12: Time Series Analysis

Hydrologic Data Analysis | Kateri Salk

Fall 2019

Lesson Objectives

1. Choose appropriate time series analyses for trend detection and forecasting
2. Discuss the influence of seasonality on time series analysis
3. Interpret and communicate results of time series analyses

Session Set Up

```
getwd()
```

```
## [1] "/Users/katerisalk/Box Sync/Courses/Hydrologic Data Analysis/Lessons"
```

```
library(tidyverse)
library(lubridate)
library(dataRetrieval)
library(LAGOSNE)
#install.packages("trend")
library(trend)
#install.packages("forecast")
library(forecast)
#install.packages(tseries)
library(tseries)

theme_set(theme_classic())

LAGOSdata <- lagosne_load()
```

```
## Warning in `_f`(version = version, fpath = fpath): LAGOSNE version
## unspecified, loading version: 1.087.3
```

Trend analysis

Two types of trends may be present in our time series dataset: **monotonic** or **step**. Monotonic trends are a gradual shift over time that is consistent in direction, for example in response to land use change. Step trends are a distinct shift at a given time point, for example in response to a policy being enacted.

Step trend analysis

Step trend analysis works well for upstream/downstream and before/after study design. We will not delve into these methods during class, but specific tests are listed below for future reference.

Note: ALWAYS look into the assumptions of a given test to ensure it matches with your data and with your research question.

- **Change point detection**, e.g., `pettitt.test` (package: trend) or `breakpoints` (package: strucchange)

- **t-test (paired or unpaired)**
- **Kruskal-Wallis test**: non-parametric version of t-test
- **ANCOVA**, analysis of covariance

Monotonic trend analysis

In general, detecting a monotonic trend requires a long sequence of data with few gaps. If we are working with monthly data, a time series of at least five years is recommended. Gaps can be accounted for, but a gap that makes up more than 1/3 of the sampling period is generally considered the threshold for considering a gap to be too long (a step trend analysis might be better in this situation).

Adjusting the data may be necessary to fulfill the assumptions of a trend test. These adjustments include **aggregation**, **subsampling**, and **interpolation**. What do each of these mean, and why might we want to use them?

aggregation:

subsampling:

interpolation:

Specific tests for monotonic trend analysis are listed below, with assumptions and tips:

- **linear regression**: no seasonality, fits the assumptions of a parametric test. Function: `lm`
- **Mann-Kendall**: no seasonality, non-parametric, no temporal autocorrelation, missing data allowed. Function: `mk.test` (package: trend)
- **modified Mann-Kendall**: no seasonality, non-parametric, accounts for temporal autocorrelation, missing data allowed. Function: `mmky` and `mmkh` (package: modifiedmk)
- **Seasonal Mann-Kendall**: seasonality, non-parametric, no temporal autocorelation, identical distribution. Function: `smk.test` (package: trend)

The packages trend, Kendall, and modifiedmk also include other modifications to monotonic trend tests. Look into the documentation for these packages if you are applying a special case.

If covariates (another predictor variable) are included in the dataset, additional tests are recommended. A great resource for trend testing for water quality monitoring, which includes guidance on these cases, has been prepared by the Environmental Protection Agency: https://www.epa.gov/sites/production/files/2016-05/documents/tech_notes_6_dec2013_trend.pdf

Trend test example: TP in Lake Mendota

Lake Mendota (Wisconsin, USA) is often considered the birthplace of limnology and the most well-studied lake in the world. It has been sampled for over 100 years, and several parts of this dataset are included in the LAGOSNE database.

Today we will work with total phosphorus data from Lake Mendota to determine whether there has been a monotonic trend in these concentrations over time.

```
# create a dataframe for nutrients
LAGOSnutrient <- LAGOSdata$epi_nutr

# what information is available for Lake Mendota?
lake_info(name = "Lake Mendota", state = "Wisconsin")
```

```
##   lagoslakeid     nhdid  nhd_lat  nhd_long   lagosname1 meandepth
## 1        5371 143249470 43.10773 -89.42273 LAKE MENDOTA      12.8
##   meandepthsource maxdepth maxdepthsource            legacyid    gnis_name
## 1  WI_LTER_SECCHI     25.3 WI_LTER_SECCHI 805400;LAKE MENDOTA Lake Mendota
```

