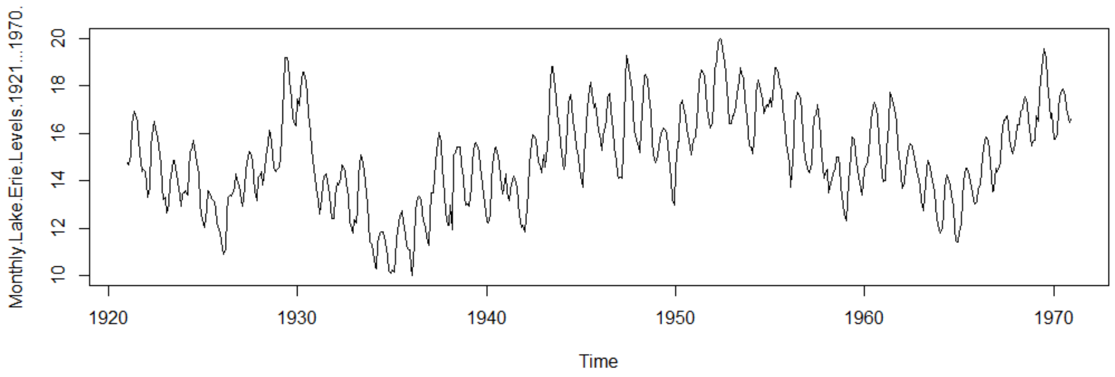
TIME SERIES ANALYSIS OF LAKE ERIE WATER LEVELS

Sean Ammirati | 5/21/2017

Introduction

The dataset I have chosen is a monthly recording of water levels in Lake Erie from the period between 1921 and 1970. The dataset is available at the following link: https://datamarket.com/data/set/22pw/monthly-lake-erie-levels-1921-1970#!ds=22pw&display=line&numberformat=n1. Below is a plot of the time series over time:



We can see that the water levels seem to jump somewhat cyclically over time. This is typical of physical processes, and thus it is an interesting dataset to analyze. We see that over time the water levels decreased initially, and then began to increase again somewhat steadily from the 1940s onwards. While there is clearly a seasonal component at work here, it is not so clear how this seasonal component operates initially from the plot above.

I first begin by fitting a seasonal means with linear trend model to summarize and get an idea of the seasonal component of the data. I then built four models – the Holt Winters model, an SARIMA model, a model decomposing the seasonal component with spectral analysis, and a model from the *prophet* package. I then compare the models using the sum squared errors and plotting them against the true data values.

I have separated the time series into two sets, as shown below. These are the training and test sets I will be using to evaluate how effective each of the models I fit are at estimating the true time series. The following code initializes the dataset as a time series, and creates a training and test dataset to perform analysis on. The test set is the last 12 observations, or one year, of the time series.

```
lkerie<-dmseries("https://datamarket.com/data/set/22pw/monthly-lake-erie-levels-1921-1970#!ds=22pw&display=line")
```

```
lkerie<-as.ts(lkerie)
lkerietrain<-as.ts(lkerie[1:(length(lkerie)-12)])
lkerietest<-lkerie[(length(lkerie)-11):length(lkerie)]
```

After this initialization, the dataset is in good condition and doesn't need to be cleaned any further. The first 6 values of the time series are shown below:

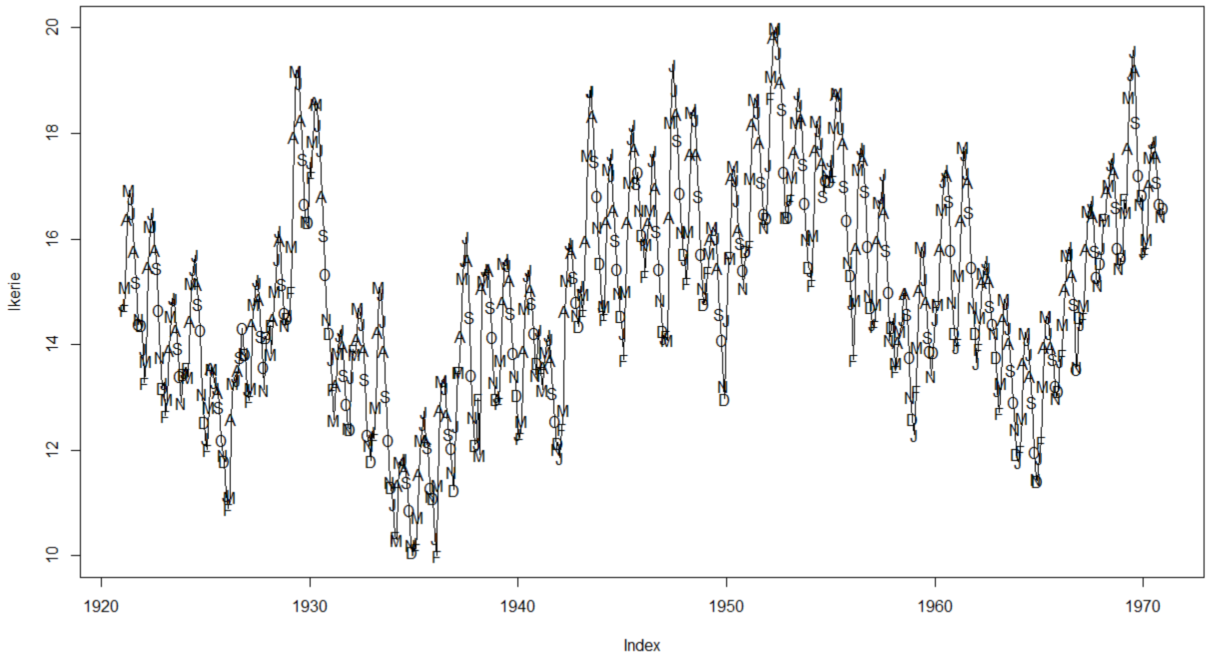
```
head(lkerie)
```

```
##         Jan    Feb    Mar    Apr    May    Jun
## 1921 14.763 14.649 15.085 16.376 16.926 16.774
```

Seasonal Means

The first model I fit was a seasonal means model, to get a sense of the seasonality of the dataset. Although this is a simplistic model that will not be used to model the data itself, it provides good insight into the seasonal nature of the time series. Below is a plot of the time series with the months added to the plot:

```
plot(lkerie)
points(y=lkerie,x=time(lkerie),pch=as.vector(season(lkerie)))
```



