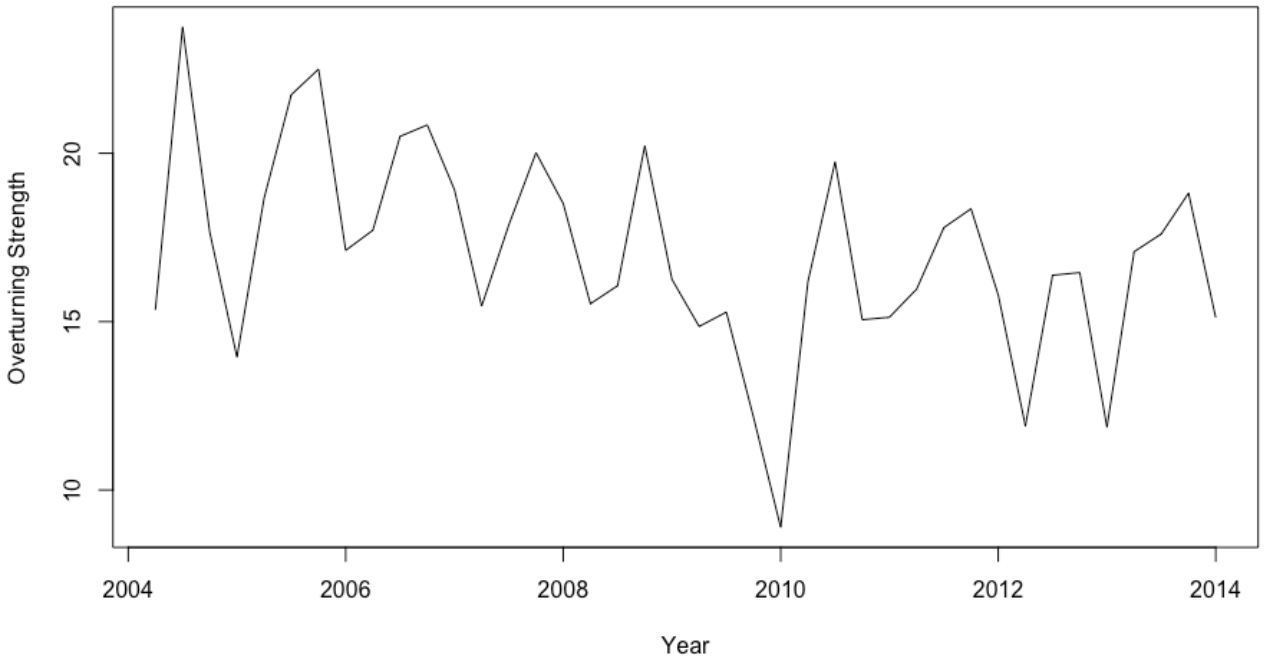
Daanish Ahsan
Statistical Modelling in Space and Time
Candidate Number: 124070

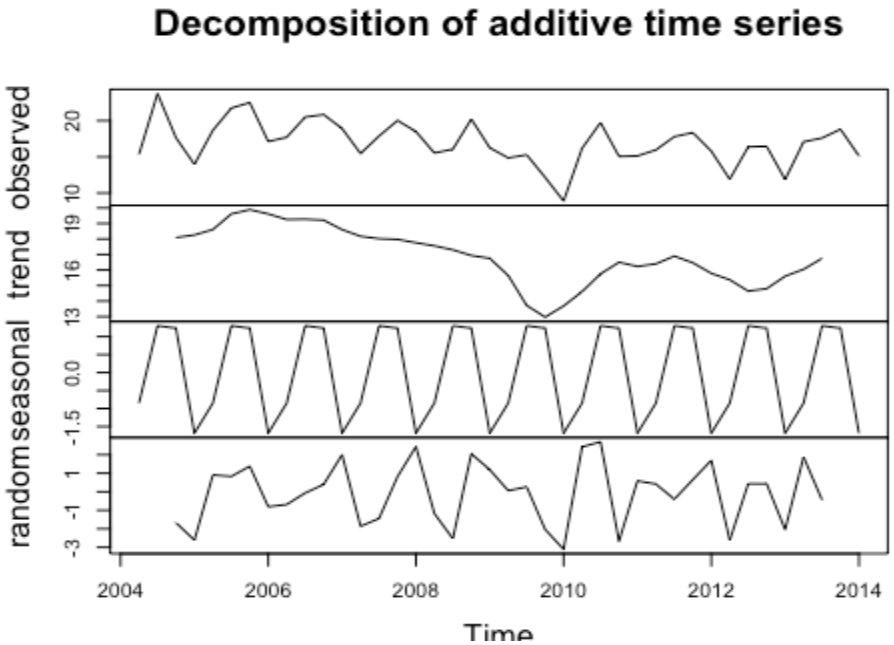
Statistical Modelling in Time and Space Assignment 2