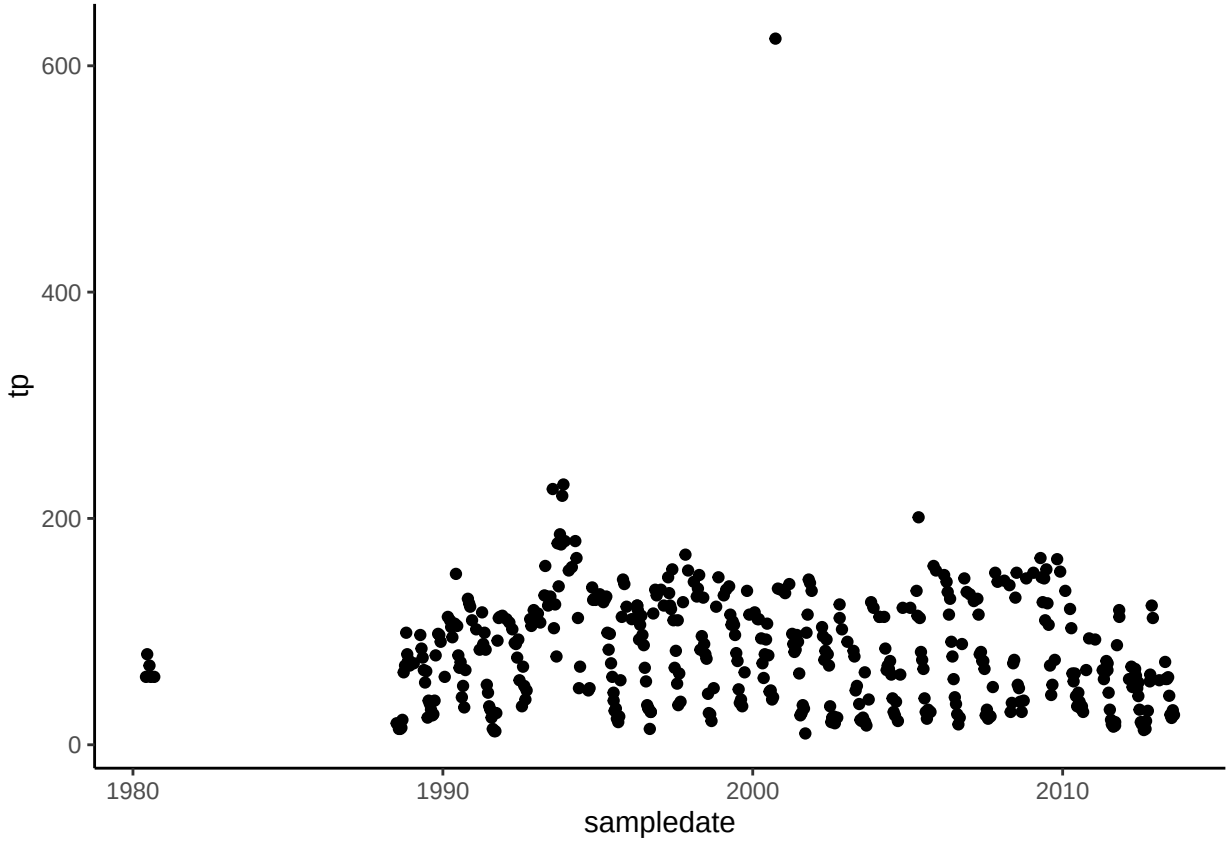
```
##   lake_area_ha lake_perim_meters nhd_fcode nhd_ftype iws_zoneid hu4_zoneid
## 1     3961.152          38422.28     39004       390  IWS_32193     HU4_34
##   hu6_zoneid hu8_zoneid hu12_zoneid edu_zoneid county_zoneid state_zoneid
## 1     HU6_46    HU8_186  HU12_11998     EDU_57    County_912      State_9
##   elevation_m state state_name state_lat state_long state_pct_in_nwi
## 1     257.987    WI  Wisconsin  44.63733  -90.01184              100
##   state_ha_in_nwi state_ha lakeconnection   iws_ha
## 1        14529517 14529517  DR_LakeStream 24360.32
```

```
  # lake_info can return info about a lake if you input name + state or lake id.

# Wrangle a dataset with just Lake Mendota TP time series
Mendotadata <- LAGOSnutrient %>%
  filter(lagoslakeid == 5371) %>%
  select(sampledate, tp)

# What do these data look like?
MendotaTP <-
ggplot(Mendotadata, aes(x = sampledate, y = tp)) +
  geom_point()
print(MendotaTP)
```

```
## Warning: Removed 10 rows containing missing values (geom_point).
```



It is crucially important to visualize your time series before moving forward with any test. In this case, we notice two major issues:

