Durham Assignment - Predict Daily Temperature

Maonan Zhao

1. Loading in required packages and Data preparation

1.1 Loading data and the library

Before the analysis begins, I load in the packages required for the subsequent analysis.

```
library("dplyr")
library("tseries")
library("urca")
library("lmtest")
library("fBasics")
library("forecast")
```

In this section, I aim to load in the file that is given. A full inspection on the file gives 3 time series data with respect to time spanning across 1st Jan 1901 to 31st Dec 2019.

```
input_path = "C:/DU/E&E/Mini-Project/"
file_name = "durhamtemp_1901_2019.csv"
dt <- read.csv(paste(input_path,file_name, sep=""), sep=",", check.names = T)
dt
```

Year <int>	Month <int>	Day <int>	Date <chr>	PPT. <dbl>	Av.temp <dbl>	Tmax <dbl>	Tmin <dbl>
1901	1	1	01/01/1901	0.0	2.1	3.5	0.6
1901	1	2	02/01/1901	0.0	4.0	7.2	0.7
1901	1	3	03/01/1901	0.0	3.0	6.7	-0.7

Year <int>	Month <int>	Day <int>	Date <chr>	PPT. <dbl>	Av.temp <dbl>	Tmax <dbl>	Tmin <dbl>
1901	1	4	04/01/1901	0.0	3.7	7.9	-0.6
1901	1	5	05/01/1901	0.0	-0.3	1.2	-1.7
1901	1	6	06/01/1901	0.0	-0.1	2.3	-2.4
1901	1	7	07/01/1901	2.9	0.8	2.3	-0.8
1901	1	8	08/01/1901	3.1	0.0	1.1	-1.2
1901	1	9	09/01/1901	1.0	-4.4	-1.1	-7.6
1901	1	10	10/01/1901	0.0	1.7	4.6	-1.2

1-10 of 44,226 rows Previous **1** 2 3 4 5 6 ... 100 Next

```
head(dt)
```

	Year <int>	Month <int>	Day <int>	Date <chr>	PPT. <dbl>	Av.temp <dbl>	Tmax <dbl>	Tmin <dbl>
1	1901	1	1	01/01/1901	0	2.1	3.5	0.6
2	1901	1	2	02/01/1901	0	4.0	7.2	0.7
3	1901	1	3	03/01/1901	0	3.0	6.7	-0.7
4	1901	1	4	04/01/1901	0	3.7	7.9	-0.6
5	1901	1	5	05/01/1901	0	-0.3	1.2	-1.7
6	1901	1	6	06/01/1901	0	-0.1	2.3	-2.4

6 rows

```
tail(dt)
```

	Year <int>	Month <int>	Day <int>	Date <chr>	PPT. <dbl>	Av.temp <dbl>	Tmax <dbl>	Tmin <dbl>
44221	*NA*	*NA*	*NA*		*NA*	*NA*	*NA*	*NA*
44222	*NA*	*NA*	*NA*		*NA*	*NA*	*NA*	*NA*
44223	*NA*	*NA*	*NA*		*NA*	*NA*	*NA*	*NA*
44224	*NA*	*NA*	*NA*		*NA*	*NA*	*NA*	*NA*
44225	*NA*	*NA*	*NA*		*NA*	*NA*	*NA*	*NA*
44226	*NA*	*NA*	*NA*		*NA*	*NA*	*NA*	*NA*

6 rows

```
dt$Year<-as.factor(dt$Year)
dt$Month<-as.factor(dt$Month)

#summarise the data to monthly average
dt_monthly <-dt%>%
  group_by(Year,Month)%>%
  summarise(totalrain= sum(PPT.),
            meanavtemp=mean(Av.temp),
            meanTmax=mean(Tmax),
            meanTmin=mean(Tmin))
```

```
`summarise()` has grouped output by 'Year'. You can override using the `.groups` argument.
```

```
dt_monthly$Date <- paste("15/",dt_monthly$Month,"/",dt_monthly$Year)
dt_monthly
```

