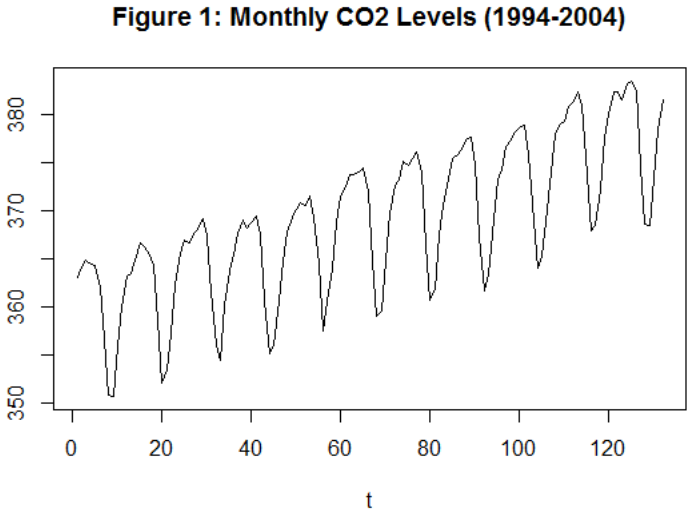
Sankeerti Haniyur
March 2016

Forecast of CO_2 Levels for every Month in 2005

The data set consists of monthly CO_2 levels Alert, Northwestern Territories, Canada from 1994 - 2004. Using R and various packages, I was able to forecast the CO_2 levels for every month in 2005. The R code and any necessary plots or R output are included at every step of the way.

Figure 1 shows a graph of the original data that was obtained using the code below:

```
library(TSA)
data(co2)
x = as.vector(co2)
n = length(x)
t = 1:n

# plot of the original data
plot(t, x, type="l", main="Figure 1: Monthly CO2 Levels (1994-2004)", ylab="")
```



Just by looking at the graph, it is clear that there is an upward trend in the data as well as a strong seasonal component that repeats itself regularly.

The first thing I did was fit a linear model to the data to get rid of the trend. I then checked the residuals to make sure the fit was accurate. Figure 2 shows a graph of the original data with the fitted trend added. Figure 3 shows the residuals after the trend is removed. The code used was:

```
# remove the trend by fitting a linear model
trend.fit = lm(x~t)

plot(t, x, type="l", main="Figure 2: Original Data with Fitted Trend", ylab="")
lines(t,fitted(trend.fit)) # add line to plot
```

```
# plot of residuals after trend is removed
y = residuals(trend.fit)
plot(t,y, type="l", main="Residuals After Trend is Removed", ylab="")
```

Figure 2: Original Data with Fitted Trend

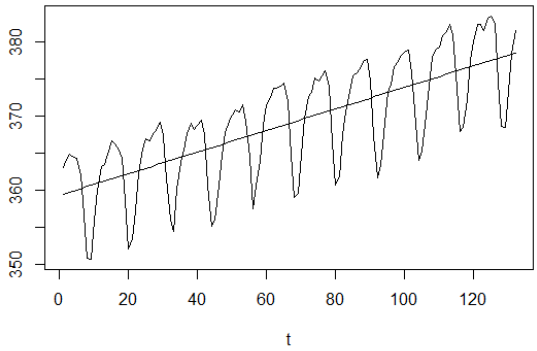
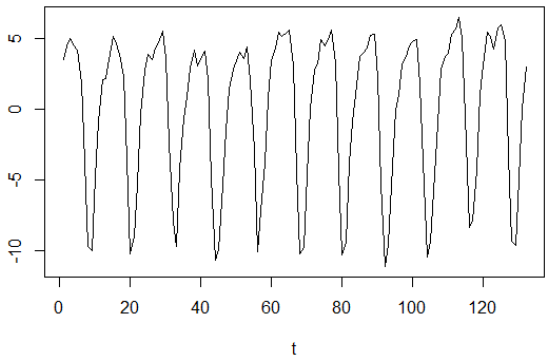