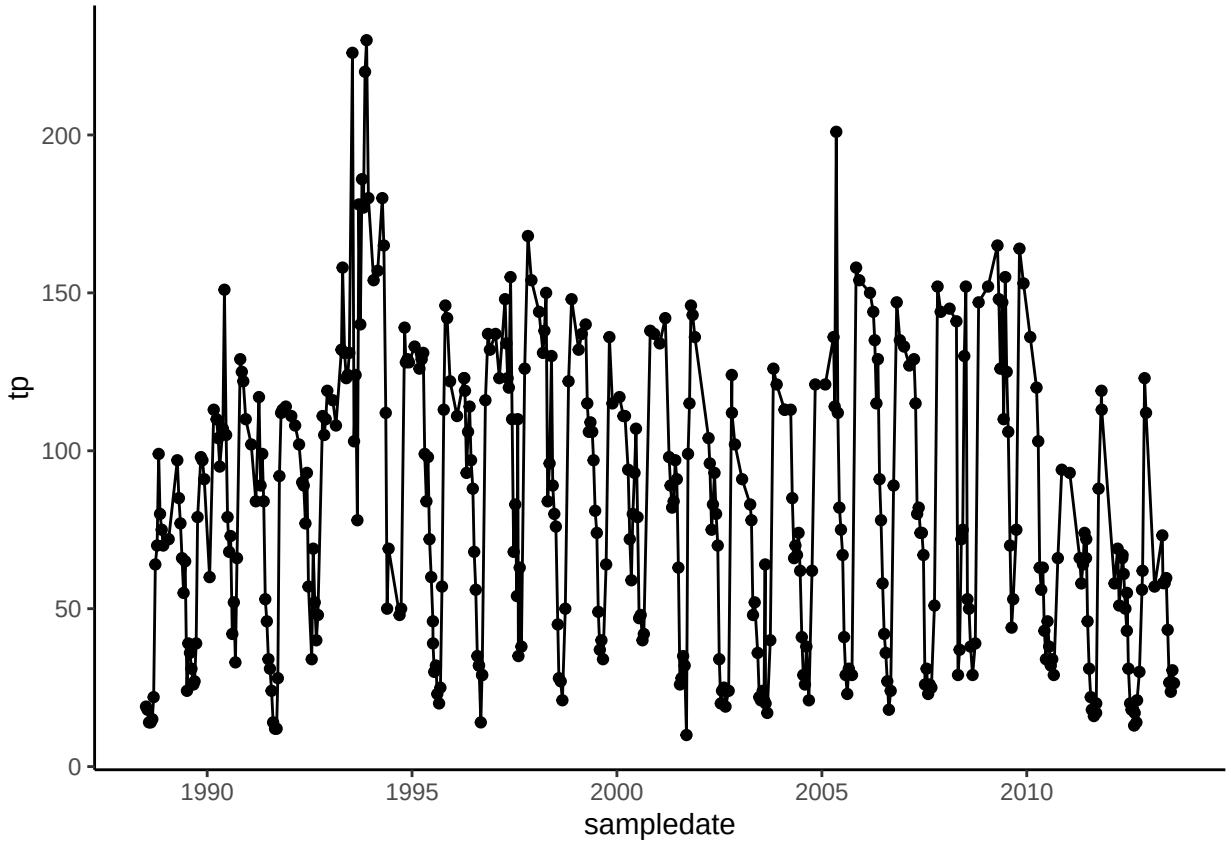
1. There is a large gap in monitoring in the 1980s

2. There is an unusually large TP value in September 2000

Detecting whether the outlier is a true measurement or a mistake in the dataset would be recommended at this point. Regardless, we know that a single large point that is more than 3x the value of the next largest value in the dataset will interfere with answering our question, particularly if we use this value for interpolation. Let's choose to leave this value out of the dataset for our purposes today (a more thorough QA/QC check would be recommended though).

```
# Remove issues 1 and 2, arrange by date
Mendotadata <- Mendotadata %>%
  filter(tp < 300 & sampledate > "1988-01-01") %>%
  arrange(sampledate)

# Re-plot data
MendotaTP <-
ggplot(Mendotadata, aes(x = sampledate, y = tp)) +
  geom_point() +
  geom_line()
print(MendotaTP)
```



We see distinct seasonality, with higher TP values occurring in the winter compared to summer. Therefore, we will proceed with a **Seasonal Mann-Kendall** test.

We see that TP data were collected somewhere between biweekly and monthly across the sampling period. However, the SMK test requires identically distributed data. We will therefore interpolate the data to generate monthly values for TP.

