Temperature Analysis Abidjan

Karen Wood

July 28, 2017

Climate Change Data Analysis

This is an example of fitting data with an ARIMA model and classical decomposition.

First import data:

```
library(readr)
GlobalLandTemperaturesByMajorCity <-
  read_csv(paste0("C:/Users/Karen/Google Drive/Kaggle/Climate Change/",
                  "climate-change-earth-surface-temperature-data/",
                  "GlobalLandTemperaturesByMajorCity.csv"))
```

```
## Parsed with column specification:
## cols(
##   dt = col_date(format = ""),
##   AverageTemperature = col_double(),
##   AverageTemperatureUncertainty = col_double(),
##   City = col_character(),
##   Country = col_character(),
##   Latitude = col_character(),
##   Longitude = col_character()
## )
```

This data is from Kaggle, posted by Berkeley Earth. It was downloaded on July 28, 2017 at 3:25pm Pacific Time. In particular, I will examine the data from a single major city. The data appears to have temperatures in Celcius. Date format is in y-m-d format, but only 1 data point for each month, at least for the first city: Abidjan.

I will be focusing my analysis on the first city: Abidjan.

```
AbidjanData<-
  GlobalLandTemperaturesByMajorCity[GlobalLandTemperaturesByMajorCity$City=="Abidjan", ]
```

I will start by checking for missing data.

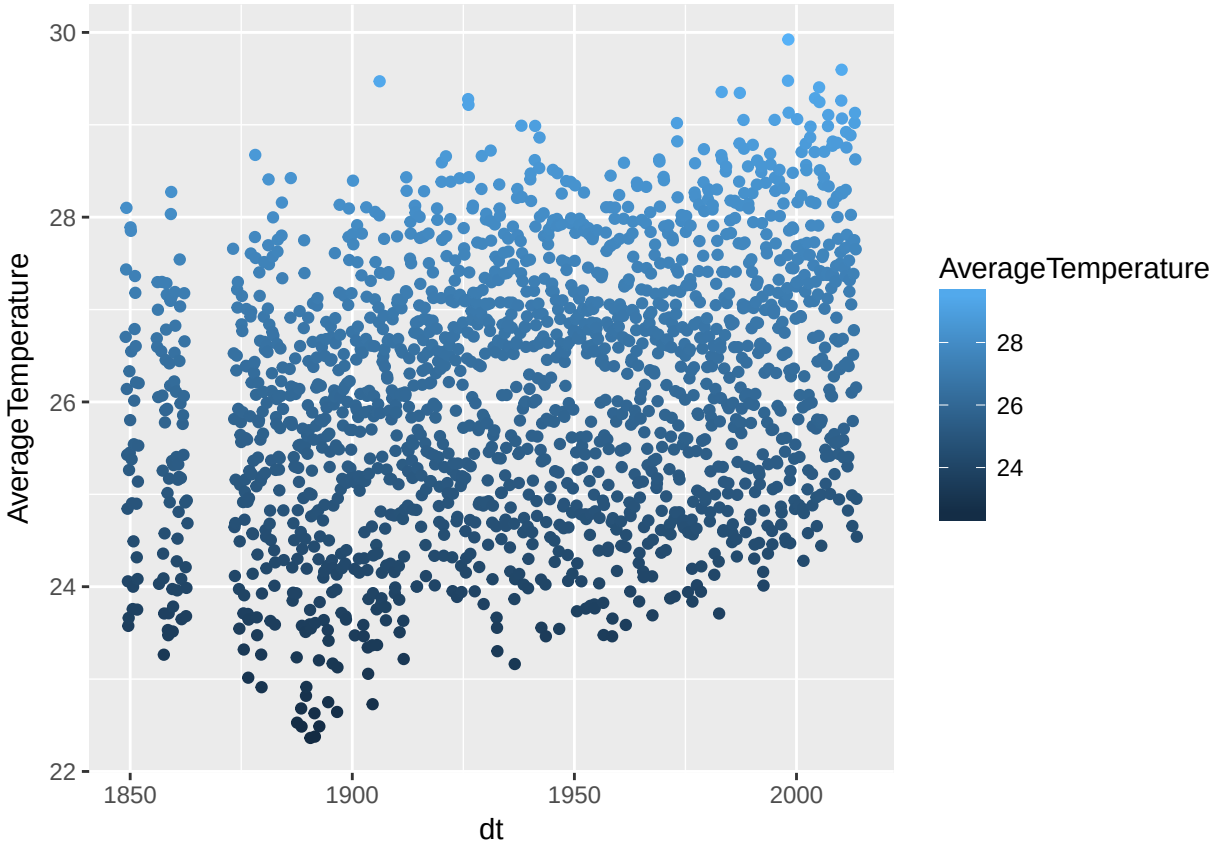
```
sum(is.na(AbidjanData$AverageTemperature))
```

```
## [1] 200
```