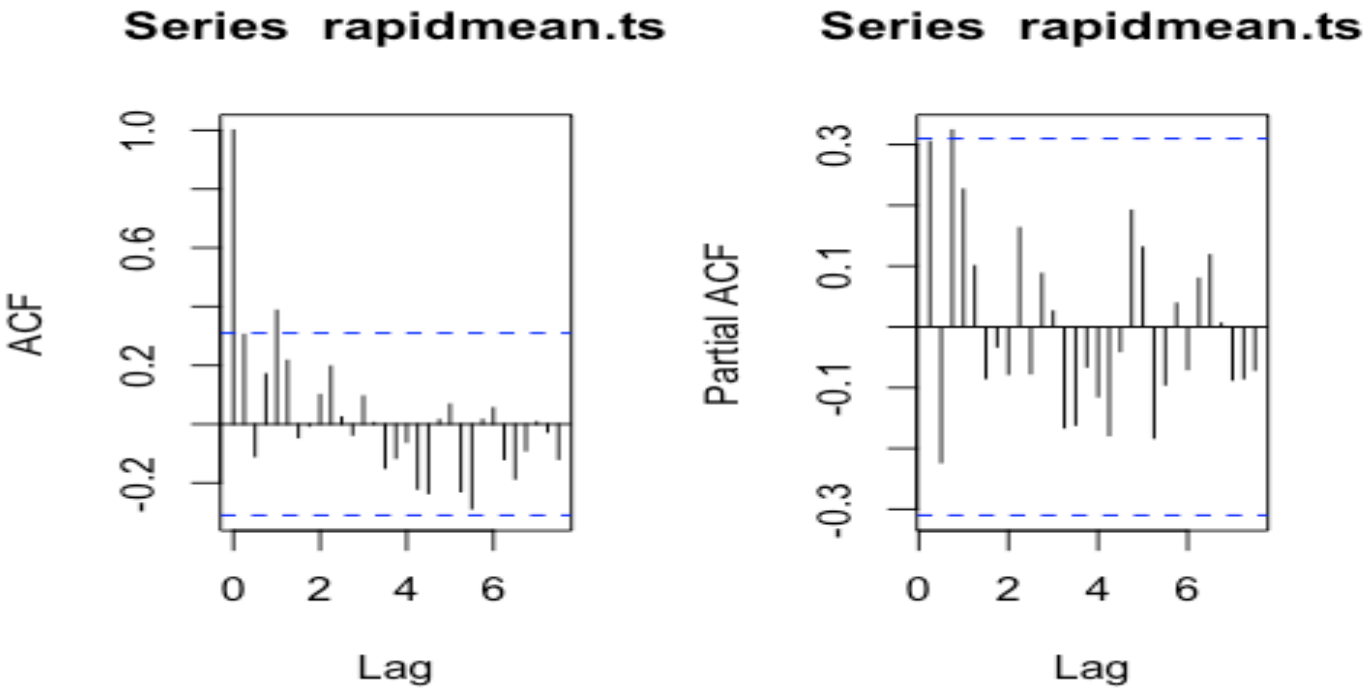
I)