We can see that the time series tends to peak around the summer time, and has troughs in the colder months. The seasonal means model gives us more insight into how this is occurring:

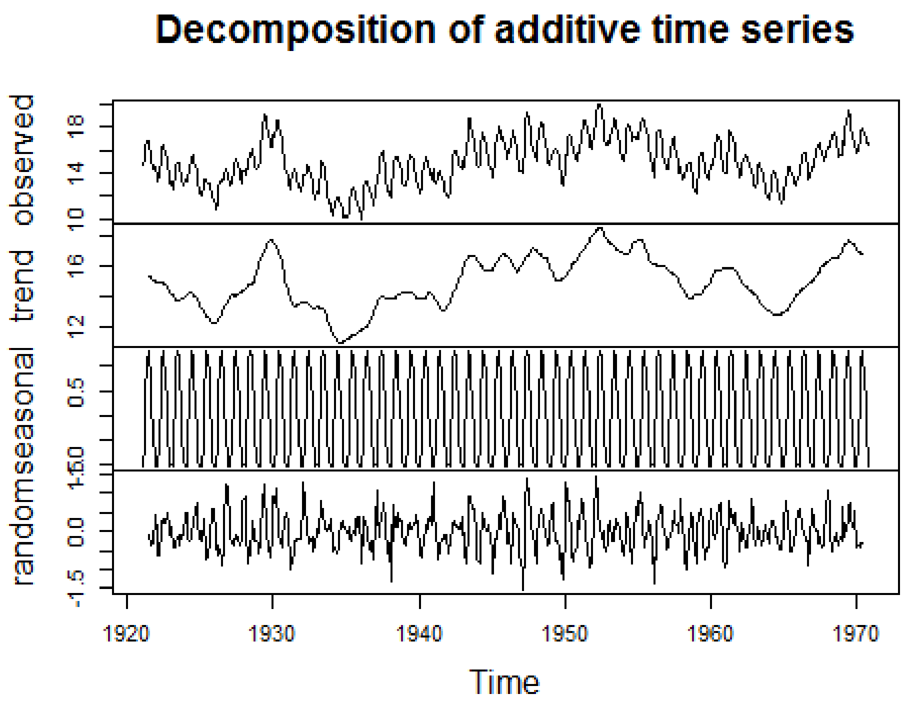
```
season.<-season(lkerie)
time.<-time(lkerie)
seasonmean<-lm(lkerie~season.+time.)

summary(seasonmean)

…
## Coefficients:
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)        -71.085340   9.350931  -7.602 1.16e-13 ***
## season.February   -0.014303   0.339687  -0.042 0.966429
## season.March       0.396175   0.339688   1.166 0.243970
## season.April       1.478212   0.339689   4.352 1.59e-05 ***
## season.May         2.101470   0.339690   6.186 1.15e-09 ***
## season.June        2.308247   0.339692   6.795 2.66e-11 ***
## season.July        2.223725   0.339695   6.546 1.29e-10 ***
## season.August      1.831822   0.339698   5.393 1.01e-07 ***
## season.September   1.257740   0.339702   3.702 0.000234 ***
## season.October     0.608217   0.339706   1.790 0.073901 .
## season.November    0.077855   0.339710   0.229 0.818809
## season.December   -0.028248   0.339715  -0.083 0.933760
## time.              0.043710   0.004805   9.097  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.698 on 587 degrees of freedom
## Multiple R-squared:  0.3018, Adjusted R-squared:  0.2875
## F-statistic: 21.14 on 12 and 587 DF,  p-value: < 2.2e-16
```

Indeed, we can see that the seasonal means model has quite a bit of significant components, suggesting that there is a good amount of seasonality in the data. However, this is not consistent, with some months having insignificant values. In general, we can see that the summer months have positive coefficients, while the colder months (which are not nearly as significant) sit somewhere close to zero. This agrees with the earlier findings from the time series plot.

Next, I perform a decomposition of the time series, to see how this package can break up the time series into seasonal, trend and random components.

```
madecomp<-decompose(lkerie)
```

```
plot(madecomp)
```

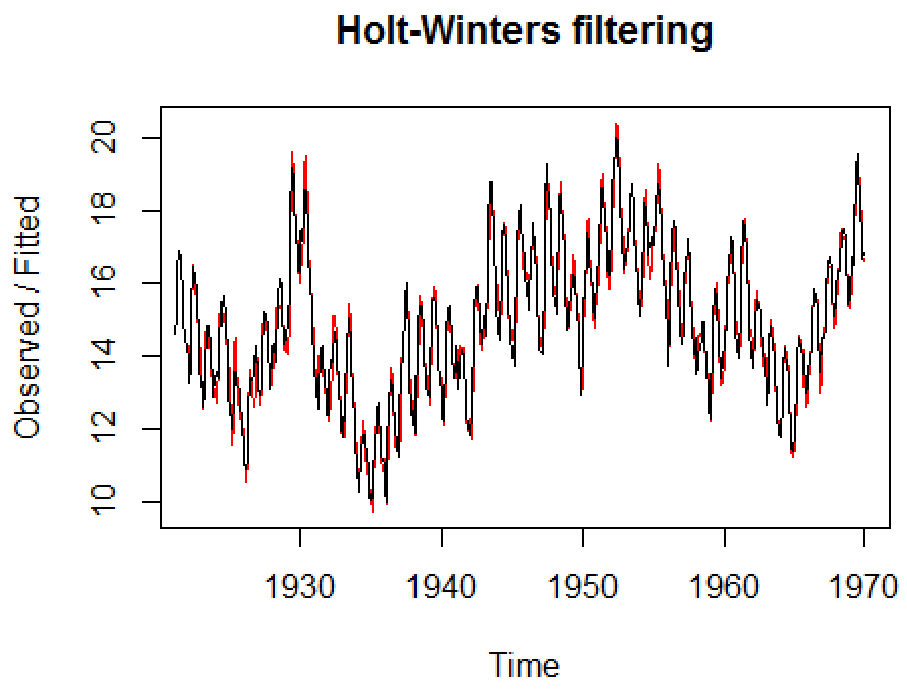


We can see again that there is clearly a seasonal component to the time series. We can also see the general trend of decreasing and increasing. When these components are removed, we see a relatively stationary random process. This cyclical nature of the seasonality lead me to consider spectral analysis. However, first we will look at two other models which can deal with seasonality – the Holt Winters' Model and a seasonal ARIMA model.