Year <fctr>	Mo… <fctr>	totalrain <dbl>	meanavtemp <dbl>	meanTmax <dbl>	meanTmin <dbl>	Date <chr>
1901	1	27.2	2.32903226	4.9870968	-3.612903e-01	15/ 1 / 1901
1901	2	28.9	2.03214286	4.5250000	-4.857143e-01	15/ 2 / 1901
1901	3	35.4	3.72903226	7.0225806	4.096774e-01	15/ 3 / 1901
1901	4	24.6	7.53333333	12.4366667	2.576667e+00	15/ 4 / 1901
1901	5	42.2	9.86451613	14.6645161	5.012903e+00	15/ 5 / 1901
1901	6	31.1	12.72333333	17.7700000	7.636667e+00	15/ 6 / 1901
1901	7	65.0	17.23870968	22.4258065	1.199032e+01	15/ 7 / 1901
1901	8	56.6	15.35806452	20.1709677	1.049677e+01	15/ 8 / 1901
1901	9	35.0	12.97666667	16.4566667	9.443333e+00	15/ 9 / 1901
1901	10	60.2	8.47419355	12.4032258	4.506452e+00	15/ 10 / 1901

1-10 of 1,429 rows Previous **1** 2 3 4 5 6 ... 100 Next

```
head(dt_monthly)
```

Year <fctr>	Month <fctr>	totalrain <dbl>	meanavtemp <dbl>	meanTmax <dbl>	meanTmin <dbl>	Date <chr>
1901	1	27.2	2.329032	4.987097	-0.3612903	15/ 1 / 1901
1901	2	28.9	2.032143	4.525000	-0.4857143	15/ 2 / 1901
1901	3	35.4	3.729032	7.022581	0.4096774	15/ 3 / 1901
1901	4	24.6	7.533333	12.436667	2.5766667	15/ 4 / 1901
1901	5	42.2	9.864516	14.664516	5.0129032	15/ 5 / 1901
1901	6	31.1	12.723333	17.770000	7.6366667	15/ 6 / 1901

6 rows

```
tail(dt_monthly)
```

Year <fctr>	Month <fctr>	totalrain <dbl>	meanavtemp <dbl>	meanTmax <dbl>	meanTmin <dbl>	Date <chr>
2019	8	81.2	16.496774	20.451613	12.519355	15/ 8 / 2019
2019	9	84.2	13.320000	17.573333	9.066667	15/ 9 / 2019
2019	10	88.4	8.938710	12.006452	5.858065	15/ 10 / 2019
2019	11	106.6	5.693333	8.150000	3.246667	15/ 11 / 2019
2019	12	32.8	5.432258	8.009677	2.825806	15/ 12 / 2019
NA	*NA*	*NA*	*NA*	*NA*	*NA*	15/ NA / NA

6 rows

Note that the “Date” column is does not imply the true value for that particular date.

