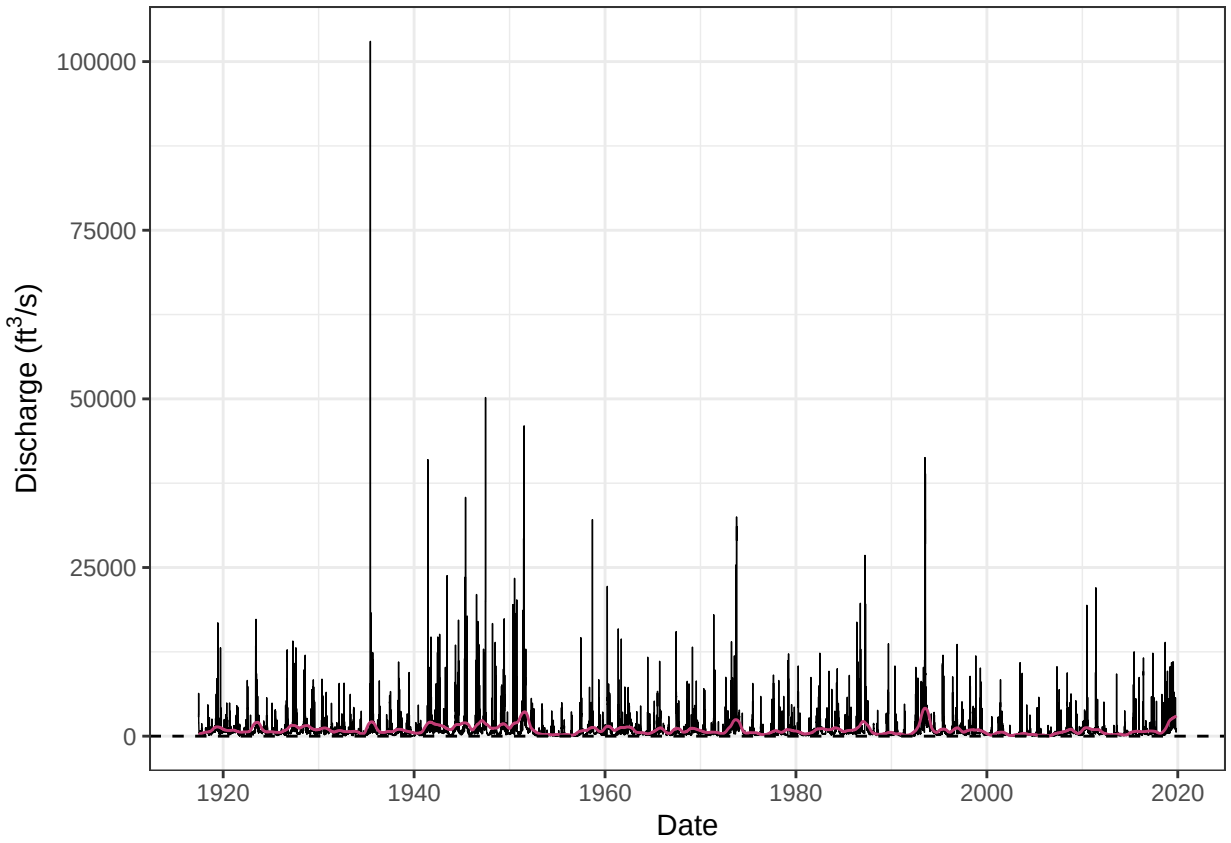

```
RRCCDischarge <- na.omit(RRCCDischarge)
RRCC_ts <- ts(RRCCDischarge[[4]], frequency = 365)
table(diff(RRCCDischarge$Date))
```

```
##
##     1
## 37419
```

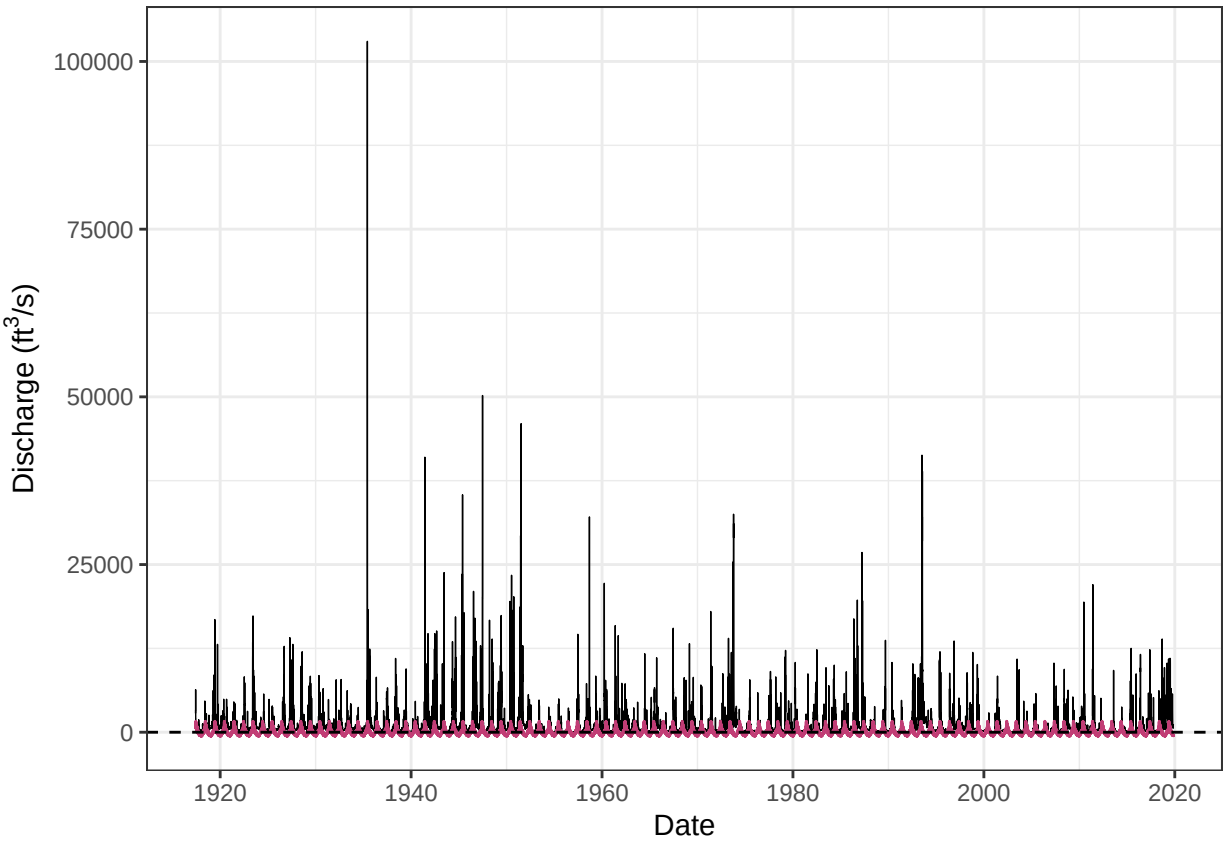
```
# Generate the decomposition
RRCC_Decomposed <- stl(RRCC_ts, s.window = "periodic")

# Visualize the decomposed series.
plot(RRCC_Decomposed)
```




```
# We can extract the components and turn them into data frames
RRCC_Components <- as.data.frame(RRCC_Decomposed$time.series[,1:3])
RRCC_Components <- mutate(RRCC_Components,
                          Observed = RRCCDischarge$Discharge,
                          Date = RRCCDischarge$Date)

# Visualize how the trend maps onto the data
ggplot(RRCC_Components) +