Holt-Winters' Model

I use two Holt-Winters' Models (an additive and multiplicative model) to model the data. Below, I fit the two models, and plot the resulting fits. The Holt-Winters models both consider seasonal components, although the multiplicative model and additive model come up with different quantities for the coefficients (see the appendix for the summaries). However, the smoothing parameters are relatively the same in character, with alpha=0.8973698 beta=0.003048736 and gamma: 1. This indicates that the level and seasonality are smoothing based on observations in the distant past (with gamma = 1 meaning it is considering only the previous observations), while a small beta indicates that it is using the trend of only the nearest values. I have plotted one of the models below, but the other is very similar in nature. However, as we will see later, the prediction error is quite high – (and the prediction error for the multiplicative model is, predictably, much higher).

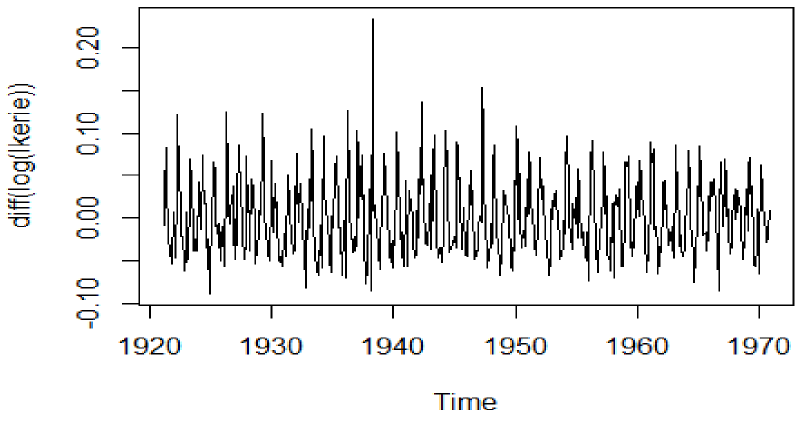
```
hw1<-HoltWinters(lkerietrain)
plot(hw1)
```



SARIMA Modeling

Next, we will turn to using an ARIMA (or in this case, a seasonal ARIMA model) to model and predict the data. The above time series is not stationary, so we must transform it. The transformation which creates an optimal situation for ARIMA modeling is one that is roughly stationary. After looking through various transformations (including logarithmic, differencing and BoxCox transformations), I settled upon a differencing of logs. By differencing the logarithms of the model, we get a result that is very close to stationary. The resulting plot is plotted below.

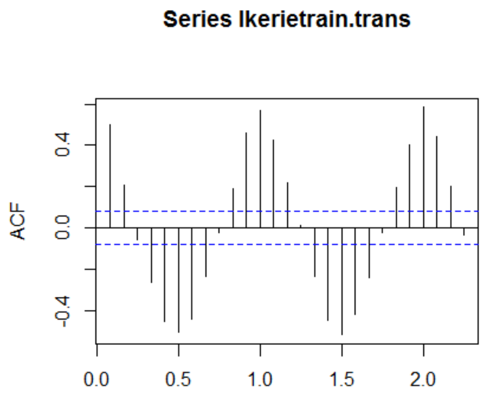
```
lkerietrain.trans<-diff(log(lkerie))
plot(lkerietrain.trans)
```

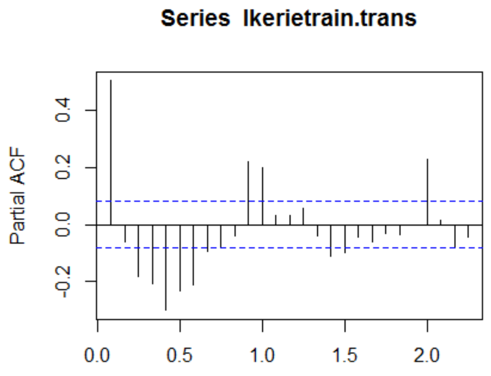


Except for the one outlier just before 1940, we can see that this time series is now stationary, and we can proceed to fit ARIMA models to it.

Using the ACF, PACF and EACF plots, we can see that it will be difficult to find a fitting ARMA (without a seasonal component) model that fits the data well. Since we believe there is a good degree of seasonality to this data, this is not surprising. In the plots below of the ACF and PACF, we see some sine and cosine patterns in the autocorrelations. This indicates a seasonal model may be more appropriate than an ordinary ARMA or ARIMA model.

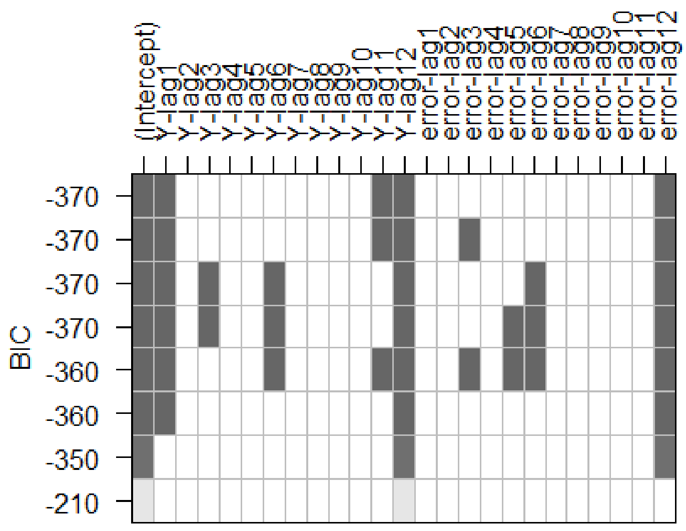
```
acf(lkerietrain.trans); pacf(lkerietrain.trans);
```





Using the arima.subsets function gives us a good idea of the seasonality and we can better infer the best models to use based on this.

```
a<-armasubsets(lkerietrain.trans,nar=12,nma=12)
plot(a)
```