In order to fit the different models to the data and predict the next 6 quarters, the quarterly means of the data were taken, and this was then converted to time series data. Shown above is the plot of that adjusted data, with overturning strength against year. An ARMA model and ARIMA model both combine a general autoregressive model AR and general moving average model MA. However, they differ in that ARMA models are fitted to stationary data, whilst an ARIMA model is used primarily when the data is non-stationary and there is a trend component. ARIMA models convert the data to stationary by using a differencing component. Thus, a good first step is to apply the augmented Dickey-Fuller test to check for stationarity of the data. The null hypothesis is that the data has unit root and is non stationary, with the alternative hypothesis being that the data is stationary. The p-value of 0.406 is greater than 0.05, suggesting we cannot reject H0, and so the data must be non stationary, suggesting an ARIMA model is most appropriate. However, we should also check if seasonality needs to be taken into account in the ARIMA model. The first diagram on page 2 decomposes the data into seasonal, trend and irregular components by using moving averages. From the diagram, the range of seasonality is 3, whilst the trend range is 6, suggesting that seasonality should be considered.

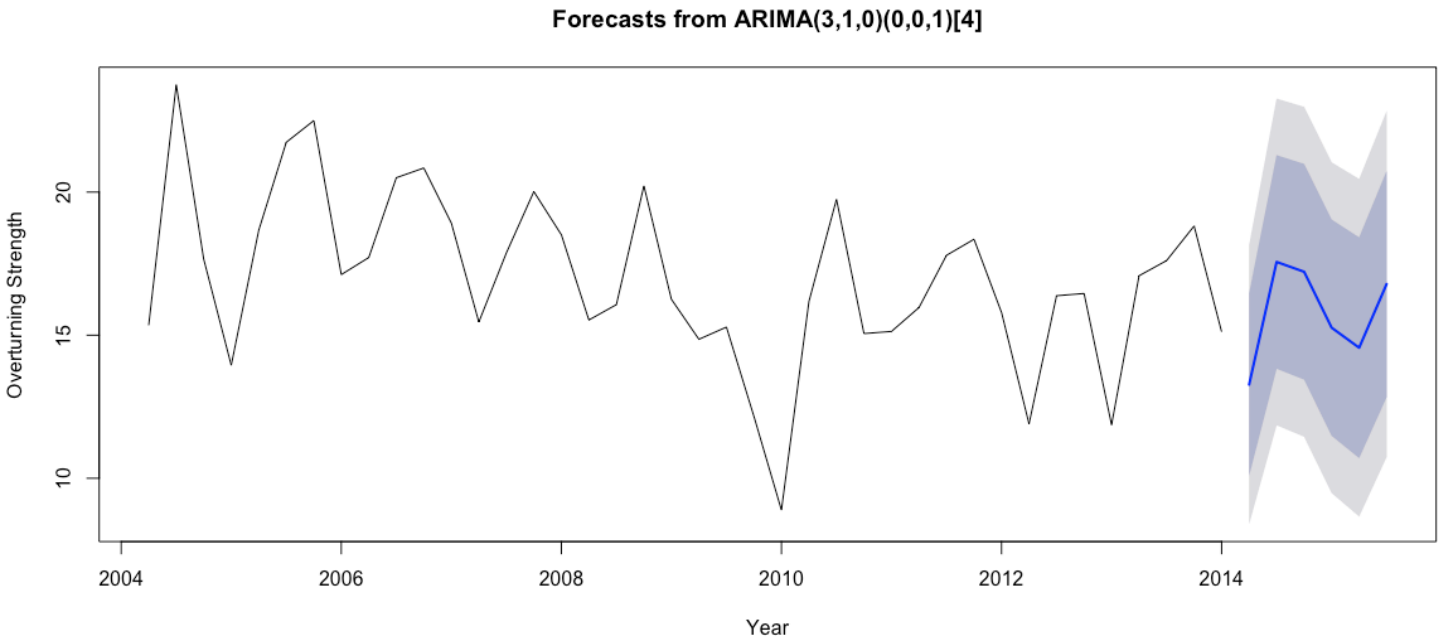
Daanish Ahsan
Statistical Modelling in Space and Time
Candidate Number: 124070





The ACF and PACF shown above are helpful in determining the MA order and AR order respectively, but are not easy to interpret. Thus, using the auto.arima function is very helpful in automatically selecting the best model, by running models with varying p, d, q values, and comparing them using AIC to prevent overfitting.