There are 200 NAs. I will first plot the data to see what the situation is. The dates are already in the format I would like to use to plot, so I do not need to change them.

```
library(ggplot2)
ggplot(AbidjanData, aes(dt,AverageTemperature))+
  geom_point(aes(color=AverageTemperature))
```

```
## Warning: Removed 200 rows containing missing values (geom_point).
```



Much of the data missing is very old data. I will restrict to just data more recent than 1900. Instead, I could use the forecast package and use tsclean() to input the missing values. However, with such large swaths of data missing, I think it better to restrict the data for now.

```
AbidjanData2<-AbidjanData[AbidjanData$dt>as.Date("1875-01-01"),]
```

Now how many missing pieces of data are there?

```
sum(is.na(AbidjanData2$AverageTemperature))
```

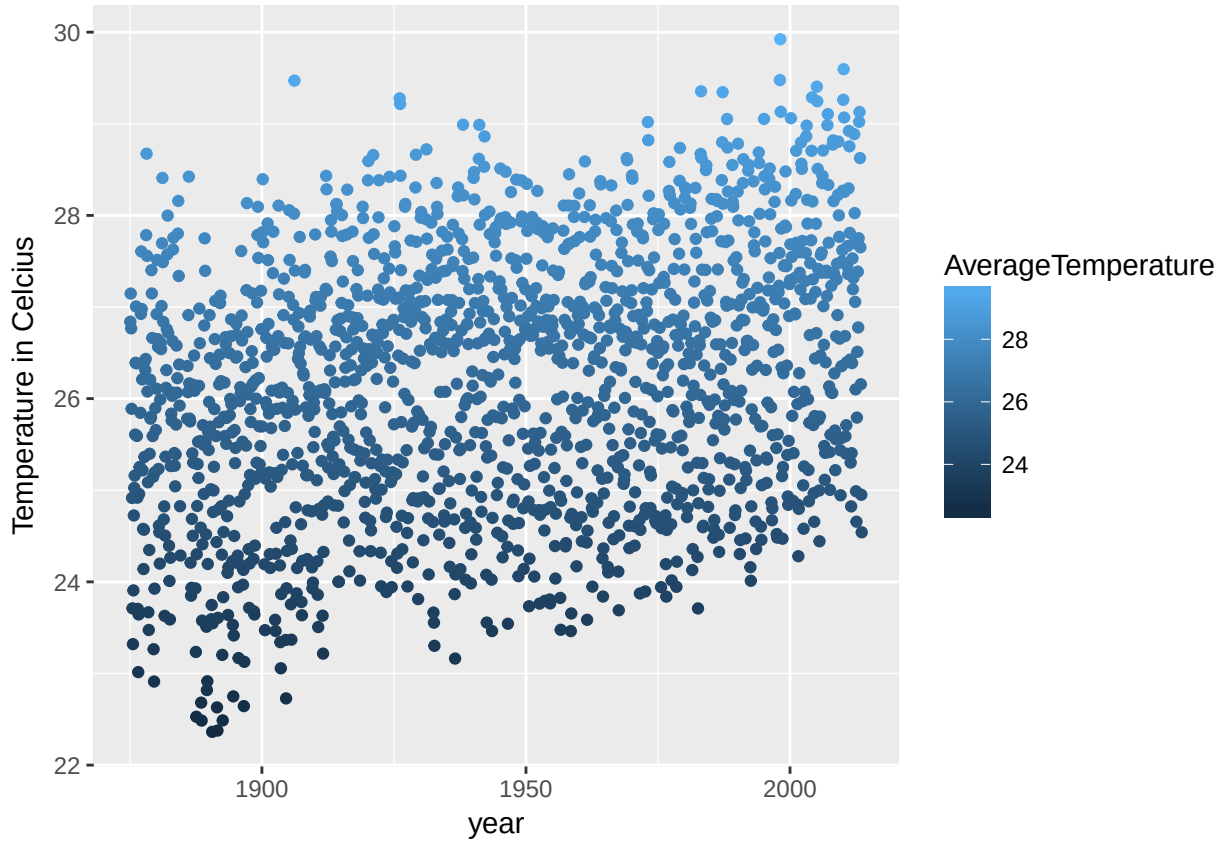
```
## [1] 15
```

Only 1. As it turns out, the last piece of data is missing.

I will now plot again.

```
ggplot(AbidjanData2, aes(dt,AverageTemperature))+
  geom_point(aes(color=AverageTemperature))+
  xlab("year")+ylab("Temperature in Celcius")
```

```
## Warning: Removed 15 rows containing missing values (geom_point).
```



I will now move on to the analysis by fitting an ARIMA model. I will now follow along with datascience.com/blog/introduction-to-forecasting-with-arima-in-r-learn-data-science-tutorials. They want to use the ggplot, forecast and tseries libraries, and we've already loaded ggplot.

```
library('forecast')
library('tseries')
```

I will start by make sure that the data is cleaned. They use count_ts=ts(AbidjanData2[, c('AverageTemperature')]), but I prefer different notation:

```
temp_ts=ts(AbidjanData2$AverageTemperature)
```

Now I will create a cleaned column:

```
AbidjanData2$cleaned_AverageTemperature= tsclean(temp_ts)
```