We can see the most significant lags are those at times 1, 11, and 12 for the AR component, and lag 12 for the MA component. Using the auto.arima and get.best.arima functions, we can get the optimal model based on the AIC.

```
auto.arima(lkerietrain.trans)
## Series: lkerietrain.trans
## ARIMA(1,0,0)(2,0,0)[12] with zero mean
##
## Coefficients:
##          ar1    sar1    sar2
##       0.2170  0.3016  0.3845
## s.e.  0.0428  0.0381  0.0383
##
## sigma^2 estimated as 0.001024:  log likelihood=1209.86
## AIC=-2411.72   AICc=-2411.65   BIC=-2394.14

get.best.arima(lkerietrain.trans,maxord=c(2,2,2,2,2,2))
## [[1]]
## [1] -2502.121
## [[2]]
##
## Call:
## arima(x = x.ts, order = c(p, d, q), seasonal = list(order = c(P, D, Q), fre
quency(x.ts)),
##     method = "CSS")
##
## Coefficients:
##          ma1    sar1    sar2     sma1     sma2  intercept
##       0.1438  0.2975  0.6861  -0.2604  -0.6219     0.0108
## s.e.  0.0374  0.1219  0.1215   0.1365   0.1321     0.0130
##
```

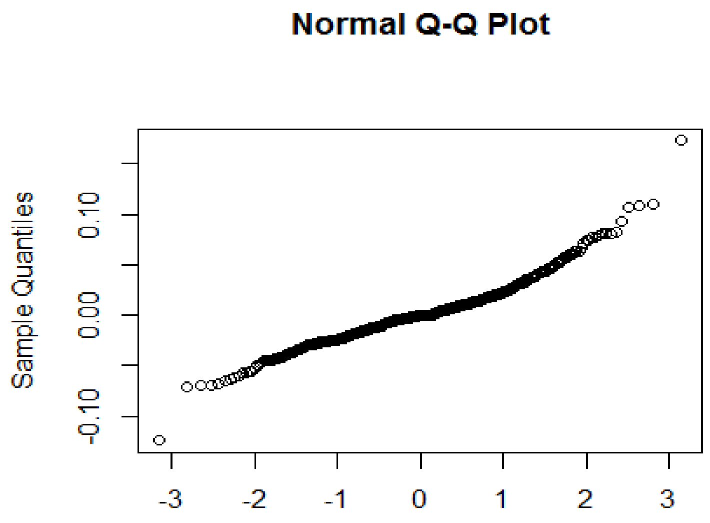
```
## [[3]]
## [1] 0 0 1 2 0 2
```

The get.best.arima's model produces the minimal AIC, and therefore is the optimal model. We can see it is very similar to the models determined by the subsets, and adds an MA component to the seasonal part of the ARIMA model. I consider both models found (that is, SARIMA[1,0,0]x[2,0,2] and SARIMA[0,0,1]x[2,0,2]). In both models, all of the terms are significant, so there is no need to remove any.

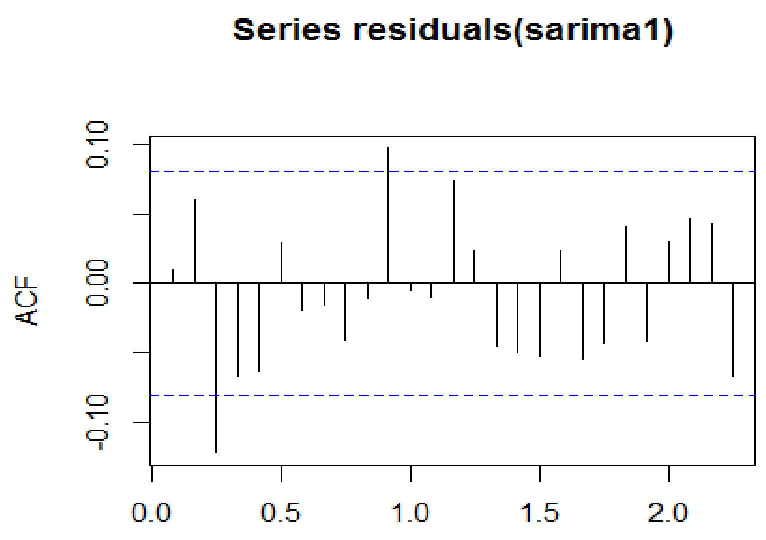
Looking at the residuals of these ARIMA models, we can see that they are roughly normally distributed, with the outliers skewing it somewhat. This is encouraging. Unfortunately, the residuals (and squared residuals) do seem to still exhibit autocorrelation. However, the ACF and PACF squared do not appear to have drastically more correlated values in the squared case. I have included the Q-Q plot and the ACF of the residuals below. For brevity, I have omitted the PACF, but it exhibits similar results.

Regardless, this result is disappointing. This led me to use spectral analysis to interpret the seasonality, as I believed there was some underlying seasonal pattern that could not be found using these standard SARIMA processes.

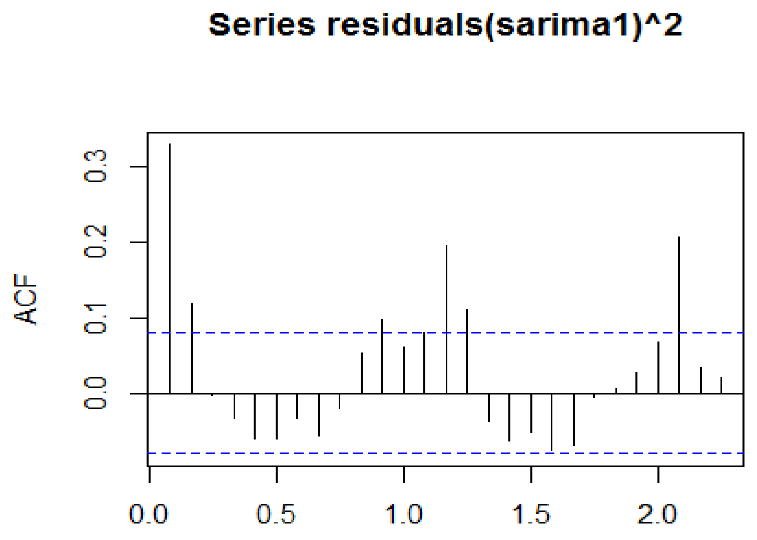
```r
qqnorm(y=residuals(sarima1))
```



```r
acf(residuals(sarima1))
```



```
acf(residuals(sarima1)^2)
```



Spectral Analysis + ARIMA

The cyclical component of the time series led me to believe that using spectral analysis to analyze the seasonality of the data may be fruitful. The method I used here was to find the highest frequencies on the periodogram, and use these to model the seasonal components.