Daanish Ahsan
Statistical Modelling in Space and Time
Candidate Number: 124070

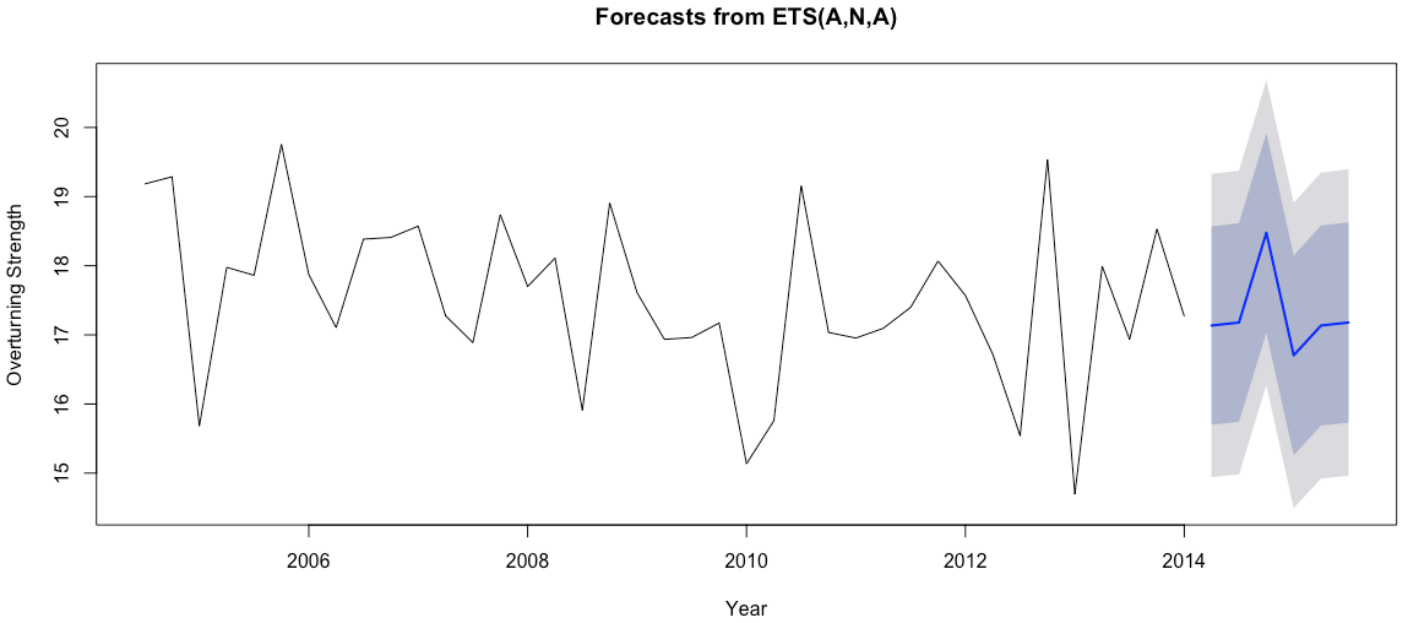


Above is the plotted forecast of the next 6 quarters, using the optimal ARIMA model given by auto.arima. The forecast seems reasonably accurate, but it is still important to take into account some model diagnostics. The AIC of this model is 190.6, and some other valuable forecasting diagnostics are seen below, the most important of which are the RMSE and MAPE. The RMSE is the square root of the average squared error, and so measures prediction errors. Its value of 2.3, when compared to the range of overturning strength of about 15, seems fairly reasonable. The MAPE is the mean absolute percentage error; here the value of 11.5% again seems fairly low and reasonable.

```
                     ME      RMSE       MAE        MPE     MAPE      MASE
Training set -0.2826744 2.323982 1.751887 -3.785926 11.50101 0.6955644
```

We can now try fitting an ARMA model for comparison, although as shown by the ADF test, the data is non stationary and so an ARIMA is more suitable. The order values of this model were adjusted multiple times until the lowest AIC was achieved. This model was then used to forecast the next 6 quarters, as with the ARIMA model. The model AIC of 185.7 is lower than the ARIMA, but perhaps due to the lesser order values ARMA models use. Virtually all forecasting accuracy measures for the ARMA shown below are lower than for the ARIMA, and whilst the forecast on the next page looks reasonable, it should again be noted that the ARMA model generally isn't used for non stationary data, whilst the ARIMA converts this to stationary.