Figure 3: Residuals After Trend is Removed

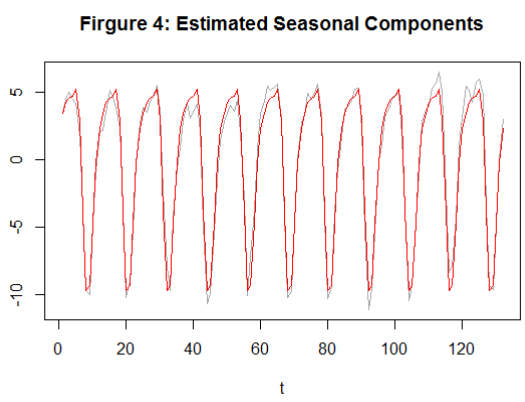


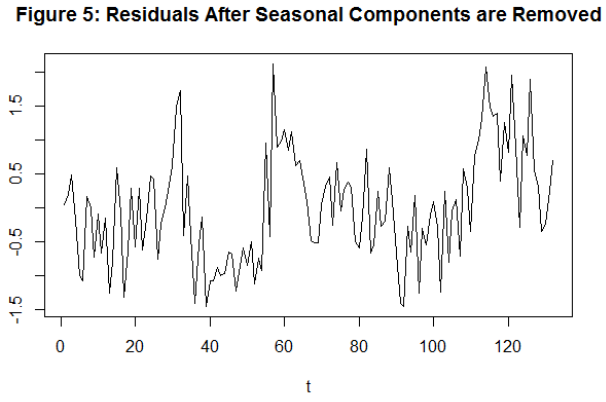
Once the trend component was taken care of, I moved on to the seasonal component. I used sum of harmonics to remove the seasonality and set d = 12 since we are working with monthly data. Once I fit it on all sines and cosines, I used stepwise regression to find the model with the lowest AIC which was the deciding factor on the "best" model. Here is the code used below:

```
# use t that is in the interval [0,1]
n = length(t)
t = 1:length(y)
t = (t) / n

# make matrix of the harmonics
d = 12 # number of time points in each season
n.harm = 6 # set to [d/2]
harm = matrix(nrow=length(t), ncol=2*n.harm)
for(i in 1:n.harm){
  harm[,i*2-1] = sin(n/d * i *2*pi*t)
  harm[,i*2] = cos(n/d * i *2*pi*t)
}
colnames(harm) = paste0(c("sin", "cos"), rep(1:n.harm, each = 2))

# fit on all of the sines and cosines
dat = data.frame(y, harm)
fit = lm(y~., data=dat)
summary(fit)

# setup the full model and the model with only an intercept
full = lm(y~.,data=dat)
reduced = lm(y~1, data=dat)

# stepwise regression starting with the full model
fit.back = step(full, scope = formula(reduced), direction = "both")
```

I graphed the estimated seasonal components as well as the residuals of the model after the seasonal component was removed. The code used was:

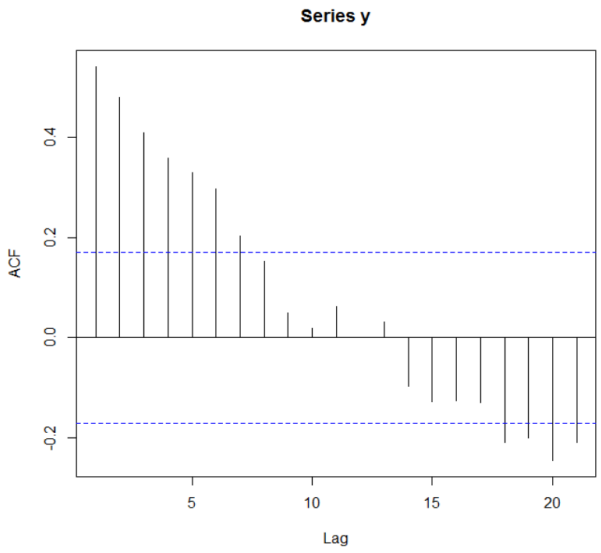
```
# get back the original t so that we can plot over this range
t = 1:n

# plot the estimated seasonal components
plot(t, y, type="l", col="darkgrey", main = "Firgure 4: Estimated Seasonal
Components", ylab="")
lines(t, fitted(fit.back), col="red")

# plot the residuals after seasonal component is removed
ts.plot(residuals(fit.back), main="Figure 5: Residuals After Seasonal Components are
        Removed", ylab = "", xlab="t")
```