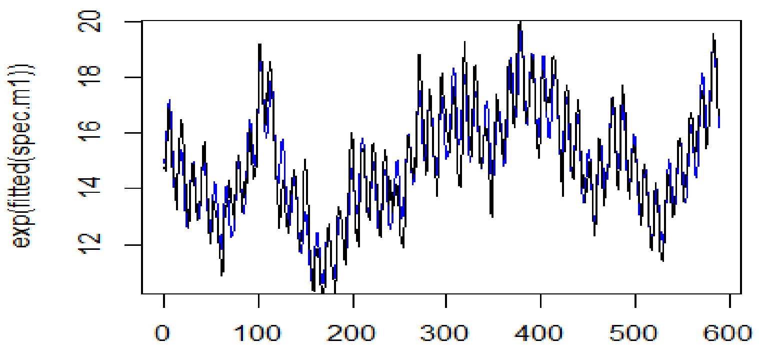
Using the ten frequencies with this highest spectrum on the log of the time series, I created twenty variables with a sin/cosine function pertaining to each. I then performed linear regression on the transformed time series with these ten values to create a harmonic model.

```
spec.per<-spec(log(lkerietrain))
frequencies<-spec.per$freq[order(spec.per$spec,decreasing=TRUE)]
cc1<-as.numeric(cos(2*pi*frequencies[1]*t))
d1<-sin(2*pi*frequencies[1]*t)
c2<-cos(2*pi*frequencies[2]*t)
d2<-sin(2*pi*frequencies[2]*t)
c3<-cos(2*pi*frequencies[3]*t)
d3<-sin(2*pi*frequencies[3]*t)
c4<-cos(2*pi*frequencies[4]*t)
d4<-sin(2*pi*frequencies[4]*t)
…
c9<-cos(2*pi*frequencies[9]*t)
d9<-sin(2*pi*frequencies[9]*t)
c10<-cos(2*pi*frequencies[10]*t)
d10<-sin(2*pi*frequencies[10]*t)
…
summary(spec.m1)

##
## Call:
## lm(formula = log(lkerietrain) ~ cc1 + c2 + c3 + c4 + c5 + c6 +
##     c7 + c8 + c9 + c10 + d1 + d2 + d3 + d4 + d5 + d6 + d7 + d8 +
##     d9 + d10 + t)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.176056 -0.029902 -0.001996  0.033973  0.149171
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.665e+00  1.126e-02 236.628  < 2e-16 ***
## cc1         -2.828e-02  2.891e-03  -9.781  < 2e-16 ***
## c2          -8.263e-02  2.806e-03 -29.448  < 2e-16 ***
## c3           5.336e-02  2.890e-03  18.468  < 2e-16 ***
## c4           1.197e-02  2.884e-03   4.151 3.82e-05 ***
## c5           6.030e-02  2.875e-03  20.975  < 2e-16 ***
## c6           4.666e-02  2.880e-03  16.204  < 2e-16 ***
## c7          -2.565e-02  2.869e-03  -8.939  < 2e-16 ***
## c8           2.151e-02  2.845e-03   7.559 1.65e-13 ***
## c9          -2.011e-02  2.864e-03  -7.023 6.27e-12 ***
## c10          4.606e-03  2.813e-03   1.638  0.10207
## d1          -5.751e-02  7.686e-03  -7.483 2.80e-13 ***
## d2          -2.139e-02  2.808e-03  -7.618 1.09e-13 ***
## d3           6.664e-02  4.508e-03  14.783  < 2e-16 ***
## d4          -5.412e-02  3.265e-03 -16.577  < 2e-16 ***
## d5          -6.328e-03  2.986e-03  -2.119  0.03450 *
## d6          -2.360e-02  3.085e-03  -7.648 8.83e-14 ***
## d7           4.081e-03  2.927e-03   1.394  0.16376
## d8          -2.407e-02  2.841e-03  -8.470  < 2e-16 ***
## d9          -7.171e-03  2.890e-03  -2.482  0.01337 *
## d10          1.811e-02  2.826e-03   6.408 3.11e-10 ***
## t            1.123e-04  3.765e-05   2.983  0.00297 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 0.04811 on 566 degrees of freedom
## Multiple R-squared:  0.8804, Adjusted R-squared:  0.8759
## F-statistic: 198.4 on 21 and 566 DF,  p-value: < 2.2e-16
```

The results here are very significant, with a very high R-squared (.8804). We can see that the plot below produced a moderately good fit, considering it is just a sum of sine and cosine waves. However, the danger of overfitting is present. To mitigate this issue, I used the training data and a ARIMA (1,0,1) to find the sum squared errors related to different amounts of frequencies. We can see that the error plateaus at 2 frequencies. I used frequencies of 2 and 4 to create these fits. I then used these frequencies as a prediction of the seasonal components.

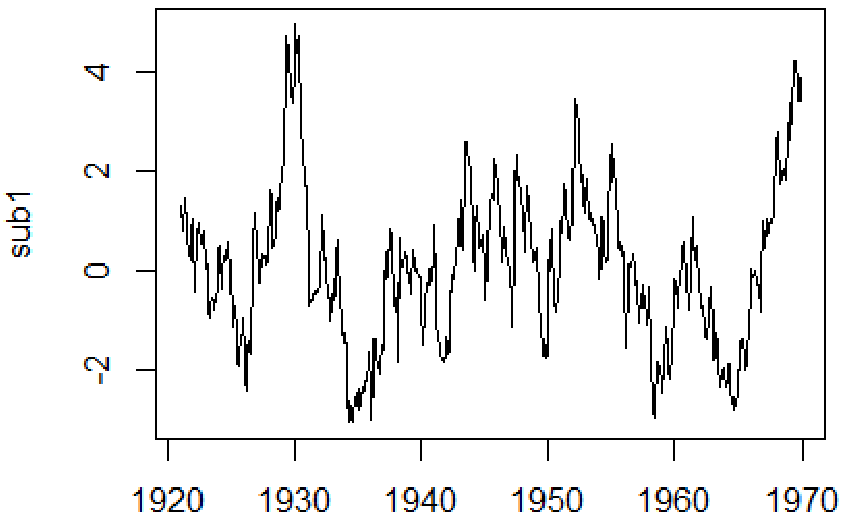
```
plot(exp(fitted(spec.m1)),type="l",col="blue")
lines(t,lkerietrain)
```



```
…print(sum(((arimasub$fitted+pre)-lkerietrain)^2))
}

## [1] 1
## [1] 177.6831
## [1] 2
## [1] 108.6809
## [1] 3
## [1] 107.2411
## [1] 4
## [1] 106.4832
## [1] 5
## [1] 103.6477
## [1] 6
## [1] 100.4989
## [1] 7
## [1] 99.84188
## [1] 8
## [1] 96.36774
## [1] 9
## [1] 94.99602
## [1] 10
## [1] 93.93375
```