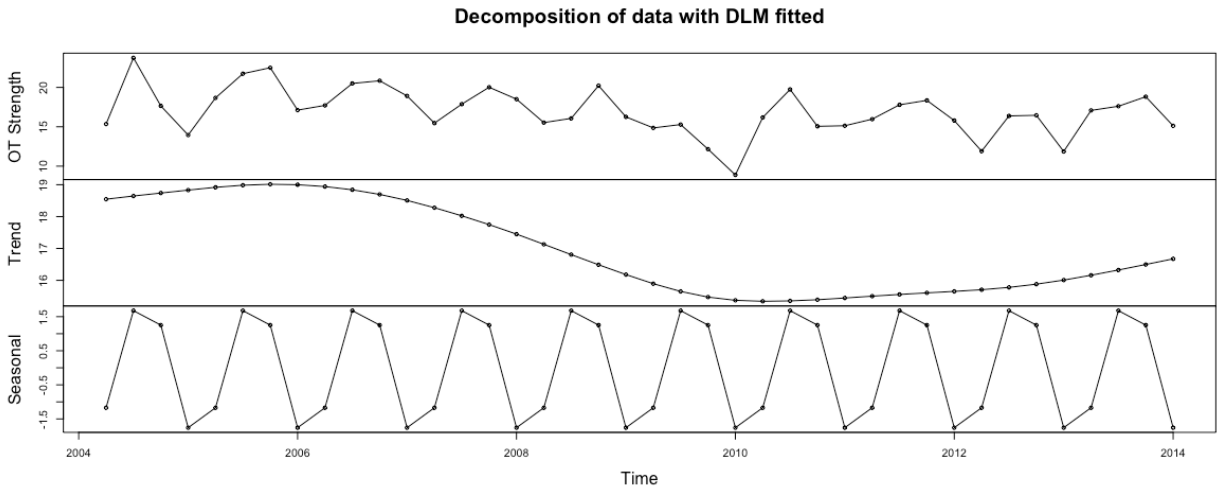
```
                     ME      RMSE       MAE        MPE     MAPE      MASE
Training set -0.1901494 1.028557 0.831914 -1.419547 4.885405 0.6985957
```



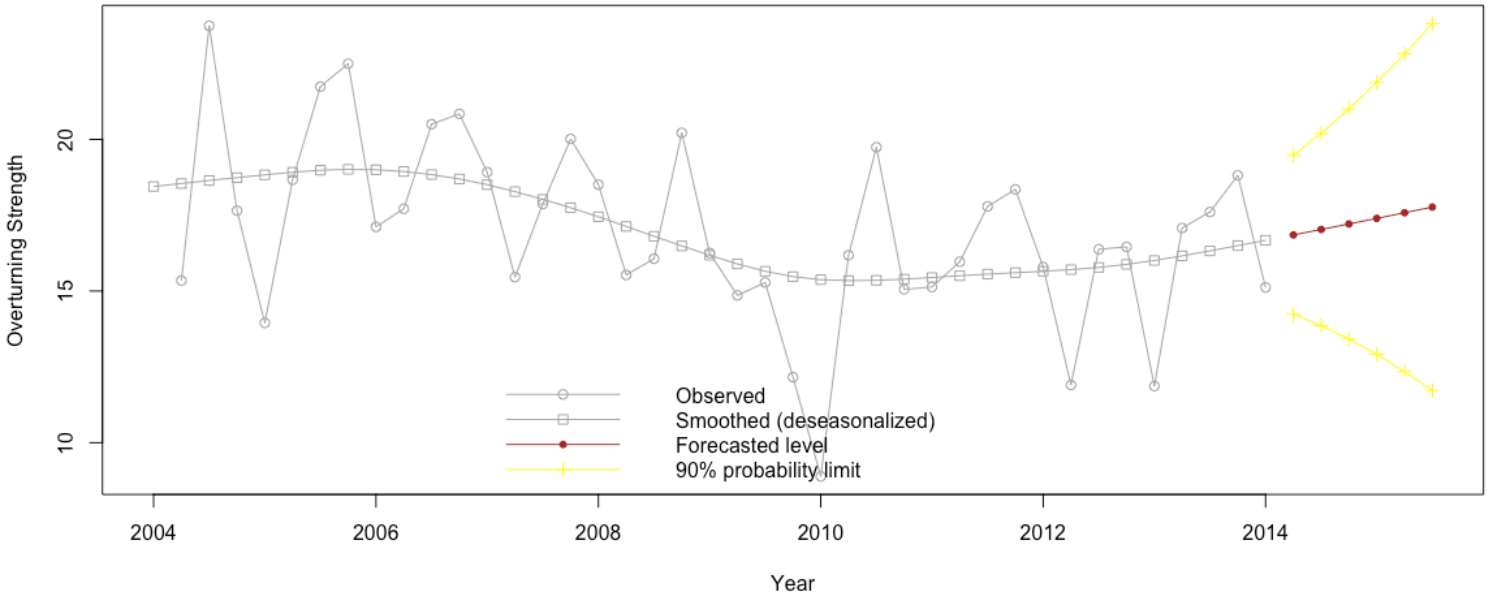
Indeed, whilst the forecast looks viable, the ARMA model has not made the data stationary, which is perhaps shown through the pattern of the data looking odd compared to the ARIMA and actual time series data. Overall, based on the ADF results and forecasts, the seasonal ARIMA model seems the best model.