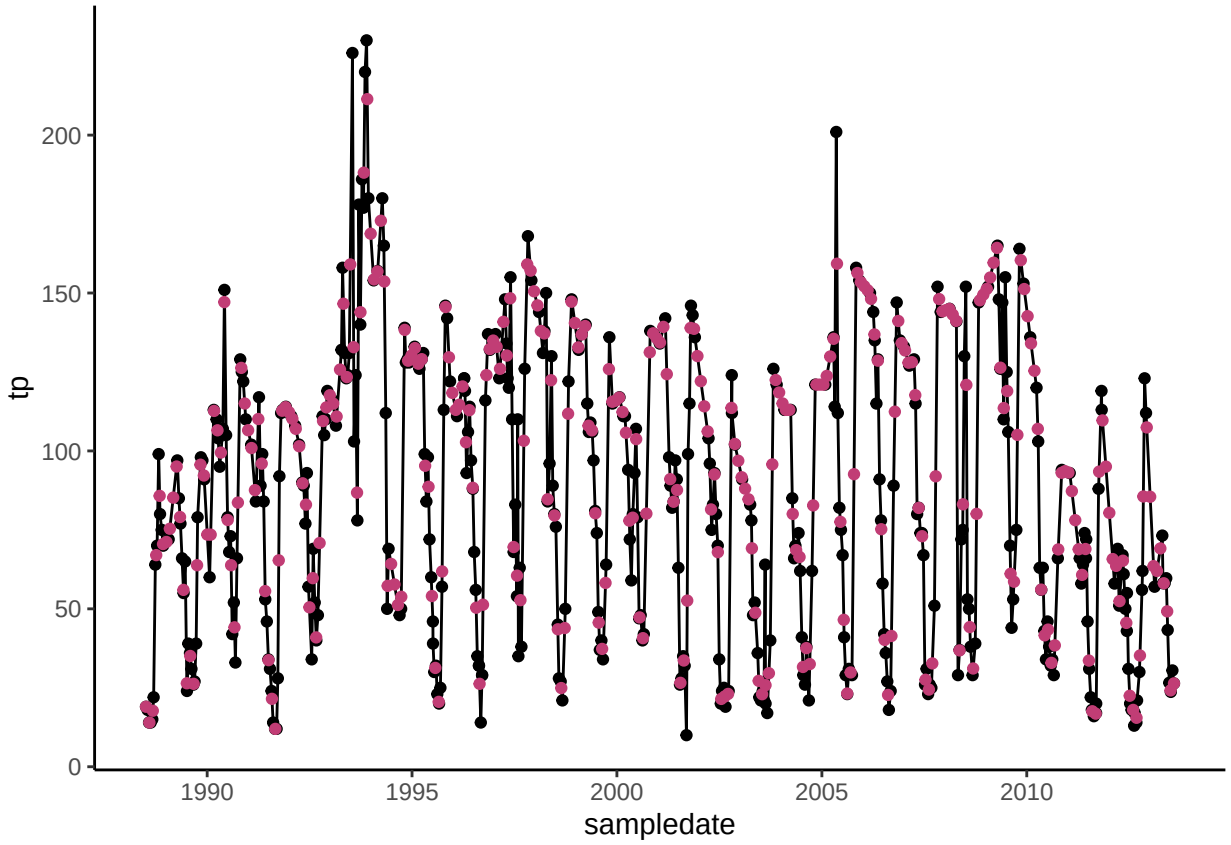
Common interpolation methods:

- **Piecewise constant**:
- **Linear**:
- **Spline**:

Linear interpolation is most common for water quality data, and fits with our understanding about how TP might change over time in this system.

```
# Generate monthly values from July 1988 to August 2013
linearinterpolation <- as.data.frame(approx(Mendotadata, n = 303, method = "linear"))
linearinterpolation$x <- as.Date(linearinterpolation$x, origin = "1970-01-01")
names(linearinterpolation) <- c("Date", "TP")

# Inspect interpolated values
MendotaTPinterpolated <-
ggplot(Mendotadata, aes(x = sampledate, y = tp)) +
  geom_point() +
  geom_line() +
  geom_point(data = linearinterpolation, aes(x = Date, y = TP), color = "#c13d75ff")
print(MendotaTPinterpolated)
```




```
# Generate time series (smk.test needs ts, not data.frame)
Mendotatimeseries <- ts(linearinterpolation$TP, frequency = 12,
                        start = c(1988, 7, 5), end = c(2013, 8, 5))
# Run SMK test
Mendotatrend <- smk.test(Mendotatimeseries)

# Inspect results
```

```
Mendotatrend
```

```
##
##  Seasonal Mann-Kendall trend test (Hirsch-Slack test)
##
## data:  Mendotatimeseries
## z = -1.4483, p-value = 0.1475
## alternative hypothesis: true S is not equal to 0
## sample estimates:
##     S  varS
##  -218 22450
```

```
summary(Mendotatrend)
```

```
##
##  Seasonal Mann-Kendall trend test (Hirsch-Slack test)
##
## data: Mendotatimeseries
## alternative hypothesis: two.sided
##
## Statistics for individual seasons
##
## H0
##                       S   varS    tau      z   Pr(>|z|)
## Season 1:   S = 0    56 1833.3  0.187  1.285 0.19895893
## Season 2:   S = 0    42 1833.3  0.140  0.958 0.33828785
## Season 3:   S = 0    28 1833.3  0.093  0.631 0.52831247
## Season 4:   S = 0    -6 1833.3 -0.020 -0.117 0.90703848
## Season 5:   S = 0   -30 1833.3 -0.100 -0.677 0.49821939
## Season 6:   S = 0   -40 1833.3 -0.133 -0.911 0.36237770
## Season 7:   S = 0    31 2058.3  0.095  0.661 0.50845423
## Season 8:   S = 0   -41 2058.3 -0.126 -0.882 0.37795960
## Season 9:   S = 0   -28 1833.3 -0.093 -0.631 0.52831247
## Season 10:   S = 0 -164 1833.3 -0.547 -3.807 0.00014074 ***
## Season 11:   S = 0 -114 1833.3 -0.380 -2.639 0.00831237  **
## Season 12:   S = 0   48 1833.3  0.160  1.098 0.27234271
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

What would we conclude based on these findings? Describe as you would to an educated but non-specialist audience.

If a significant trend was present, we could compute a **Sen's Slope** to quantify that trend (`sens.slope` function in the trend package).

Autoregressive and Moving Average Models (ARMA)

We might be interested in characterizing a time series in order to understand what happened in the past and to effectively forecast into the future. Two common models that can approximate time series are **autoregressive** and **moving average** models. To classify these models, we use the **ACF (autocorrelation function)** and the **PACF (partial autocorrelation function)**, which correspond to the autocorrelation of a series and the correlation of the residuals, respectively.

Autoregressive models operate under the framework that a given measurements is correlated with previous measurements. For example, an AR1 formulation dictates that a measurement is dependent on the previous measurement, and the value can be predicted by quantifying the lag.

Moving average models operate under the framework that the covariance between a measurement and the previous measurement is zero. While AR models use past forecast *values* to predict future values, MA models use past forecast *errors* to predict future values.

Let's look at how ACF and PACF lags look under different formulations of AR and MA models. https://nwfsc-timeseries.github.io/atsa-labs/sec-tslab-autoregressive-ar-models.html https://nwfsc-timeseries.github.io/atsa-labs/sec-tslab-moving-average-ma-models.html

Let's upload the Clear Creek discharge dataset. We will **aggregate** the data by averaging monthly values. We will then turn this into a time series, which is the format needed for ARMA modeling.

```
ClearCreekDischarge <- readNWISdv(siteNumbers = "06719505",
                      parameterCd = "00060", # discharge (ft3/s)
                      startDate = "",
                      endDate = "")
names(ClearCreekDischarge)[4:5] <- c("Discharge", "Approval.Code")

ClearCreekDischarge.Monthly <- ClearCreekDischarge %>%
  mutate(Year = year(Date),
         Month = month(Date)) %>%
  group_by(Year, Month) %>%
  summarise(Discharge = mean(Discharge))