Using the models with 2 and 4 frequencies, I subtracted these fitted spectral models from the original time series. As the plots are very similar in nature, I will only produce one of them here:

```
sub1<-lkerietrain-exp(fitted(models[[2]]))
sub2<-lkerietrain-exp(fitted(models[[4]]))
plot(sub1)
```



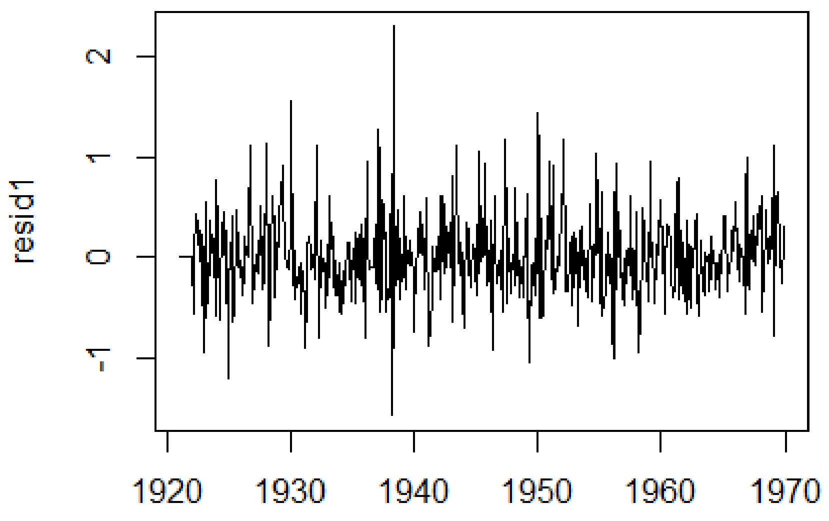
This produced two time series that are very reminiscent of random noise, which is what is expected, as this was to remove the seasonal component from the data. After a similar analysis to this data to the above SARIMA models, I arrived at two models to fit this 'white noise' like data, SARIMA[1,0,1]x[0,1,1] for the first and SARIMA[2,1,1]x[1,0,1] for the second (the code is included in the appendix, but this result had the smallest AIC from the get.best.arima() function.)

```
a3spec1<-Arima(y = sub1, order = c(1, 0, 1), seasonal = c(0, 1, 1),
               method = "CSS")
```

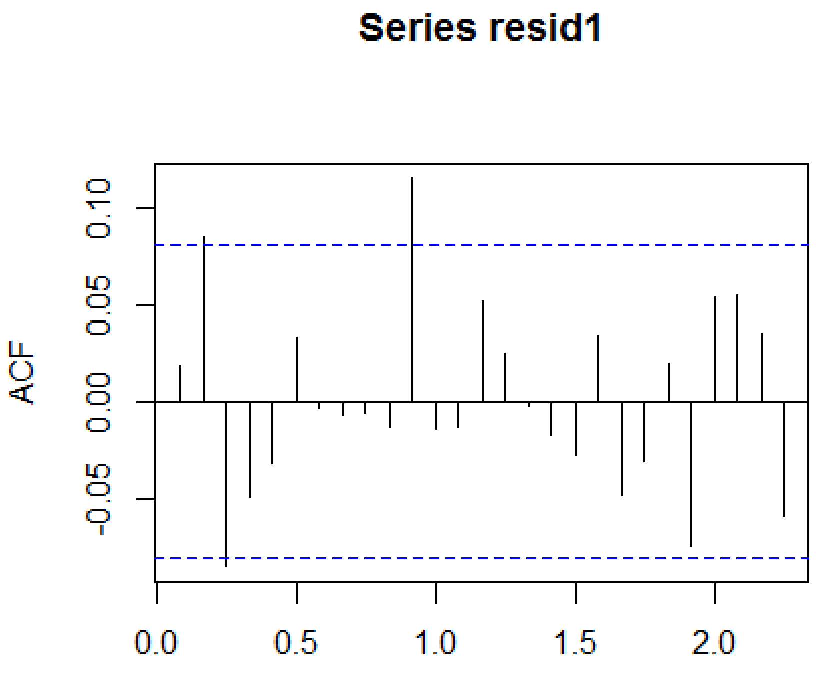
```
a3spec2<-Arima(y = sub2, order = c(2, 1, 1), seasonal = c(1, 0, 1),
               method = "CSS")
```

The plots below show that the residuals for the first model are relatively normal and that they do not exhibit very high auto-correlation or partial auto-correlation in the squared cases, which suggests that a these ARIMA models are sufficient, and a GARCH model is unneccessary.

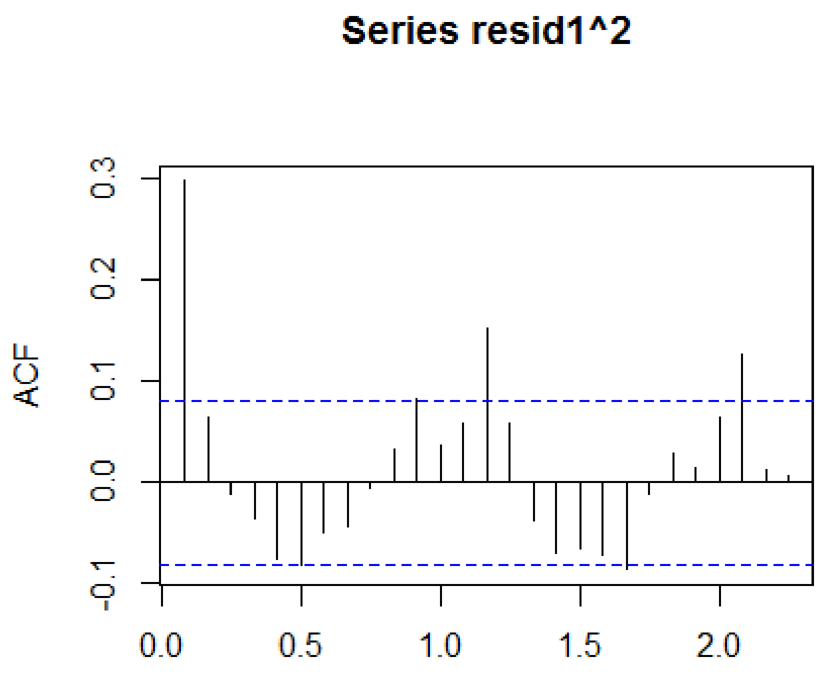
```
plot(resid1)
```



```
acf(resid1)
```



```
acf(resid1^2)
```