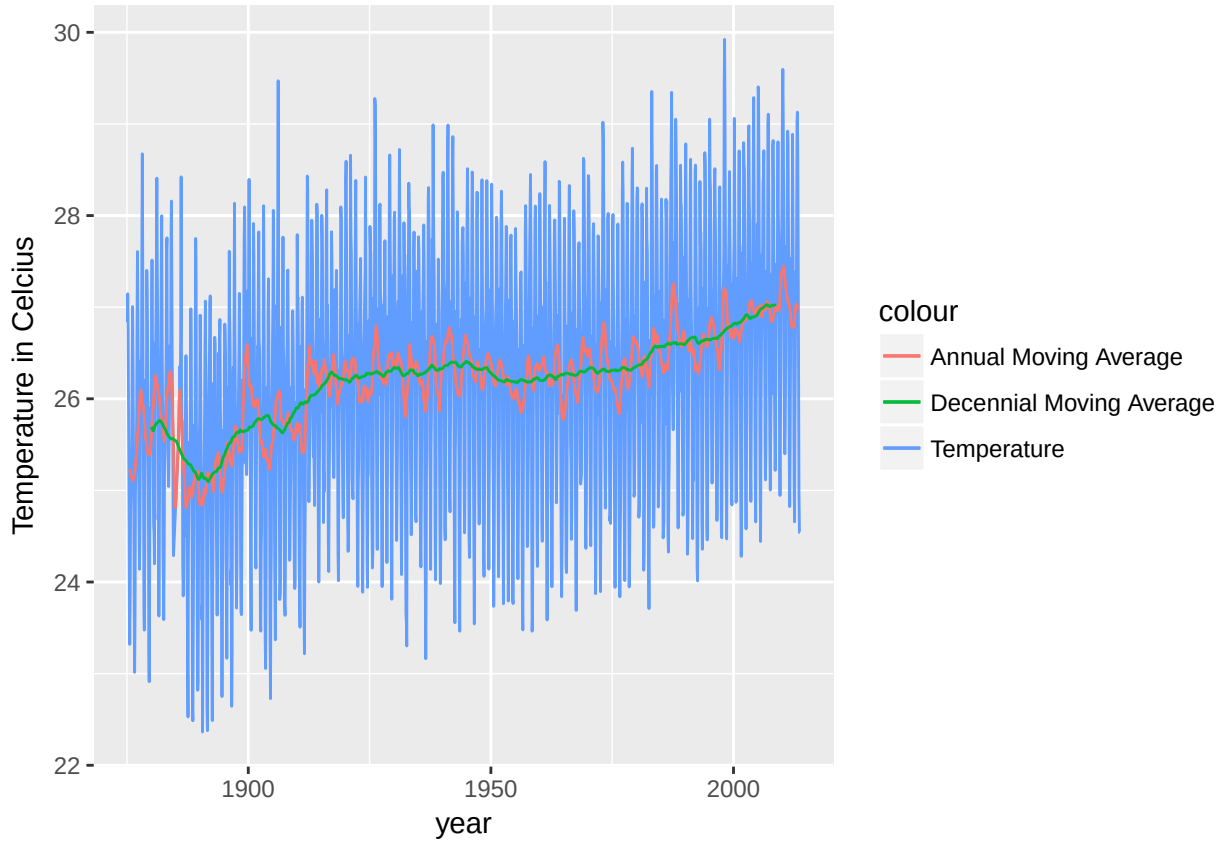
Before fitting the ARIMA model, I will first examing the moving average. R has a command: ma, which creates the moving average.

```
AbidjanData2$temp_ma=ma(AbidjanData2$cleaned_AverageTemperature,order=12)
AbidjanData2$temp_ma_decade=ma(AbidjanData2$cleaned_AverageTemperature,order=120)
```

I am using an order 12 for the 12 months of the year. This will give our annual average. Using an order 120 for 10 years. This will give the Decennial average.