  geom_line(aes(y = Observed, x = Date),  size = 0.25) +
  geom_line(aes(y = trend, x = Date), color = "#c13d75ff") +
  geom_hline(yintercept = 0, lty = 2) +
  ylab(expression("Discharge (ft"^3*"/s)"))
```




```
# Visualize how the seasonal cycle maps onto the data
ggplot(RRCC_Components) +
  geom_line(aes(y = Observed, x = Date),  size = 0.25) +
  geom_line(aes(y = seasonal, x = Date), color = "#c13d75ff") +
  geom_hline(yintercept = 0, lty = 2) +
  ylab(expression("Discharge (ft"^3*"/s)"))
```

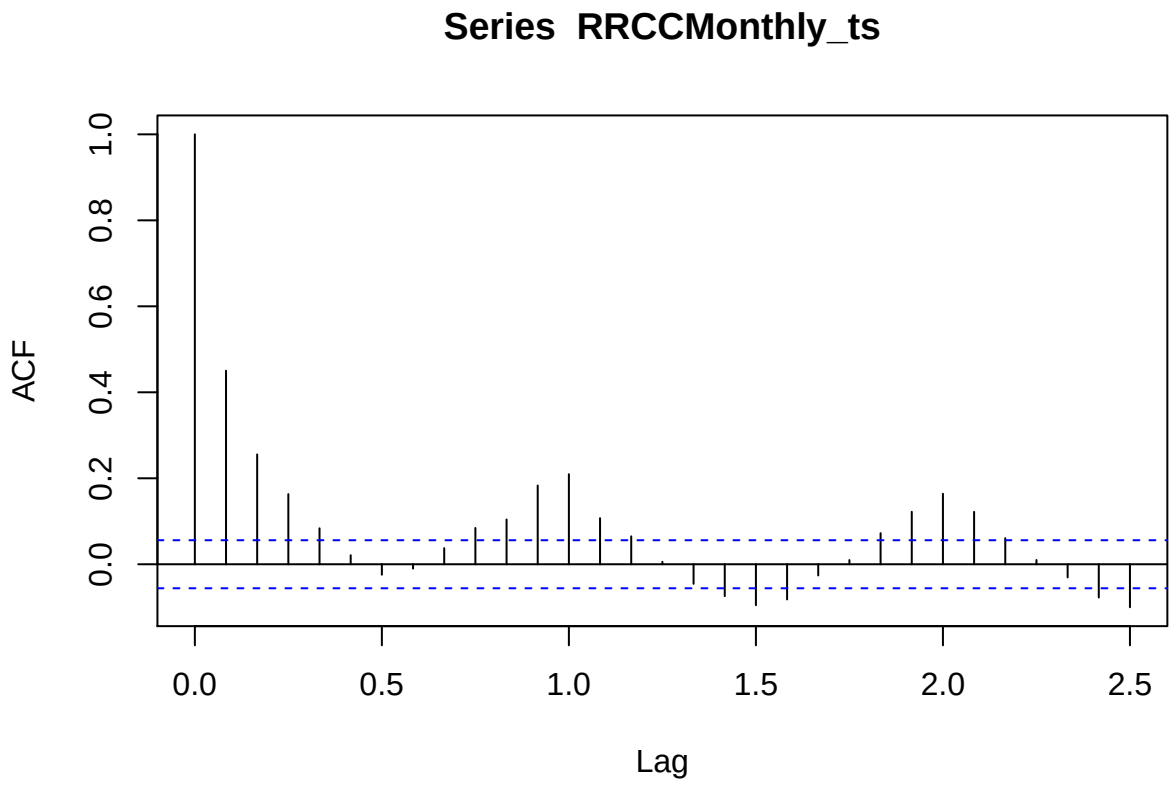



```
RRCCDischarge.Monthly <- RRCCDischarge %>%
  mutate(Year = year(Date),
         Month = month(Date)) %>%
  group_by(Year, Month) %>%
  summarise(Discharge = mean(Discharge))
RRCCMonthly_ts <- ts(RRCCDischarge.Monthly[[3]], frequency = 12)