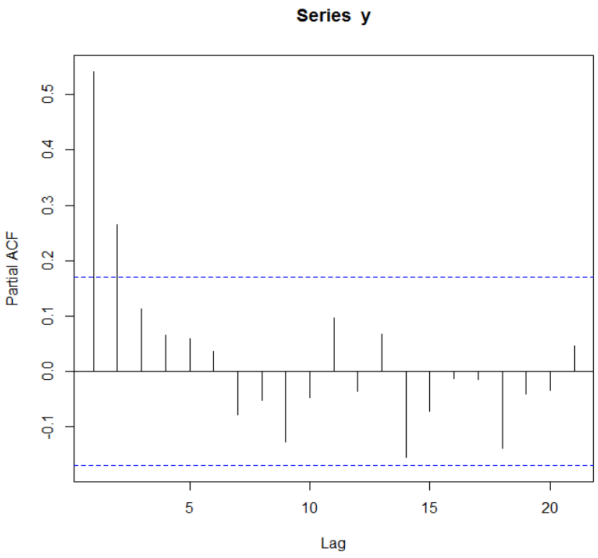




Since the residuals look a little skewed, I checked the ACF and PACF plots of the data to see if there was any dependence structure left in them. Here are the graphs and the code:

```
# plot the acf and pacf of the residuals
y = residuals(fit.back)
par(mfrow = c(1,2))
acf(y)
pacf(y)
```





The ACF plot showed some dependence structure towards the later lags, but that may have been due to Type 1 error. Therefore, I used the KPSS test to further test the residuals. Here is the R code and output below:

```
kpss.test(y)
```

```
        KPSS Test for Level Stationarity

data:  y
KPSS Level = 0.7543, Truncation lag parameter = 2, p-value = 0.01

Warning message:
In kpss.test(y) : p-value smaller than printed p-value
```

Since the p-value is small, we must reject the null hypothesis that the residuals are stationary. I then fitted a model to the residuals using the auto.arima function in R in order to make the residuals stationary. Here is the code and output:

```
arma.fit = auto.arima(y, allowmean = F, trace=T, stepwise=F)
```

```
Series: y
ARIMA(0,1,1)

Coefficients:
          ma1
      -0.5973
s.e.   0.0751

sigma^2 estimated as 0.4519:  log likelihood=-134.08
AIC=272.16   AICc=272.25   BIC=277.91
```

As shown above, the function fit an ARIMA(0,1,1) model. I then examined the residuals of this model to make sure that there is enough to reject independence. I did so using the Ljung-Box test whose null hypothesis is that there is independence. The code and output are:

```
wn = resid(arma.fit)
Box.test(wn, type="Ljung-Box",lag = 24)
```

```
        Box-Ljung test

data:  wn
X-squared = 19.569, df = 24, p-value = 0.721
```

Since there is a large p-value, we can fail to reject the null hypothesis of independence and conclude that the residuals are white noise. Next, I needed to make sure that I tested the validity of the forecast intervals before I even began the forecasting process. I used the Shapiro-Wilk test to do so, and it tells whether or not the residuals are normal. Here is the code and output:

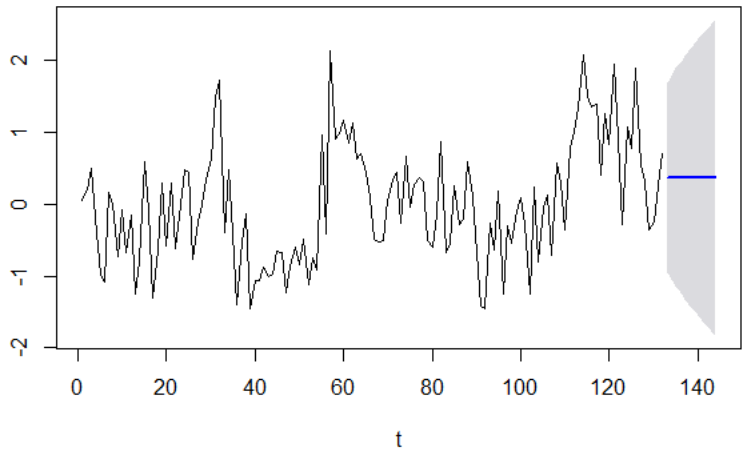
```
shapiro.test(wn)
```

```
        Shapiro-Wilk normality test

data:  wn
W = 0.98866, p-value = 0.3524
```