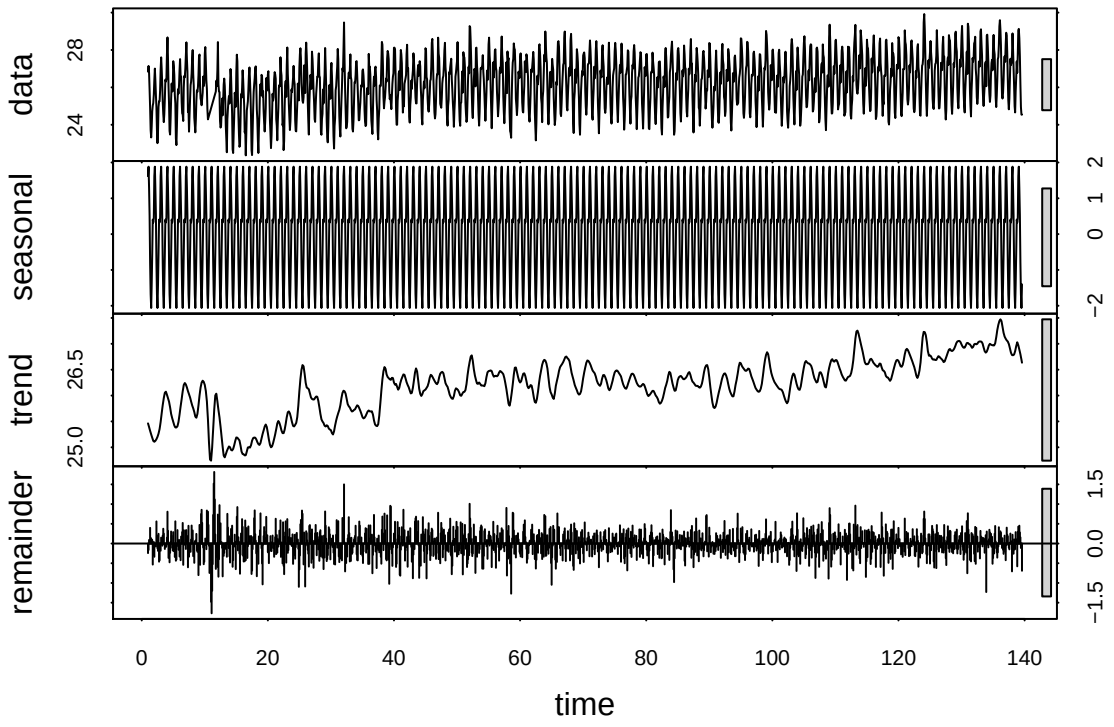
```
ggplot()+
  geom_line(data=AbidjanData2,
            aes(x=dt, y= cleaned_AverageTemperature, colour="Temperature"))+
  geom_line(data=AbidjanData2,
```

```
          aes(x=dt, y= temp_ma, colour="Annual Moving Average"))+
  geom_line(data=AbidjanData2,
          aes(x=dt, y= temp_ma_decade, colour="Decennial Moving Average"))+
  xlab("year")+ylab("Temperature in Celcius")
```