II)

A dynamic linear model is a special type of state space model, where the errors of the state and observed components are normally distributed. To do this a Kalman filter and smoother are fitted. The filter computes filtered values of the state vectors, alongside their variance/covariance matrices, meaning filtering is estimating the current values of the state from past and current observations. The smoother computes the smoothed equivalent, and is thus estimating the past values of the state given the observations. Shown below is a decomposition plot for the data with a fitted DLM, but only the smoother has been applied, as the filter is later used for forecasting.



Daanish Ahsan
Statistical Modelling in Space and Time
Candidate Number: 124070
There is a clear trend, and seasonal pattern shown in the diagram.



Lastly, shown above is a plot of the applied DLM model, alongside forecasts for the next 6 quarters and the observed data. The smoothed line reflects the trend line from the decomposed plot, as expected, and broadly follows the trend of the observed data. The forecast seems accurate, as it follows the anticipated pattern of the smoothed line, and emulates the trend line from the decomposed diagram. Thus we can say with 90% certainty that the forecast for the next 6 quarters will fall between these yellow lines.

Daanish Ahsan
Statistical Modelling in Space and Time
Candidate Number: 124070

Code Appendix

```
library(astsa)
library(tseries)
library(dlm)
library(forecast)

set.seed(123)

rapid <- read.csv('moc_transports.csv')
#Now take quarterly means (the dlm code does not work for the whole dataset - too many points)
rapid$qyyyy <- paste(rapid$year,rapid$Quarter, sep='-')
rapidmean <- tapply(rapid$Overturning_Strength,rapid$qyyyy,mean)
#Now convert it to a time series object
rapidmean.ts <- ts(as.vector(rapidmean),start=c(2004,2),frequency = 4)

#Plotting the data
plot(rapidmean.ts, ylab='Overturning Strength', xlab='Year')

adf.test(rapidmean.ts, alternative = 'stationary', k = trunc((length(rapidmean.ts)-1)^(1/3))) #null is that data has unit root and is non stationary
#alternative is that it is stationary

#p value of 0.406 >0.05 means cant reject ho, so data is there is unit root and so data is non stationary

#decomposing time series into seasonal, trend, and irregular componenents using moving averages
m <- decompose(rapidmean.ts, type = c('additive', 'multiplicative'))

plot(m)

#seasonality range is significant compared to trend range, so we want a seasonal model

#acf is used to determine the MA order (q), pacf is used to determine AR order (p)
par(mfrow=c(1,2))
acf(rapidmean.ts,lag.max=30)
pacf(rapidmean.ts,lag.max=30)

#Automatic model selection for ARIMA for all of data

potato <- auto.arima(rapidmean.ts)
```

```
potato[["arma"]] #7 order values imply it should be a seasonal arima model
 which makes sense
#order values are 3, 0, 0, 1, 4, 1, 0