adf.test(RRCCMonthly_ts, alternative = "stationary")
```

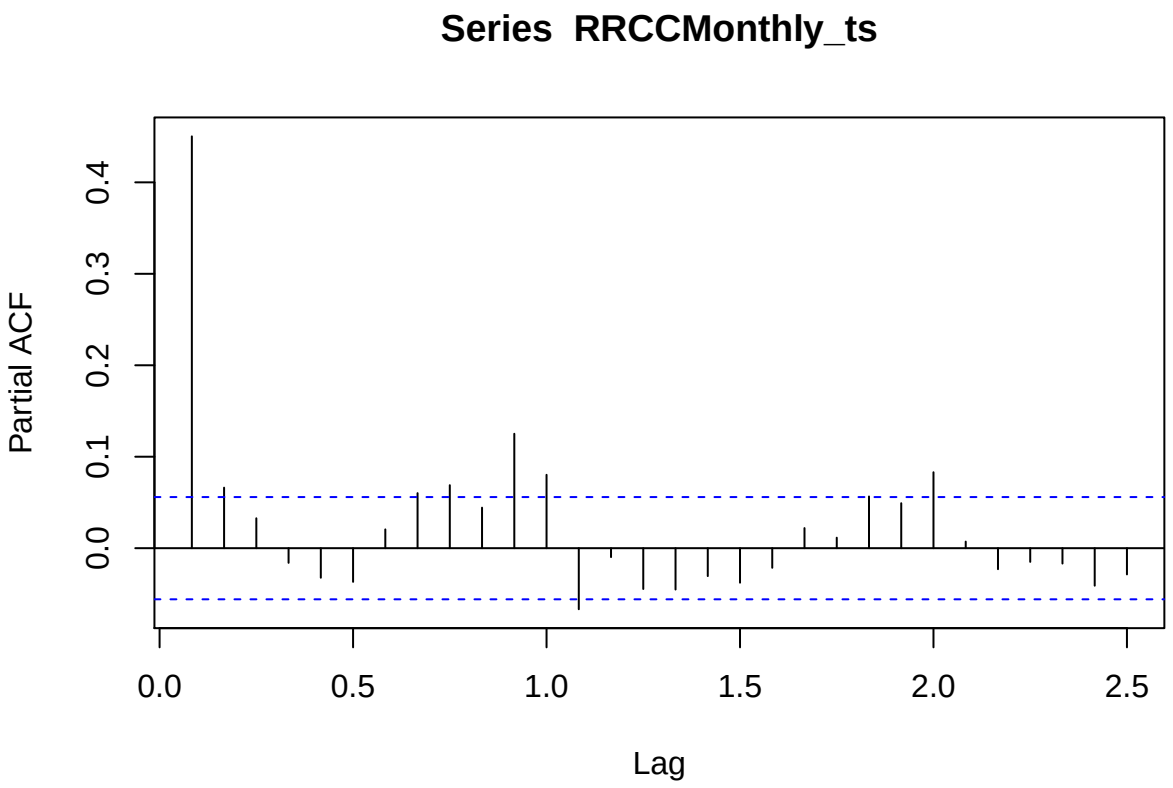
```
## Warning in adf.test(RRCCMonthly_ts, alternative = "stationary"): p-value
## smaller than printed p-value
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  RRCCMonthly_ts
## Dickey-Fuller = -7.4083, Lag order = 10, p-value = 0.01
## alternative hypothesis: stationary
```

```
acf(RRCCMonthly_ts)
```




```
pacf(RRCCMonthly_ts)
```



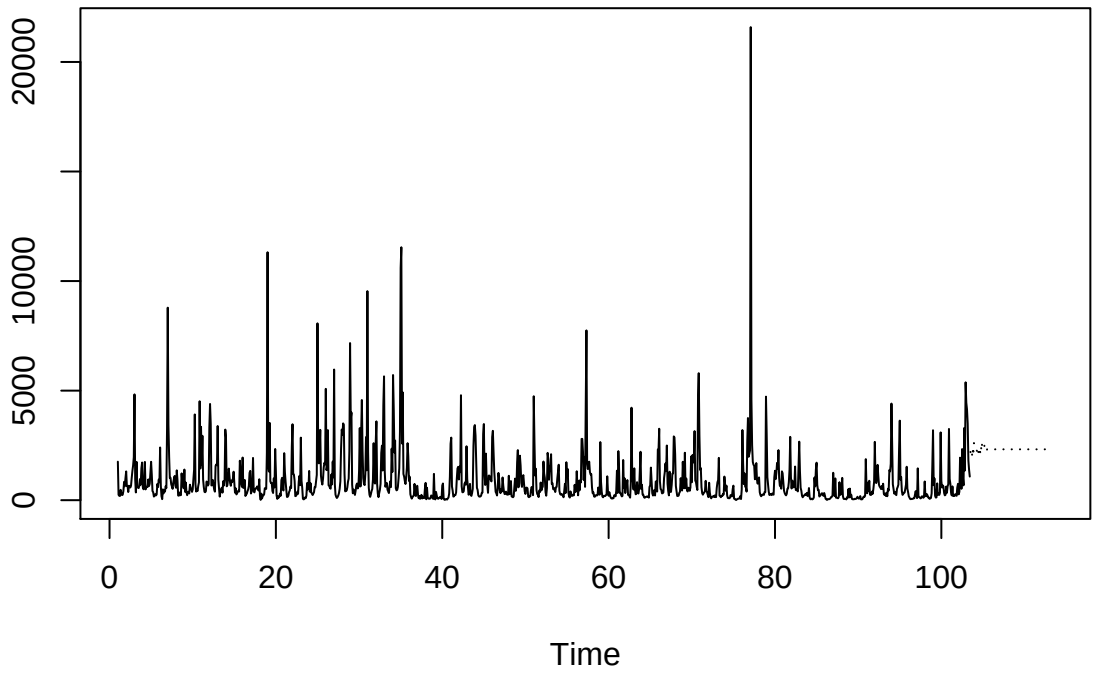

```
# run the arima function and search for best fit
auto.arima(RRCCMonthly_ts, trace = TRUE)
```

```
##
##  Fitting models using approximations to speed things up...
##
##  ARIMA(2,1,2)(1,0,1)[12] with drift         : Inf
##  ARIMA(0,1,0)            with drift         : 21233.6
##  ARIMA(1,1,0)(1,0,0)[12] with drift         : 21088.27
##  ARIMA(0,1,1)(0,0,1)[12] with drift         : 20969.16
##  ARIMA(0,1,0)                               : 21231.6
##  ARIMA(0,1,1)            with drift         : 20998.71
##  ARIMA(0,1,1)(1,0,1)[12] with drift         : Inf
##  ARIMA(0,1,1)(0,0,2)[12] with drift         : 20952.71
##  ARIMA(0,1,1)(1,0,2)[12] with drift         : Inf
##  ARIMA(0,1,0)(0,0,2)[12] with drift         : 21216.56
##  ARIMA(1,1,1)(0,0,2)[12] with drift         : 20830.75
##  ARIMA(1,1,1)(0,0,1)[12] with drift         : 20837.91
##  ARIMA(1,1,1)(1,0,2)[12] with drift         : Inf
##  ARIMA(1,1,1)(1,0,1)[12] with drift         : Inf
##  ARIMA(1,1,0)(0,0,2)[12] with drift         : 21073.73
##  ARIMA(2,1,1)(0,0,2)[12] with drift         : 20819.33
```