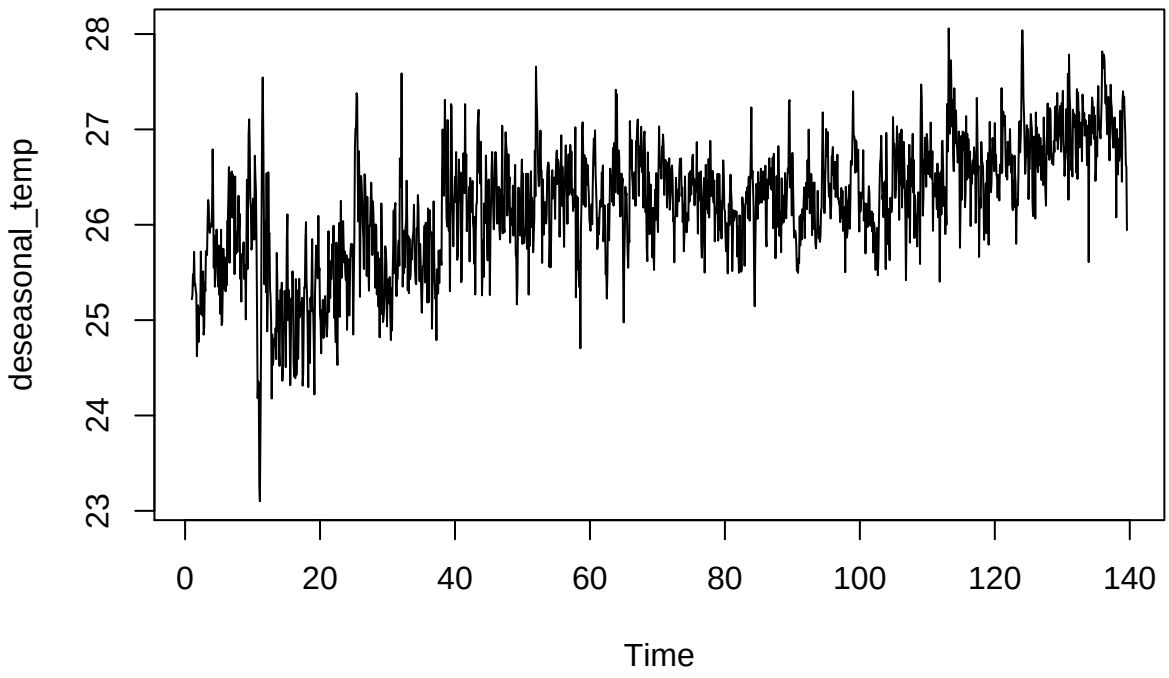


I will now calculate the seasonal component:

```
temp_ma=ts(na.omit(AbidjanData2$cleaned_AverageTemperature),frequency=12)
decomp=stl(temp_ma,s.window="periodic")
deseasonal_temp <- seasadj(decomp)
plot(decomp)
```




```
plot(deseasonal_temp)
```



In this case, we have 12 observations per period, since a year is a period and we only have 1 data point per month.