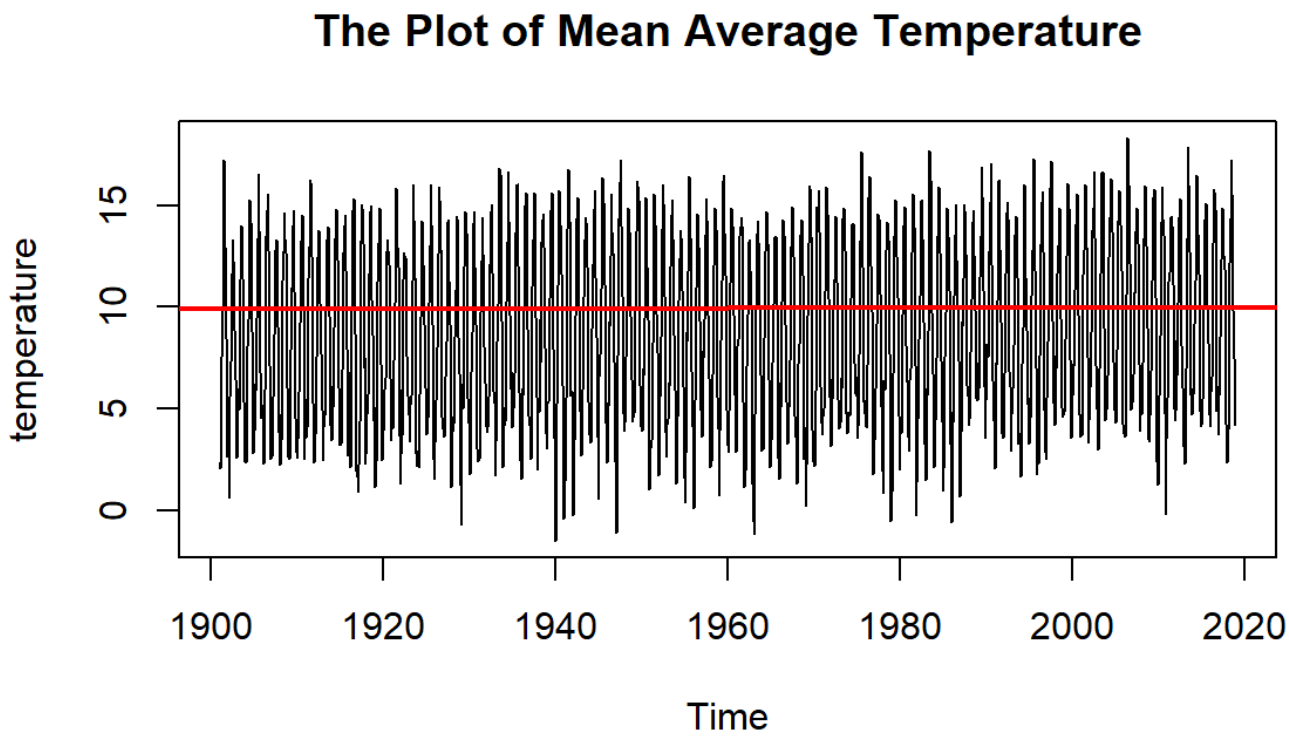
2. Analysis on the Time Series

In this section, there are 4 subsections. Subsection 1 is to show the monthly mean average temperature plot,and find the fit line . Subsection 2

2.1 Plot on the Series (Visualisation)

This is a time series plot of monthly average temperature in Durham from 1901 to 2019. It seems that there is a trend in the series (This may be due to global warming). Hence, this series may be a trend stationary series. We can de-trend it by fitting in a linear regression model and subtract the slope. This leads to subsection 2.2.

```
meantemp <- ts(dt_monthly$meanavtemp, start = 1901, end= 2019, frequency = 12)
plot(meantemp, type ="l", main = "The Plot of Mean Average Temperature", ylab="temperature")
abline(reg=lm(data =dt_monthly, meanavtemp~ c(1:nrow(dt_monthly))),col='red',lwd=2)
```