ClearCreek_ts <- ts(ClearCreekDischarge.Monthly[[3]], frequency = 12)
```

ARMA models require stationary data. This means that there is no monotonic trend over time and there is also equal variance and covariance across the time series. The function `adf.test` will determine whether our data are stationary. The null hypothesis is that the data are not stationary, so we infer that the data are stationary if the p-value is < 0.05.

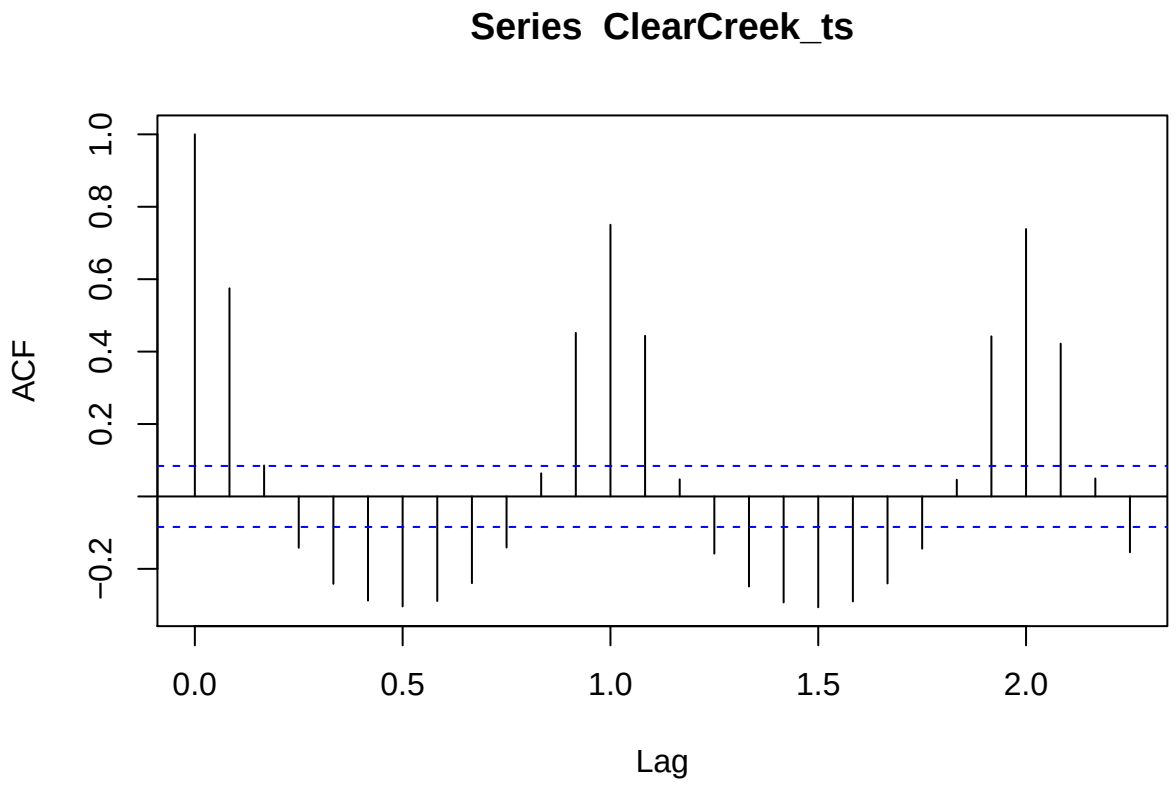
```
adf.test(ClearCreek_ts, alternative = "stationary")
```

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

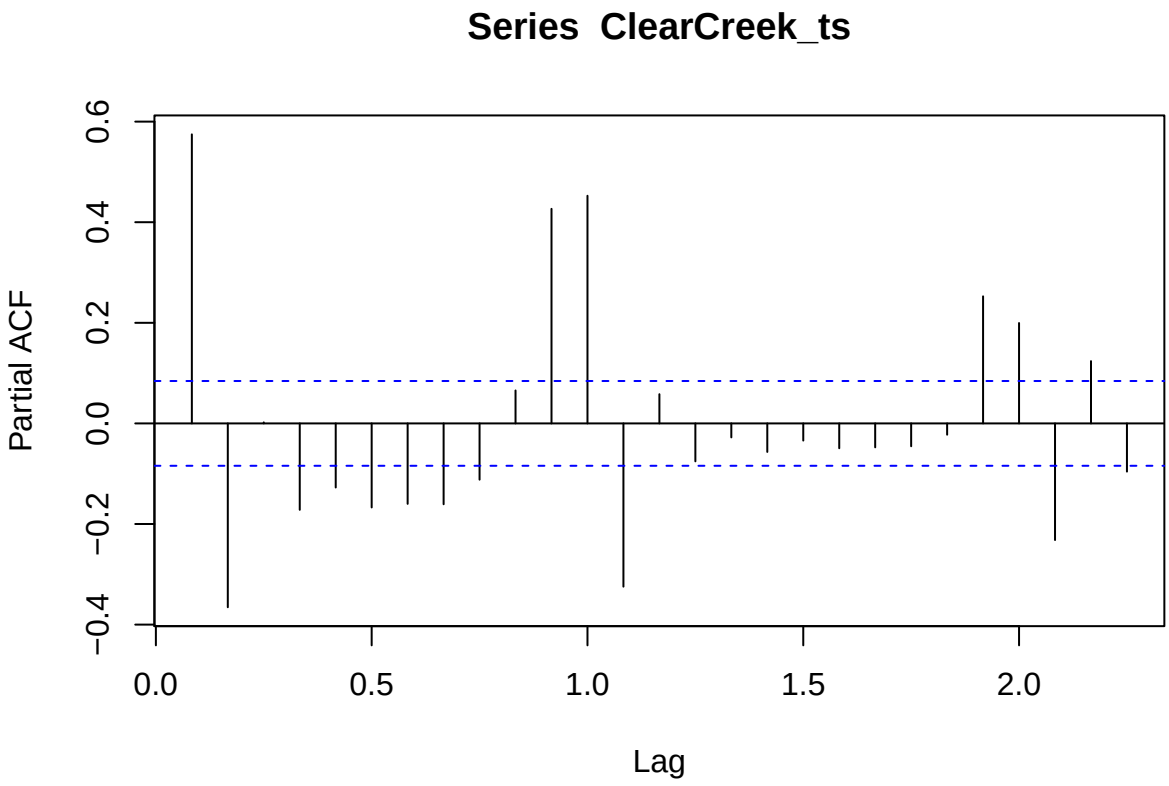
```
##
##  Augmented Dickey-Fuller Test
##
## data:  ClearCreek_ts
## Dickey-Fuller = -13.26, Lag order = 8, p-value = 0.01
## alternative hypothesis: stationary
```

Let's inspect the ACF and pacf plots. Notice these don't match up perfectly with just one AR or MA formulation, so we likely have interactions of both processes at play.