I will now check for stationarity. To do this I will use the augmented Dickey-Fuller (ADF) test. Since temperatures depend largely on the month, and we've adjusted by using the mean temperature over the year, we shouldn't see any seasonality, this data shoud now be stationary.

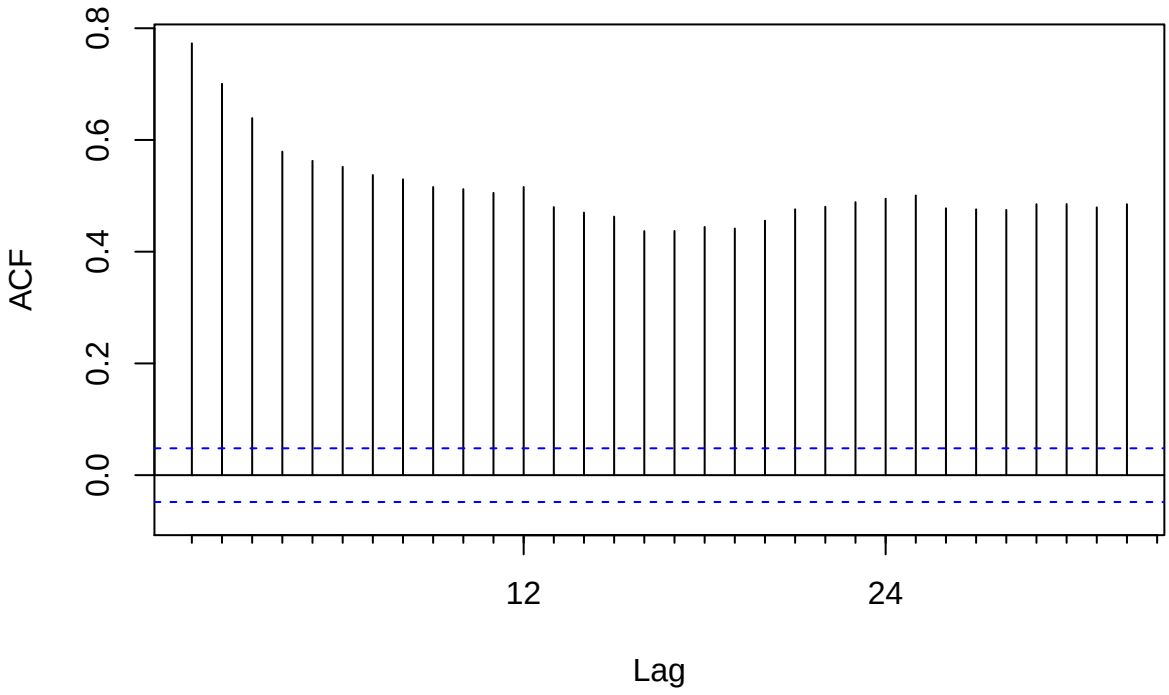
```
adf.test(deseasonal_temp, alternative= "stationary")
```