Due to the large p-value, we can fail to reject the null hypothesis and conclude that the residuals are normal and the model is fit for forecasting. I used the forecast function to predict the next 12 months and limited the level of confidence to 0.95 since I only wanted a 95% confidence interval to be shown on the plot. The code and Figure 6 plot of the forecast are shown below:

```
# forecast the next year
fc = forecast(arma.fit, 12, level = 0.95)
plot(fc, main= "Figure 6: Forecasts from ARIMA(0,1,1)")
```

Figure 6: Forecasts from ARIMA(0,1,1)



The forecast at this point do not include any trend or seasonality, so I had to add them back in. With the seasonal component, I took the fitted values from the stepwise regression function I acquired earlier. Also, I estimated that the noise (residuals) would be the mean of the residuals of the forecasts and used the predict function to estimate the trend using the code below:

```
# forecast the seasonal component and noise
season.fc = fit.back$fitted.values[1:12]

# forecast the trend
trend.fc = predict(trend.fit, newdata = data.frame(t = 133:144))

# add the seasonal and trend forecasts to get final forecasts
x.hat = season.fc + trend.fc + fc$mean
```

I then plotted the forecasts alongside the original data and added the forecast intervals to the plot. The intervals are distinguished as dashed lines. I have provided a zoomed in version as well so it is more clearly visible. The code and plots are given below in Figures 7 & 8:

Figure 7: Forecasts for 2005

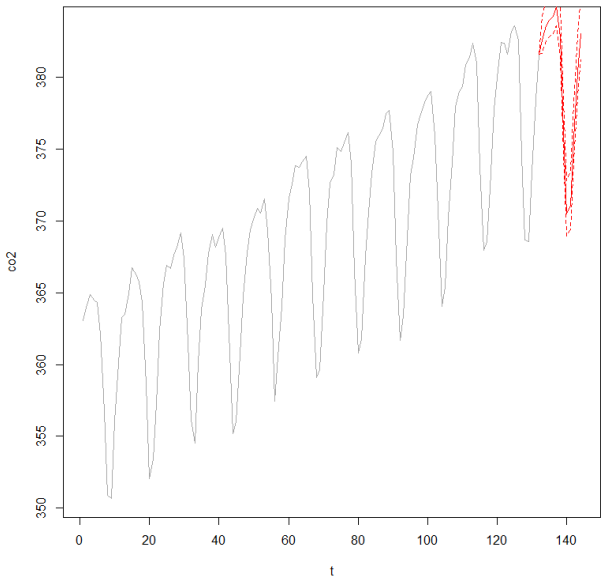
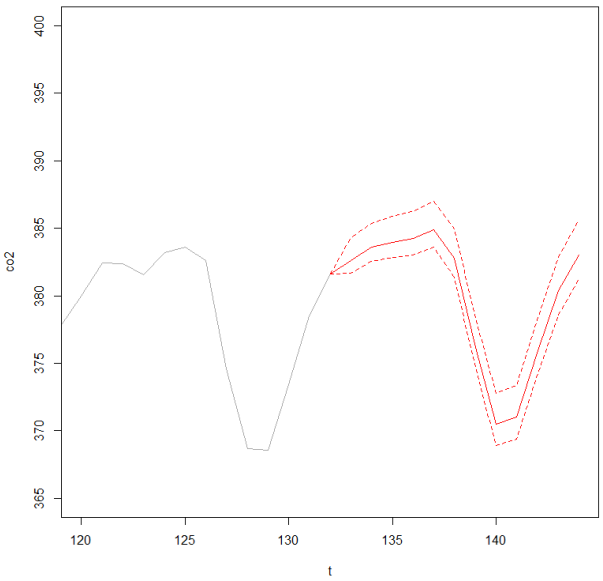


Figure 8: Forecasts for 2005 (Zoomed-in Version)



These are my final forecasts for monthly CO_2 levels at Alert, Northwest Territories, Canada. The results show that 2005 should likely follow the same increasing trend and seasonal changes in CO_2 levels that were previously experienced. The lower forecast interval is approximately the same as the data in 2004. The upper forecast interval predicts the levels of carbon dioxide will be much higher than the previous year, which hopefully is not the case.