```
##  ARIMA(2,1,1)(0,0,1)[12] with drift         : 20827.19
##  ARIMA(2,1,1)(1,0,2)[12] with drift         : Inf
##  ARIMA(2,1,1)(1,0,1)[12] with drift         : Inf
##  ARIMA(2,1,0)(0,0,2)[12] with drift         : 21009.64
##  ARIMA(3,1,1)(0,0,2)[12] with drift         : 20819.54
##  ARIMA(2,1,2)(0,0,2)[12] with drift         : Inf
##  ARIMA(1,1,2)(0,0,2)[12] with drift         : 20832.46
##  ARIMA(3,1,0)(0,0,2)[12] with drift         : 20983.45
##  ARIMA(3,1,2)(0,0,2)[12] with drift         : 20822.26
##  ARIMA(2,1,1)(0,0,2)[12]                    : 20817.47
##  ARIMA(2,1,1)(0,0,1)[12]                    : 20825.33
##  ARIMA(2,1,1)(1,0,2)[12]                    : Inf
##  ARIMA(2,1,1)(1,0,1)[12]                    : Inf
##  ARIMA(1,1,1)(0,0,2)[12]                    : 20829.04
##  ARIMA(2,1,0)(0,0,2)[12]                    : 21007.62
##  ARIMA(3,1,1)(0,0,2)[12]                    : 20817.66
##  ARIMA(2,1,2)(0,0,2)[12]                    : Inf
##  ARIMA(1,1,0)(0,0,2)[12]                    : 21071.71
##  ARIMA(1,1,2)(0,0,2)[12]                    : 20830.76
##  ARIMA(3,1,0)(0,0,2)[12]                    : 20981.43
##  ARIMA(3,1,2)(0,0,2)[12]                    : 20820.4
##
##  Now re-fitting the best model(s) without approximations...
##
##  ARIMA(2,1,1)(0,0,2)[12]                    : Inf
##  ARIMA(3,1,1)(0,0,2)[12]                    : Inf
##  ARIMA(2,1,1)(0,0,2)[12] with drift         : Inf
##  ARIMA(3,1,1)(0,0,2)[12] with drift         : Inf
##  ARIMA(3,1,2)(0,0,2)[12]                    : Inf
##  ARIMA(3,1,2)(0,0,2)[12] with drift         : Inf
##  ARIMA(2,1,1)(0,0,1)[12]                    : Inf
##  ARIMA(2,1,1)(0,0,1)[12] with drift         : Inf
##  ARIMA(1,1,1)(0,0,2)[12]                    : Inf
##  ARIMA(1,1,1)(0,0,2)[12] with drift         : Inf
##  ARIMA(1,1,2)(0,0,2)[12]                    : Inf
##  ARIMA(1,1,2)(0,0,2)[12] with drift         : Inf
##  ARIMA(1,1,1)(0,0,1)[12] with drift         : Inf
##  ARIMA(0,1,1)(0,0,2)[12] with drift         : 20966.7
##
##  Best model: ARIMA(0,1,1)(0,0,2)[12] with drift

## Series: RRCCMonthly_ts
## ARIMA(0,1,1)(0,0,2)[12] with drift
##
## Coefficients:
```