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

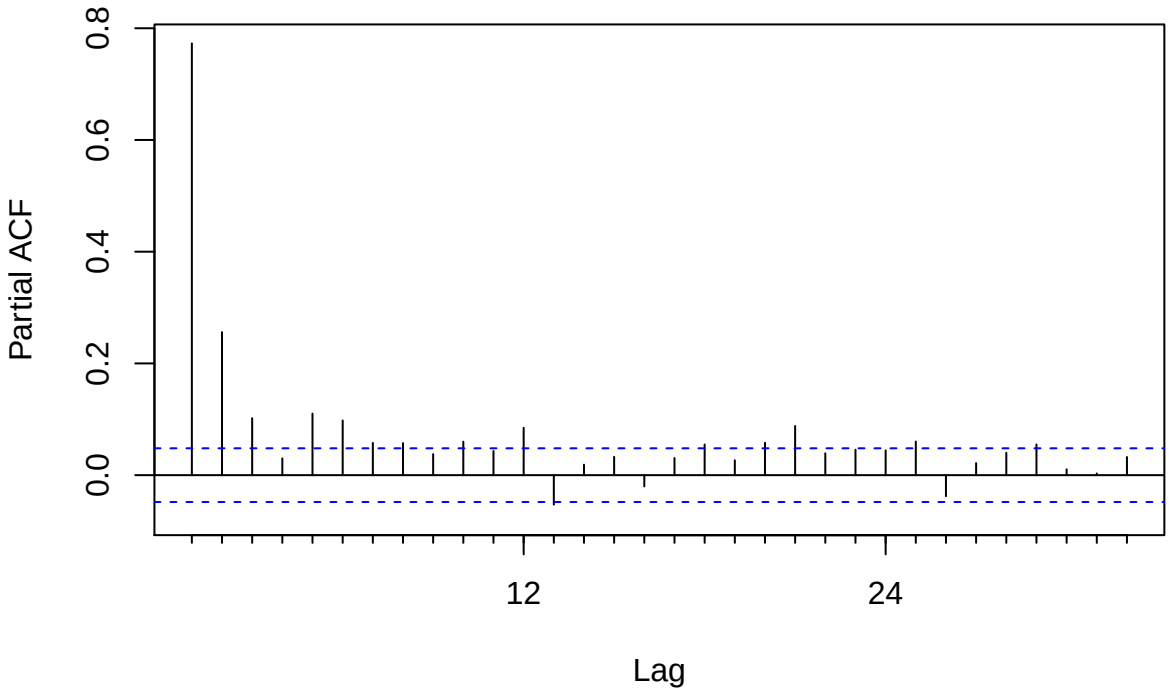
```
##
##  Augmented Dickey-Fuller Test
##
## data:  deseasonal_temp
## Dickey-Fuller = -7.3646, Lag order = 11, p-value = 0.01
## alternative hypothesis: stationary
```

The p-value of .01 indicates that we should assume that we reject the null hypothesis of non-stationarity. So I don't have to use differencing on the data. If we were concerned with these, we would want to examine the AcF and Pacf for any peaks, but we already have stationary data.

```
Acf(deseasonal_temp, main='')
```




```
Pacf(deseasonal_temp,main='')
```



Looking at the ACF, it looks like the data is not non-stationary. Let's use the KPSS test:

```
kpss.test(deseasonal_temp)
```

```
## Warning in kpss.test(deseasonal_temp): p-value smaller than printed p-value
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  deseasonal_temp
## KPSS Level = 9.8835, Truncation lag parameter = 9, p-value = 0.01
```

kpss thinks that it is non-stationary.

So, now I will use auto.arima:

```
myarima=auto.arima(deseasonal_temp,seasonal=FALSE)
summary(myarima)
```

```
## Series: deseasonal_temp
## ARIMA(1,1,3)            with drift
##
## Coefficients:
##          ar1      ma1     ma2      ma3  drift
##       0.6667  -1.1828  0.2606  -0.0583  9e-04
## s.e.  0.0544   0.0577  0.0440   0.0371  6e-04
##
## sigma^2 estimated as 0.1465:  log likelihood=-760.87
## AIC=1533.74   AICc=1533.79   BIC=1566.24
```

```
##
## Training set error measures:
##                       ME      RMSE       MAE        MPE     MAPE      MASE
## Training set 0.001317971 0.3820438 0.2918352 -0.0157162 1.117221 0.6032804
##                      ACF1
## Training set 0.0006877293
```

We will now plot the prediction.

```
q=forecast(myarima,h=120)
plot(q)
```