Prophet

The final model that I considered was one produced by Facebook's prophet package for R. All that is needed is to initialize the data and set it up appropriately. Although the prophet package is somewhat of a "black box" – and it is most effective on daily data – I thought it would be interesting to include it and see how it performs as compared with the other models.

```
ds<-as.vector(as.Date(time(as.ts((lkerietrain)))))
y<-as.numeric((log(lkerietrain)))
df<-data.frame(ds,y)
prop<-prophet(df,weekly.seasonality=TRUE)
future <- make_future_dataframe(prop, period = 12,freq="m")
forecast <- prophet:::predict.prophet(prop, future)
```

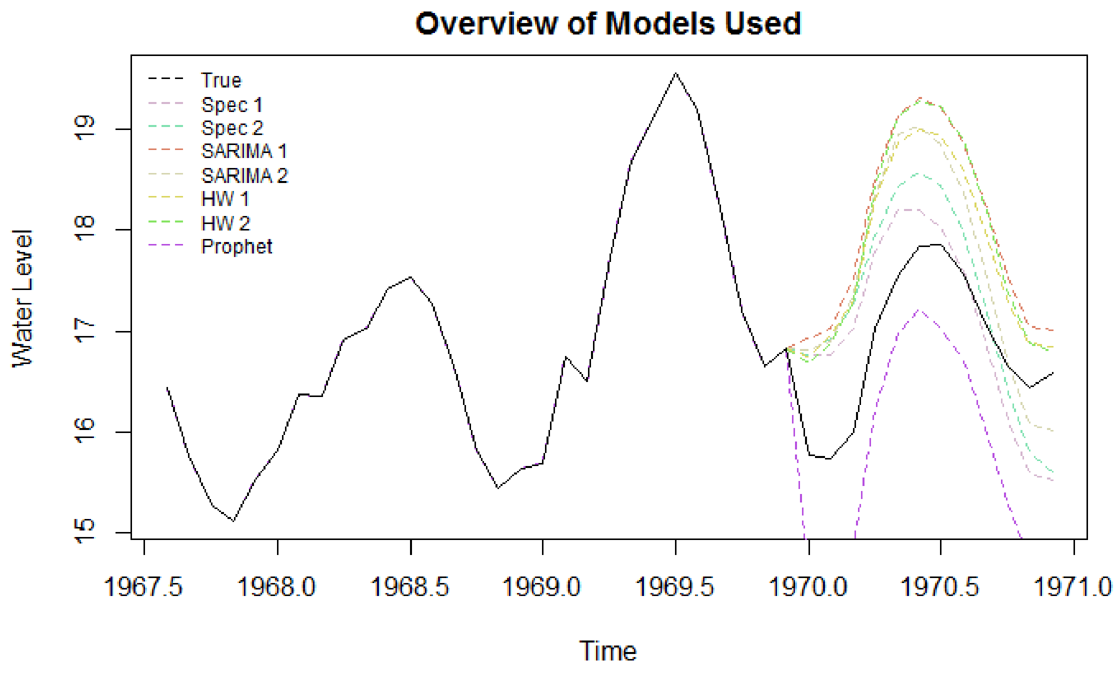
Results

Below is a table representing the sum squared errors of each of the models when forecasting for the next 12 months. These are the errors of each model in comparison to the test data.

```
results<-
data.frame(c(ssehw1,ssehw2,ssesarima1,ssesarima2,ssesp1,ssesp2,sseproph))
results
```

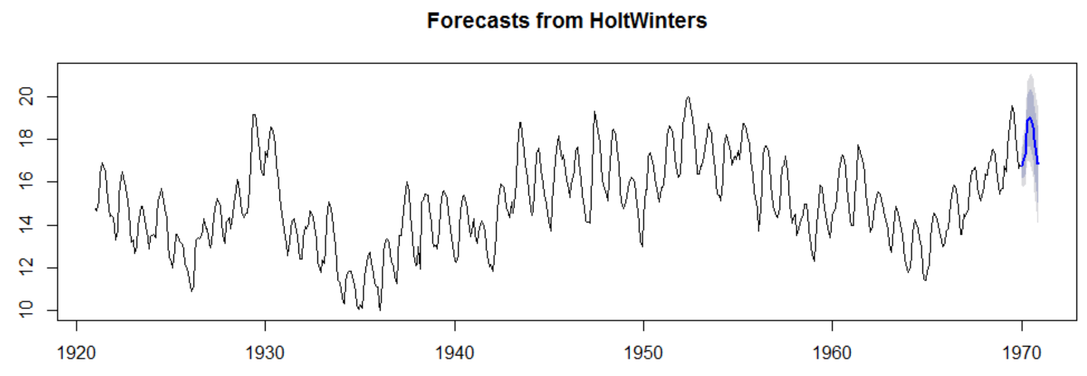
```
results

##                 SSE
## HW 1      12.666109
## HW 2      16.072445
## SARIMA 1 18.664534
## SARIMA 2 11.787142
## Spec 1     6.490393
## Spec 2     8.206515
## Prophet   16.377620
```

We can see that the models which use the frequencies from spectral analysis to remove the seasonal components are the best at predicting the data. This makes sense, as many geophysical processes can be modeled well using the sin and cosine waves in spectral analysis. Besides these two models, the SARIMA 2 model predicted with the least error. Below, I produce a plot which shows all the models' predictions as well as the true values for the time series.