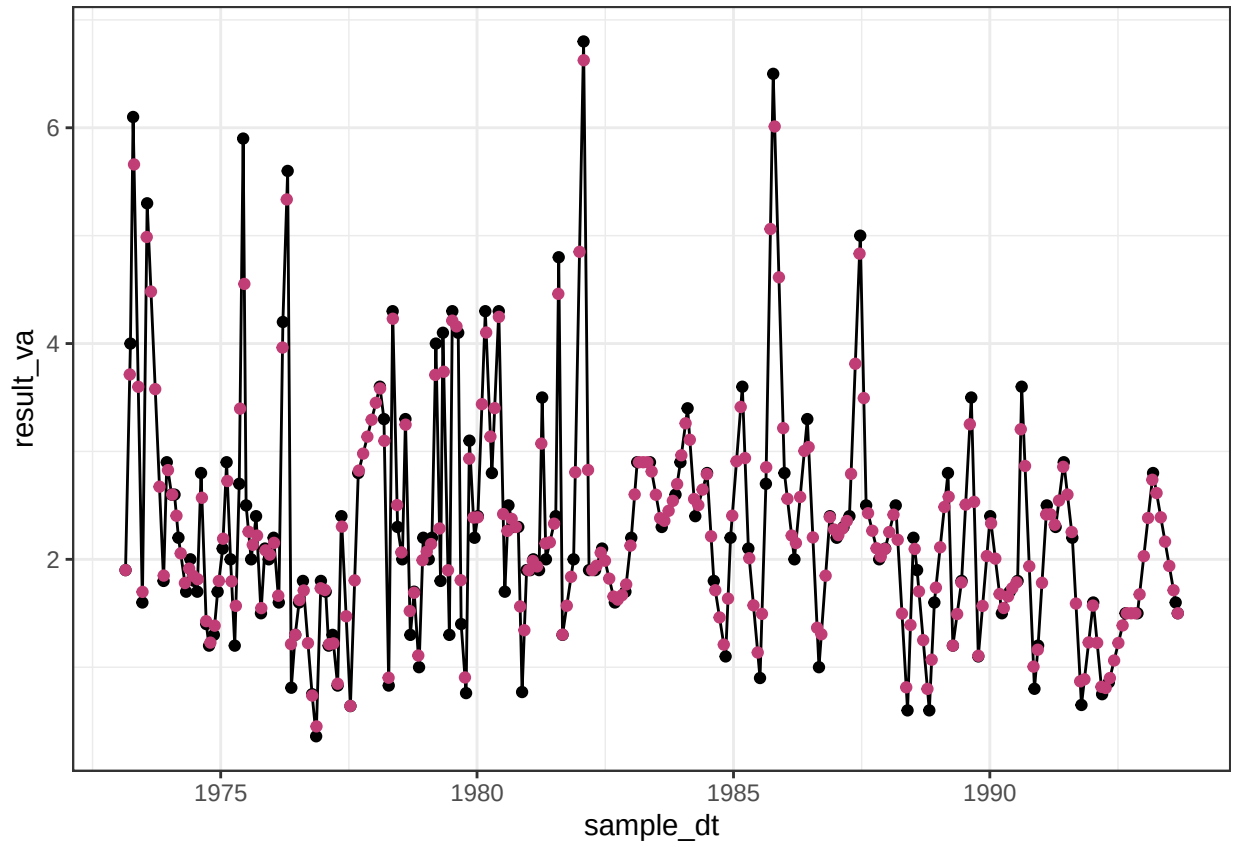
```
##           ma1    sma1    sma2    drift
##       -0.6719  0.1524  0.1086   1.4116
## s.e.   0.0403  0.0288  0.0252  14.3820
##
## sigma^2 estimated as 1492963:  log likelihood=-10478.33
## AIC=20966.66   AICc=20966.7   BIC=20992.23
```

```r
# create an object that defines the best fit model
RRCCfit <- arima(RRCCMonthly_ts, c(0, 1, 1),seasonal = list(order = c(0, 0, 2), period =

# make a prediction into the future
RRCCprediction <- predict(RRCCfit, n.ahead = 10*12)

# plot future predictions
ts.plot(RRCCMonthly_ts, RRCCprediction$pred, lty = c(1, 3))
```




```r
############################## Nitrogen Analysis ##############################
# Generate monthly values from February 1973 to September 1993
RRCCNitrogenlp <- as.data.frame(approx(RRCCNitrogen, n = 249, method = "linear"))
RRCCNitrogenlp$x <- as.Date(RRCCNitrogenlp$x, origin = "1970-01-01")
names(RRCCNitrogenlp) <- c("Date", "N")
```

```r
# Inspect interpolated values
RRCCNinterpolated <-
  ggplot(RRCCNitrogen, aes(x = sample_dt, y = result_va)) +
  geom_point() +
  geom_line() +
  geom_point(data = RRCCNitrogenlp, aes(x = Date, y = N), color = "#c13d75ff")
print(RRCCNinterpolated)
```




```r
# Generate time series (smk.test needs ts, not data.frame)
RRCCNtimeseries <- ts(RRCCNitrogenlp$N, frequency = 12,
                      start = c(1973, 2, 22), end = c(1993, 9, 1))
# Run SMK test
RRCCNtrend <- smk.test(RRCCNtimeseries)

# Inspect results
RRCCNtrend
```

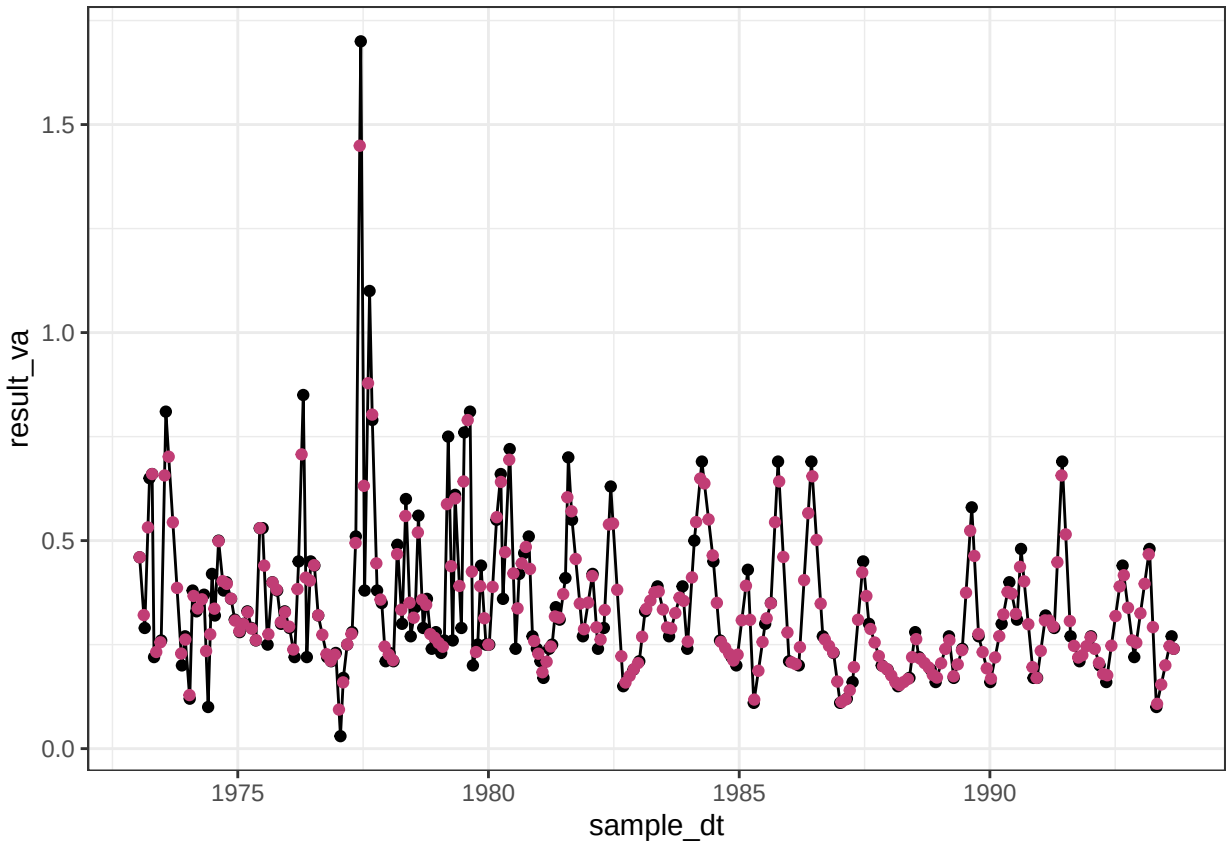
```
##
##  Seasonal Mann-Kendall trend test (Hirsch-Slack test)
##
## data:  RRCCNtimeseries
## z = -3.0233, p-value = 0.002501
```

```
## alternative hypothesis: true S is not equal to 0
## sample estimates:
##        S     varS
##  -340.00 12573.33
```

```
summary(RRCCNtrend)
```

```