Now, just to show that my method is sound, I also used pretty much the exact same code and steps to forecast the 2004 data and plot it against the actual data. All of the code for this is given in the code appendix at the end of the report, however, only a few lines are changed from the code given throughout this report. Here is the zoomed in version of the results of this trial:

Figure 9: Forecasts for 2004

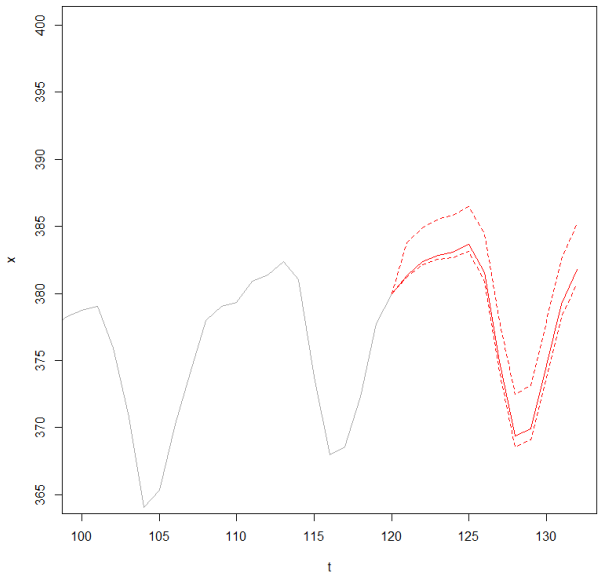
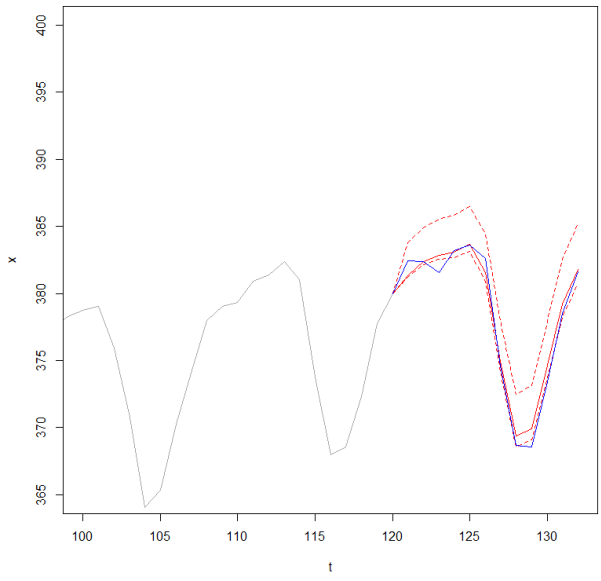


Figure 10: Forecasts for 2004



The blue line in Figure 10 represents the actual 2004 data, and as you can see, the forecasted values are actually very close. The actual data almost hugs the lower forecast interval meaning that the actual levels were on the lower end of what was predicted. Hopefully this is the case for 2005 as well. However, there could always be some unexpected dips and rises in the real data because of natural causes. Nevertheless, it is useful when we want to be prepared for the future.

Finally, here is a quick summary of the steps I took to achieve both of these results:

- Plotted original data
- Removed trend using linear model
    - > Plotted residuals after trend was removed
- Used sum of harmonics and stepwise regression to remove seasonal component
    - > Plotted estimated seasonal components
- Plotted residuals after both trend and seasonality were removed
    - > Utilized KPSS test and there was some dependence left in the residuals
- Auto.arima helped fit a ARIMA(0,1,1) model to the residuals
    - > Used Ljung-Box test to assure the independence of the residuals after fit
- Verified the validity of the forecasts using the Shapiro-Wilk test
- Used forecast function to forecast monthly CO_2 levels in the next year (2005)
- Added estimates of the trend, seasonal, and noise components to get final forecasts
- Plotted results

I hope this information will be useful to you and the analysis helped demystify the data set. Thank you for your time.