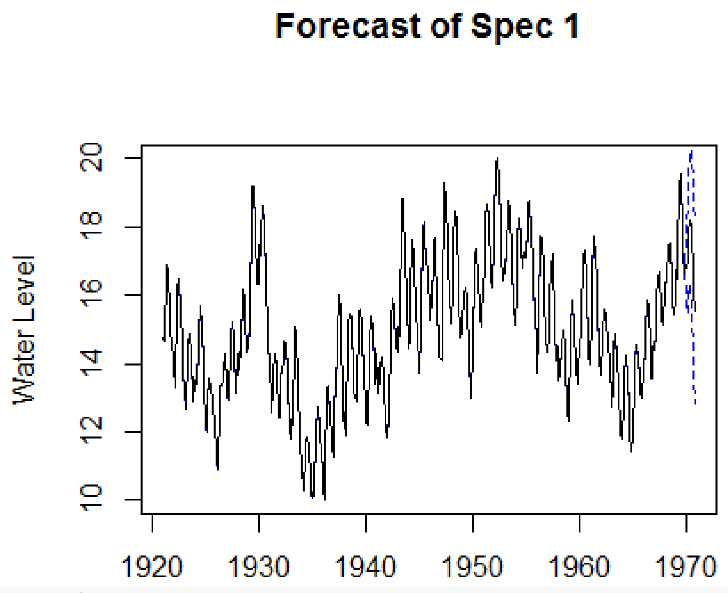


Finally, I produce graphs with confidence intervals for the most effective models of each type.

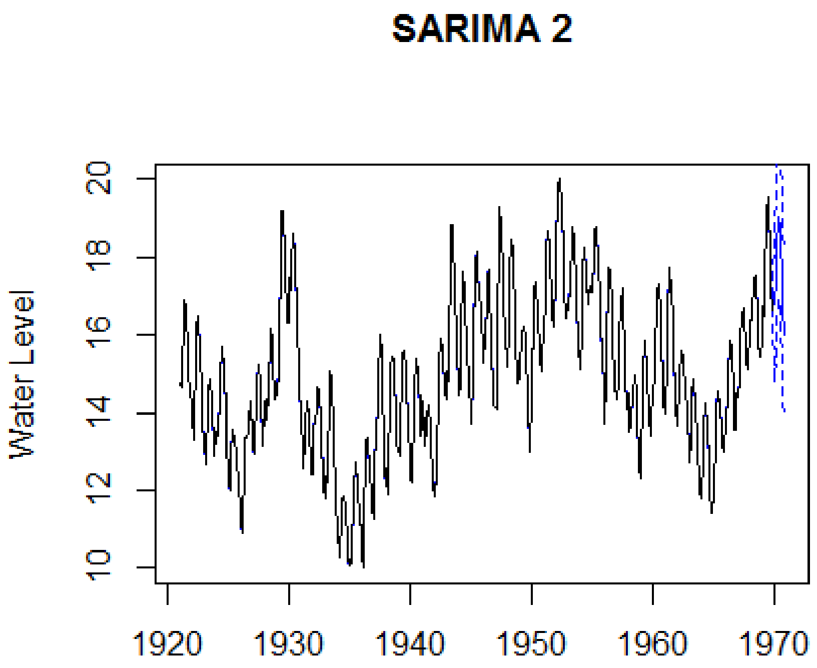
```
plot(forecast(hw1,12))
```



```
plot(x=time(lkerie),c(lkerietrain,finalpred1),ty="l",main="Forecast of Spec 1"
,xlab="Time",ylab="Water Level")
lines(x=c(time(lkerie)),c(lkerietrain,finalpred1+1.96*arimase1),lty="dashed",c
ol="blue")
lines(x=c(time(lkerie)),c(lkerietrain,finalpred1-1.96*arimase1),lty="dashed",c
ol="blue")
lines(lkerietrain,col="black")
```



```
ts.plot(lkerietrain,predarima2,col="blue",main="SARIMA 2",xlab="Time",ylab="Wa
ter Level")
lines(x=c(time(lkerie)),c(lkerietrain,(predarima2+1.96*searima2)),col="blue",l
ty="dashed")
lines(x=c(time(lkerie)),c(lkerietrain,(predarima2-1.96*searima2)),col="blue",l
ty="dashed")
lines(lkerietrain)
```



Appendix 1: Holt Winter's Model Summary

```
hw1<-HoltWinters(lkerietrain)
hw1

## Holt-Winters exponential smoothing with trend and additive seasonal compone
nt.
##
## Call:
## HoltWinters(x = lkerietrain)
##
## Smoothing parameters:
##  alpha: 0.8973698
##  beta : 0.003048736
##  gamma: 1
##
## Coefficients:
##             [,1]
## a   17.681805977
## b    0.001184109
## s1  -0.939515621
## s2  -0.725719220
## s3  -0.382678751
## s4   0.615978236
## s5   1.191656593
## s6   1.313130483
## s7   1.254551291
## s8   0.906224262
## s9   0.307195065
## s10 -0.410974316
## s11 -0.818406209
## s12 -0.850805977

hw2<-HoltWinters(lkerietrain,beta=NULL,seasonal="multiplicative")
hw2

## Holt-Winters exponential smoothing with trend and multiplicative seasonal c
omponent.
##
## Call:
## HoltWinters(x = lkerietrain, beta = NULL, seasonal = "multiplicative")
##
## Smoothing parameters:
##  alpha: 0.8949784
##  beta : 0.00251526
##  gamma: 1
##
## Coefficients:
##             [,1]
## a   17.825739589
## b   -0.002136618
## s1   0.936075995
## s2   0.946946720
## s3   0.969976647
## s4   1.034920056
## s5   1.072923841
## s6   1.082042562
## s7   1.080047070
## s8   1.060526755
## s9   1.023815328
## s10  0.976372750
## s11  0.948679604
## s12  0.944196448
```

Appendix 2: Code to Create Optimal Spectral Models

```
models<-list()

for(i in 1:10){
  beg<-"cc1+d1"
  if(i>1){
    for(s in 2:i){
      beg<-paste(beg,paste("c",s,sep=""),paste("d",s,sep=""),sep="+")
    }
  }
  end<-paste(beg,"t",sep="+")
  models[[i]]<-lm(formula(paste("log(lkerietrain)","~",end,sep="")))
  pre<-exp(fitted(models[[i]]))
  sub<-lkerietrain-exp(fitted(models[[i]]))
  arimasub<-Arima(sub, order=c(1,0,1))
  print(i)
  print(sum(((arimasub$fitted+pre)-lkerietrain)^2))
}

## [1] 1
## [1] 177.6831
## [1] 2
## [1] 108.6809
## [1] 3
## [1] 107.2411
## [1] 4
## [1] 106.4832
## [1] 5
## [1] 103.6477
## [1] 6
## [1] 100.4989
## [1] 7
## [1] 99.84188
## [1] 8
## [1] 96.36774
## [1] 9
## [1] 94.99602
## [1] 10
## [1] 93.93375
```