##
##  Seasonal Mann-Kendall trend test (Hirsch-Slack test)
##
## data: RRCCNtimeseries
## alternative hypothesis: two.sided
##
## Statistics for individual seasons
##
## H0
##                      S   varS    tau      z Pr(>|z|)
## Season 1:   S = 0  -66  950.0 -0.347 -2.109 0.034955 *
## Season 2:   S = 0  -32 1096.7 -0.152 -0.936 0.349219
## Season 3:   S = 0  -40 1096.7 -0.190 -1.178 0.238924
## Season 4:   S = 0  -18 1096.7 -0.086 -0.513 0.607708
## Season 5:   S = 0  -60 1096.7 -0.286 -1.782 0.074811 .
## Season 6:   S = 0   -6 1096.7 -0.029 -0.151 0.879988
## Season 7:   S = 0  -12 1096.7 -0.057 -0.332 0.739764
## Season 8:   S = 0  -46 1096.7 -0.219 -1.359 0.174190
## Season 9:   S = 0   16 1096.7  0.076  0.453 0.650582
## Season 10:   S = 0  16  950.0  0.084  0.487 0.626496
## Season 11:   S = 0 -20  950.0 -0.105 -0.616 0.537603
## Season 12:   S = 0 -72  950.0 -0.379 -2.304 0.021248 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
############################## Phosphorus Analysis ##############################
# Generate monthly values from January 1973 to September 1993
RRCCPhosphoruslp <- as.data.frame(approx(RRCCPhosphorus, n = 250, method = "linear"))
RRCCPhosphoruslp$x <- as.Date(RRCCPhosphoruslp$x, origin = "1970-01-01")
names(RRCCPhosphoruslp) <- c("Date", "P")

# Inspect interpolated values
RRCCPinterpolated <-
  ggplot(RRCCPhosphorus, aes(x = sample_dt, y = result_va)) +
  geom_point() +
  geom_line() +
  geom_point(data = RRCCPhosphoruslp, aes(x = Date, y = P), color = "#c13d75ff")
print(RRCCPinterpolated)
```




```
# Generate time series (smk.test needs ts, not data.frame)
RRCCPtimeseries <- ts(RRCCPhosphoruslp$P, frequency = 12,
                      start = c(1973, 1, 16), end = c(1993, 9, 1))
# Run SMK test
RRCCPtrend <- smk.test(RRCCPtimeseries)

# Inspect results
RRCCPtrend
```

```
##
##  Seasonal Mann-Kendall trend test (Hirsch-Slack test)
##
## data:  RRCCPtimeseries
## z = -4.7614, p-value = 1.923e-06
## alternative hypothesis: true S is not equal to 0
## sample estimates:
##     S  varS
##  -538 12720
```

```
summary(RRCCPtrend)
```

```
##
##  Seasonal Mann-Kendall trend test (Hirsch-Slack test)
```