```
acf(ClearCreek_ts)
```




```
pacf(ClearCreek_ts)
```



While some processes might be easy to identify, it is often complicated to predict the order of AR and MA processes when the operate in the same dataset. To get around this issue, we will run multiple potential formulations of the model and see which one results in the most parsimonious fit using AIC. The function `auto.arima` does this automatically.

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

```
##
##  Fitting models using approximations to speed things up...
##
##  ARIMA(2,0,2)(1,1,1)[12] with drift         : Inf
##  ARIMA(0,0,0)(0,1,0)[12] with drift         : 6813.318
##  ARIMA(1,0,0)(1,1,0)[12] with drift         : 6511.189
##  ARIMA(0,0,1)(0,1,1)[12] with drift         : 6403.274
##  ARIMA(0,0,0)(0,1,0)[12]                    : 6811.303
##  ARIMA(0,0,1)(0,1,0)[12] with drift         : 6651.239
##  ARIMA(0,0,1)(1,1,1)[12] with drift         : Inf
##  ARIMA(0,0,1)(0,1,2)[12] with drift         : 6402.382
##  ARIMA(0,0,1)(1,1,2)[12] with drift         : Inf
##  ARIMA(0,0,0)(0,1,2)[12] with drift         : Inf
##  ARIMA(1,0,1)(0,1,2)[12] with drift         : 6396.46
##  ARIMA(1,0,1)(0,1,1)[12] with drift         : 6396.632
##  ARIMA(1,0,1)(1,1,2)[12] with drift         : Inf
##  ARIMA(1,0,1)(1,1,1)[12] with drift         : Inf
##  ARIMA(1,0,0)(0,1,2)[12] with drift         : 6410.387
##  ARIMA(2,0,1)(0,1,2)[12] with drift         : 6399.128
```