Best Regards,
Sankeerti Haniyur

Code Appendix

```
library(TSA)
data(co2)
x = window(co2, start=c(1994,1), end=c(2003,12))
x = as.vector(x)
n = length(x)
t = 1:n

# plot of the original data
plot(t, x, type="l", main="Figure 1: Monthly CO2 Levels (1994-2003)", ylab="")

# remove the trend by fitting a linear model
trend.fit = lm(x~t)

plot(t, x, type="l", main="Figure 2: Original Data with Fitted Trend", ylab="")
lines(t,fitted(trend.fit)) # add line to plot

# plot of residuals after trend is removed
y = residuals(trend.fit)
plot(t,y, type="l", main="Figure 3: Residuals After Trend is Removed", ylab="")

# use t that is in the interval [0,1]
n = length(t)
t = 1:length(y)
t = (t) / n

# make matrix of the harmonics
d = 12 # number of time points in each season
n.harm = 6 # set to [d/2]
harm = matrix(nrow=length(t), ncol=2*n.harm)
for(i in 1:n.harm){
  harm[,i*2-1] = sin(n/d * i *2*pi*t)
  harm[,i*2] = cos(n/d * i *2*pi*t)
}
colnames(harm) = paste0(c("sin", "cos"), rep(1:n.harm, each = 2))

# fit on all of the sines and cosines
dat = data.frame(y, harm)
fit = lm(y~., data=dat)
summary(fit)

# setup the full model and the model with only an intercept
full = lm(y~.,data=dat)
reduced = lm(y~1, data=dat)

# stepwise regression starting with the full model
fit.back = step(full, scope = formula(reduced), direction = "both")

# get back the original t so that we can plot over this range
t = 1:n

# plot the estimated seasonal components
plot(t, y, type="l", col="darkgrey", main = "Firgure 4: Estimated Seasonal
Components", ylab="")
```

```
lines(t, fitted(fit.back), col="red")

# plot the residuals after seasonal component is removed
ts.plot(residuals(fit.back), main="Figure 5: Residuals After Seasonal Components are
Removed",
        ylab = "", xlab="t")

# plot the acf and pacf of the residuals
y = residuals(fit.back)
par(mfrow = c(1,2))
acf(y)
pacf(y)
par(mfrow = c(1,1))
kpss.test(y)
# looks like there is some dependence left so need to fit a model to them

# Use H-K algorithm to determine best model
require(forecast)
arma.fit = auto.arima(y, allowmean = F, trace=T, stepwise=F)
arma.fit

# examine the residuals of the arma fit
wn = resid(arma.fit)
Box.test(wn, type="Ljung-Box",lag = 24)
# large p-value indicates we fail to reject the null of independence

# verify the validity of the forecast intervals
shapiro.test(wn)
# large p-value indicates we fail to reject the null of normality

# forecast the next year
fc = forecast(arma.fit, 12, level = 0.95)
plot(fc, main= "Figure 6: Forecasts from ARIMA(0,1,1)")

# forecast the seasonal component and noise
season.fc = fit.back$fitted.values[1:12]+fc$mean

# forecast the trend
trend.fc = predict(trend.fit, newdata = data.frame(t = 121:132))

# add the seasonal and noise forecasts
x.hat = season.fc + trend.fc

x = window(co2, start=c(1994,1), end=c(2003,12))
par(mfrow = c(1,2))

# zoomed in version 1
# plot the forecasts after the actual data
plot(t, x, xlim = c(100,132), ylim = c(365, 400), type="l", col="darkgrey",
     main = "Figure 9: Forecasts for 2004")
lines(120:132, c(x[120], x.hat), col="red")
#add the forecast intervals
lines(120:132, c(x[120], x.hat+fc$lower), col="red", lty=2)
lines(120:132, c(x[120], x.hat+fc$upper), col="red", lty=2)
```

```
# zoomed in version 2
# plot the forecasts after the actual data
plot(t, x, xlim = c(100,132), ylim = c(365, 400), type="l", col="darkgrey",
     main = "Figure 10: Forecasts for 2004")
lines(120:132, c(x[120], x.hat), col="red")
#add the forecast intervals
lines(120:132, c(x[120], x.hat+fc$lower), col="red", lty=2)
lines(120:132, c(x[120], x.hat+fc$upper), col="red", lty=2)

# now add a line for the actual 2004 data
lines(120:132, co2[120:132], col = "blue")
```