```
##
## data: RRCCPtimeseries
## alternative hypothesis: two.sided
##
## Statistics for individual seasons
##
## H0
##                       S   varS    tau      z Pr(>|z|)
## Season 1:   S = 0  -42 1096.7 -0.200 -1.238 0.215689
## Season 2:   S = 0  -32 1096.7 -0.152 -0.936 0.349219
## Season 3:   S = 0  -54 1096.7 -0.257 -1.600 0.109502
## Season 4:   S = 0  -68 1096.7 -0.324 -2.023 0.043053 *
## Season 5:   S = 0  -40 1096.7 -0.190 -1.178 0.238924
## Season 6:   S = 0  -52 1096.7 -0.248 -1.540 0.123550
## Season 7:   S = 0  -60 1096.7 -0.286 -1.782 0.074811 .
## Season 8:   S = 0  -62 1096.7 -0.295 -1.842 0.065473 .
## Season 9:   S = 0  -54 1096.7 -0.257 -1.600 0.109502
## Season 10:   S = 0  -4  950.0 -0.021 -0.097 0.922462
## Season 11:   S = 0 -20  950.0 -0.105 -0.616 0.537603
## Season 12:   S = 0 -50  950.0 -0.263 -1.590 0.111887
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

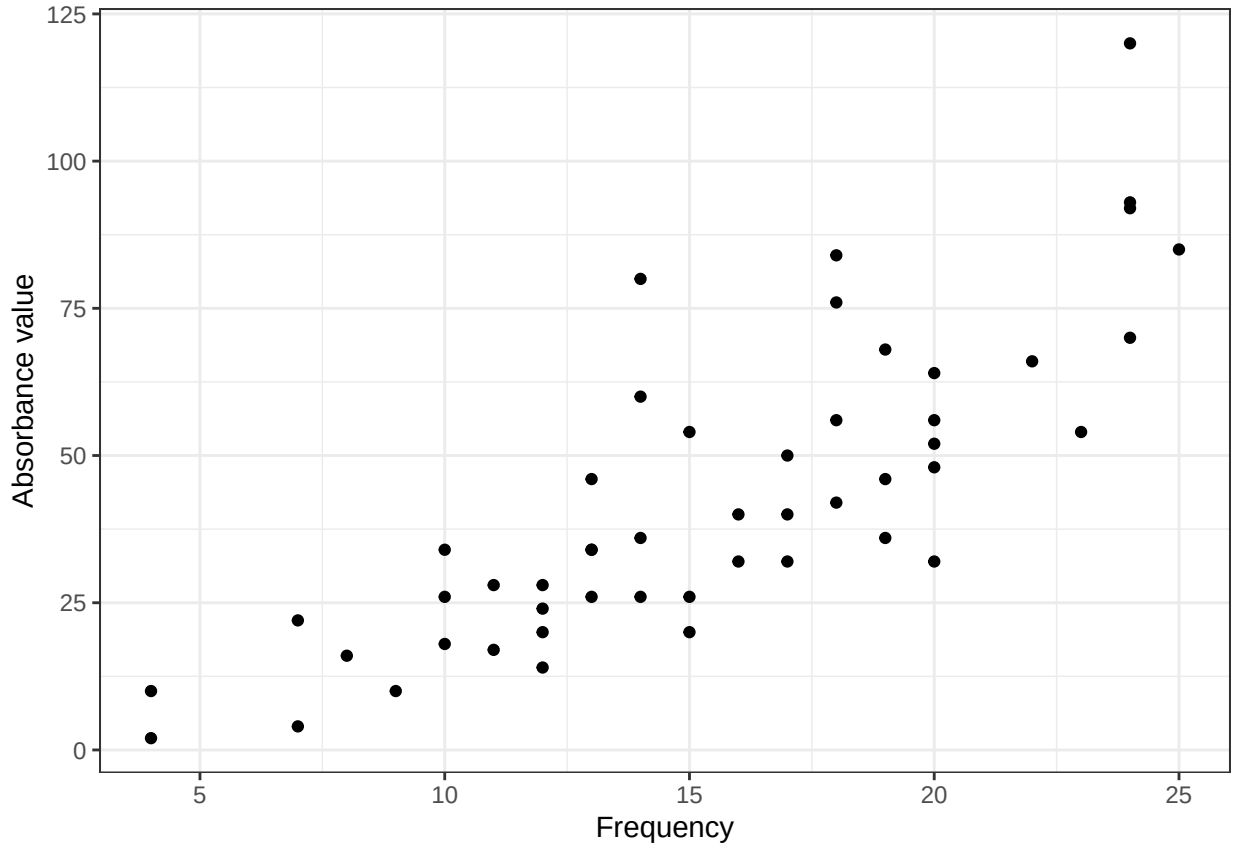


Figure 13: Absorbance frequency

5 Summary and Conclusions

<Summarize your major findings from your analyses. What conclusions do you draw from your findings? Make sure to apply this to a broader application for the research question you have answered.>

5.1 Example for autoreferencing

As seen by Figure 13, Absorbance values are not normally distributed. This is expected, as we are dealing with ecological data.