```
##  ARIMA(1,0,2)(0,1,2)[12] with drift         : 6398.114
##  ARIMA(0,0,2)(0,1,2)[12] with drift         : 6396.501
##  ARIMA(2,0,0)(0,1,2)[12] with drift         : 6400.75
##  ARIMA(2,0,2)(0,1,2)[12] with drift         : 6401.172
##  ARIMA(1,0,1)(0,1,2)[12]                    : 6394.433
##  ARIMA(1,0,1)(0,1,1)[12]                    : 6394.613
##  ARIMA(1,0,1)(1,1,2)[12]                    : Inf
##  ARIMA(1,0,1)(1,1,1)[12]                    : 6387.76
##  ARIMA(1,0,1)(1,1,0)[12]                    : 6490.458
##  ARIMA(1,0,1)(2,1,1)[12]                    : 6415.177
##  ARIMA(1,0,1)(0,1,0)[12]                    : 6642.766
##  ARIMA(1,0,1)(2,1,0)[12]                    : 6478.571
##  ARIMA(1,0,1)(2,1,2)[12]                    : Inf
##  ARIMA(0,0,1)(1,1,1)[12]                    : 6398.352
##  ARIMA(1,0,0)(1,1,1)[12]                    : 6404.96
##  ARIMA(2,0,1)(1,1,1)[12]                    : 6390.559
##  ARIMA(1,0,2)(1,1,1)[12]                    : 6389.532
##  ARIMA(0,0,0)(1,1,1)[12]                    : 6579.017
##  ARIMA(0,0,2)(1,1,1)[12]                    : 6388.49
##  ARIMA(2,0,0)(1,1,1)[12]                    : 6392.537
##  ARIMA(2,0,2)(1,1,1)[12]                    : 6392.605
##
##  Now re-fitting the best model(s) without approximations...
##
##  ARIMA(1,0,1)(1,1,1)[12]                    : Inf
##  ARIMA(0,0,2)(1,1,1)[12]                    : Inf
##  ARIMA(1,0,2)(1,1,1)[12]                    : Inf
##  ARIMA(2,0,1)(1,1,1)[12]                    : Inf
##  ARIMA(2,0,0)(1,1,1)[12]                    : Inf
##  ARIMA(2,0,2)(1,1,1)[12]                    : Inf
##  ARIMA(1,0,1)(0,1,2)[12]                    : Inf
##  ARIMA(1,0,1)(0,1,1)[12]                    : Inf
##  ARIMA(1,0,1)(0,1,2)[12] with drift         : Inf
##  ARIMA(0,0,2)(0,1,2)[12] with drift         : Inf
##  ARIMA(1,0,1)(0,1,1)[12] with drift         : Inf
##  ARIMA(1,0,2)(0,1,2)[12] with drift         : Inf
##  ARIMA(0,0,1)(1,1,1)[12]                    : Inf
##  ARIMA(2,0,1)(0,1,2)[12] with drift         : Inf
##  ARIMA(2,0,0)(0,1,2)[12] with drift         : Inf
##  ARIMA(2,0,2)(0,1,2)[12] with drift         : Inf
##  ARIMA(0,0,1)(0,1,2)[12] with drift         : Inf
##  ARIMA(0,0,1)(0,1,1)[12] with drift         : Inf
##  ARIMA(1,0,0)(1,1,1)[12]                    : Inf
##  ARIMA(1,0,0)(0,1,2)[12] with drift         : Inf
##  ARIMA(1,0,1)(2,1,1)[12]                    : Inf
##  ARIMA(1,0,1)(2,1,0)[12]                    : 6597.333
##
##  Best model: ARIMA(1,0,1)(2,1,0)[12]
```

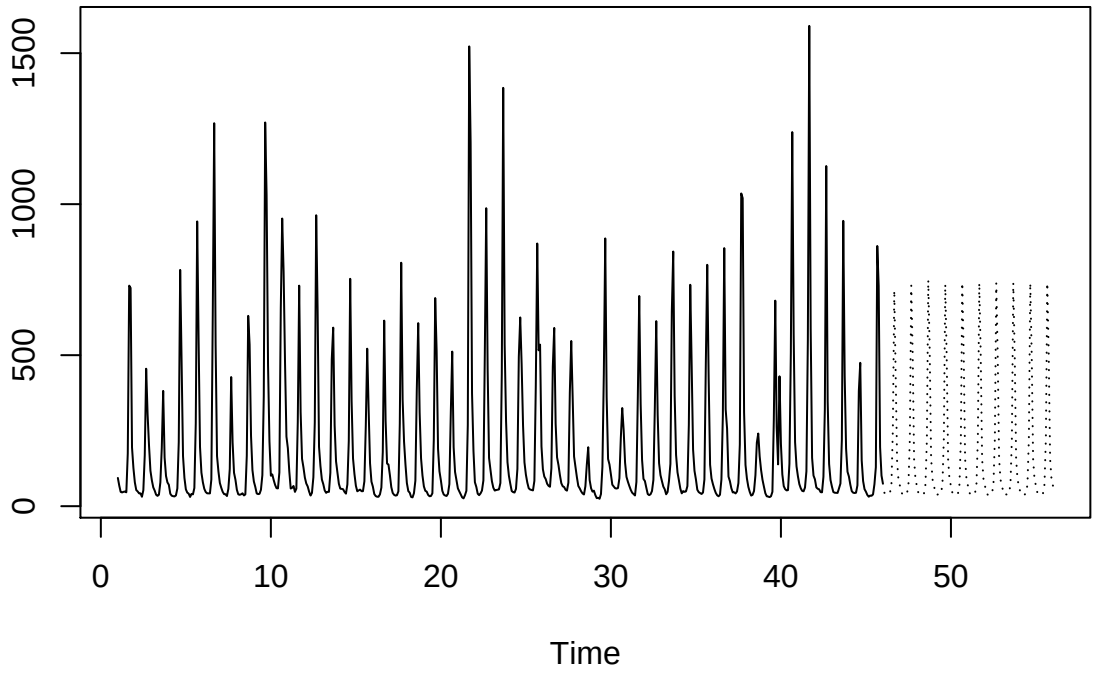
```
## Series: ClearCreek_ts
## ARIMA(1,0,1)(2,1,0)[12]
##
## Coefficients:
##          ar1     ma1     sar1     sar2
```

```
##       0.2820  0.3544  -0.6077  -0.1886
## s.e.  0.0723  0.0707   0.0435   0.0437
##
## sigma^2 estimated as 14952:  log likelihood=-3293.61
## AIC=6597.22   AICc=6597.33   BIC=6618.57
```

```r
# create an object that defines the best fit model
fit <- arima(ClearCreek_ts, c(1, 0, 1),seasonal = list(order = c(2, 1, 0), period = 12))

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

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



How do future predictions compare to the past? What other covariates might you bring into the analysis to improve forecasting capabilities?