#forecasting the next 6 quarters
plot(forecast(potato, h=6), ylab = 'Overturning Strength', xlab = 'Year')

#model diagnostics

accuracy(potato)

summary(potato)

AIC(potato) #AIC of 190.6

#repeating but with seasonality as false

potato2 <- auto.arima(rapidmean.ts, seasonal = FALSE)

potato2[["arma"]]
#order values are 0, 0, 0, 1, 4, 1, 0
plot(forecast(potato2, h=6), ylab = 'Overturning Strength', xlab = 'Year') #
 this is a poor forecast

#model diagnostics

accuracy(potato2)

summary(potato2)

AIC(potato2) #AIC of 210.7

#splitting into training and test data to see how good auto arima model is on
 limited data

test <- rapidmean.ts[1:15] #these 15 quarters are test data

test <- ts(test)

#Fitting Arima model onto test data

testarima <- arima(test, order = c(3, 0, 0), seasonal=list(order=c(1, 4, 1),
 period=0))

plot(forecast(testarima, h=6), ylab = 'Overturning Strength', xlab = 'Year')

#predicted values from this test data model for Q16-20 are between 20 and
 34.9,
```

```
#compared to real Q16-20 values which are between 15.5 and 20.2, so model
 doesnt seem great w just test data

AIC(testarima) #AIC of 90

##ARMA- adjust order till lowest AIC, and plotting
rec.arma <- arma(rapidmean.ts, order = c(1, 1))
summary(rec.arma)
plot(forecast(rec.arma[["fitted.values"]], h=6), ylab = 'Overturning
 Strength', xlab = 'Year') #plot looks alright for arma, use this one

#DLM
plot(rapid$Overturning_Strength, rapid$year) #plot of overturning vs year

#function for DLM model
dlmco2 <- dlmModPoly(3) + dlmModSeas(4)
buildFun <- function(x) {
  diag(W(dlmco2))[2:4] <- exp(x[1:3])
  V(dlmco2) <- exp(x[4])
  return(dlmco2)
}

fit <- dlmMLE(rapidmean.ts, parm = rep(0, 4), build = buildFun)
dlmco2 <- buildFun(fit$par) #fitting dlm model
drop(V(dlmco2))
diag(W(dlmco2))[2:11]

co2Smooth <- dlmSmooth(rapidmean.ts, mod = dlmco2) #applying Kalman Smoother
xs <- cbind(rapidmean.ts, dropFirst(co2Smooth$s[,c(1,4)]))
colnames(xs) <- c("Overturning Strength", "Trend", "Seasonal")
plot(xs, type = 'o', main = "Overturning Strength") #decomposition plot

co2filt <- dlmFilter(rapidmean.ts,mod=dlmco2) #applying Kalman filter
co2fore <- dlmForecast(co2filt,nAhead=6)

sqrtR <- sapply(co2fore$R, function(x) sqrt(x[1,1]))
pl <- co2fore$a[,1] + qnorm(0.05, sd = sqrtR) #confidence intervals
pu <- co2fore$a[,1] + qnorm(0.95, sd = sqrtR) #confidence intervals
x <- ts.union(window(rapidmean.ts, start = c(500, 1)),
 window(co2Smooth$s[,1], start = c(500, 1)),co2fore$a[,1], pl, pu)

#plotting observed, smoothed, forecasts, and 90% probabilty limit
plot(x, plot.type = "single", type = 'o', pch = c(1, 0, 20, 3, 3),col =
 c("darkgrey", "darkgrey", "brown", "yellow", "yellow"), ylab = "Overturning
 Strength")
legend("bottomright", legend = c("Observed", "Smoothed (deseasonalized)",
 "Forecasted level", "90% probability limit"), bty = 'n', pch = c(1, 0, 20, 3,
 3), lty = 1,col = c("darkgrey", "darkgrey", "brown", "yellow", "yellow"))
```