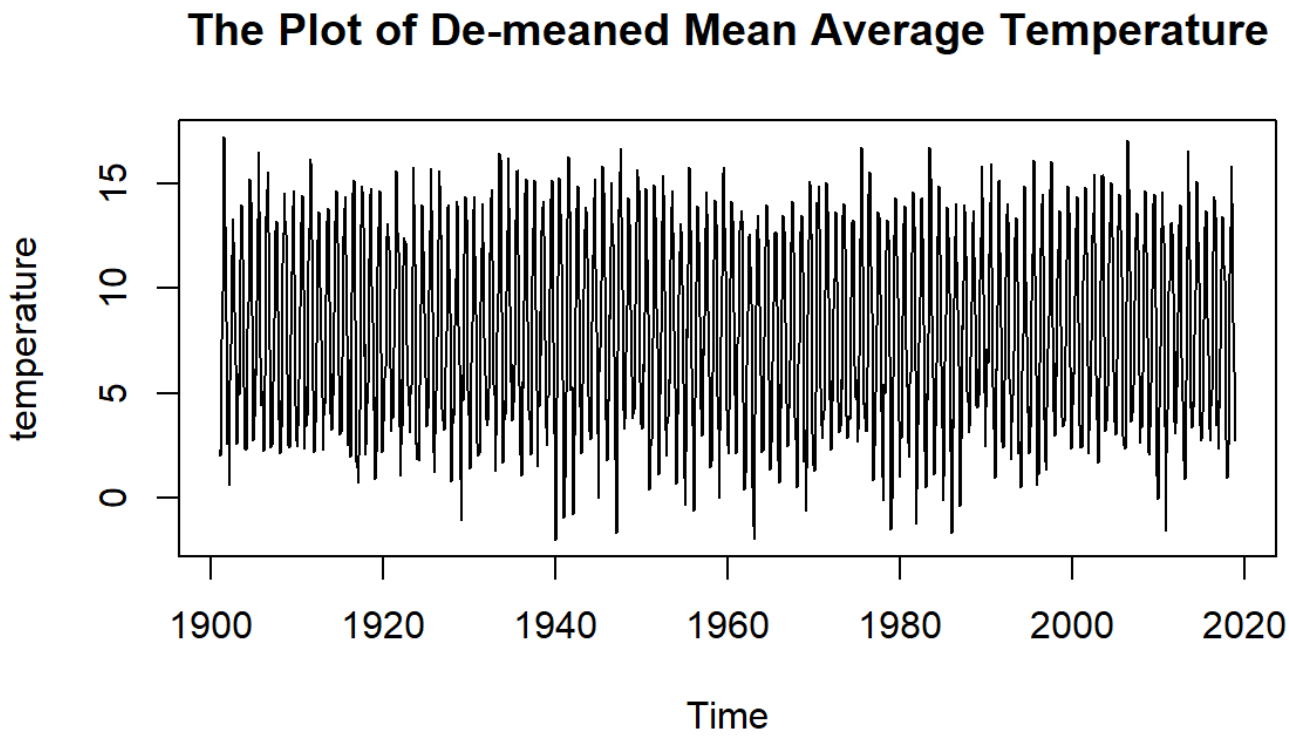


2.2 De-trending the Time Series

```
model_linear<-lm(data =dt_monthly, meanavtemp~ c(1:nrow(dt_monthly)))

summary(model_linear)
```

```
Call:
lm(formula = meanavtemp ~ c(1:nrow(dt_monthly)), data = dt_monthly)

Residuals:
    Min      1Q  Median      3Q     Max
-9.9598 -3.7616 -0.3242  4.1645  9.2588

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)           7.9728670  0.2343724  34.018  < 2e-16 ***
c(1:nrow(dt_monthly)) 0.0010107  0.0002841   3.557 0.000387 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.426 on 1426 degrees of freedom
  (因为不存在，1个观察量被删除了)
Multiple R-squared:  0.008796,  Adjusted R-squared:  0.008101
F-statistic: 12.65 on 1 and 1426 DF,  p-value: 0.0003869
```

```
trending <- c(1:length(meantemp))*model_linear[[1]][2]

demean_meantemp <- meantemp - trending
plot(demean_meantemp, type ="l", main = "The Plot of De-meaned Mean Average Temperature", ylab
="temperature")
```



The linear fitting results show both time trend and the intercept are statistical significant at p values lesser than 0.001.We de-trend the series according to the coefficient and the given time.

2.3 Unit-Root Testing on Non-Stationarity

In this subsection, I deployed Augmented Dickey Fuller Test to test its unit-root non-stationarity. The null hypothesis is that the series exhibits a unit-root.

```
adf.test(meantemp)
```

```
Warning: p-value smaller than printed p-value
```

```

    Augmented Dickey-Fuller Test

data:  meantemp
Dickey-Fuller = -7.6177, Lag order = 11, p-value = 0.01
alternative hypothesis: stationary
```

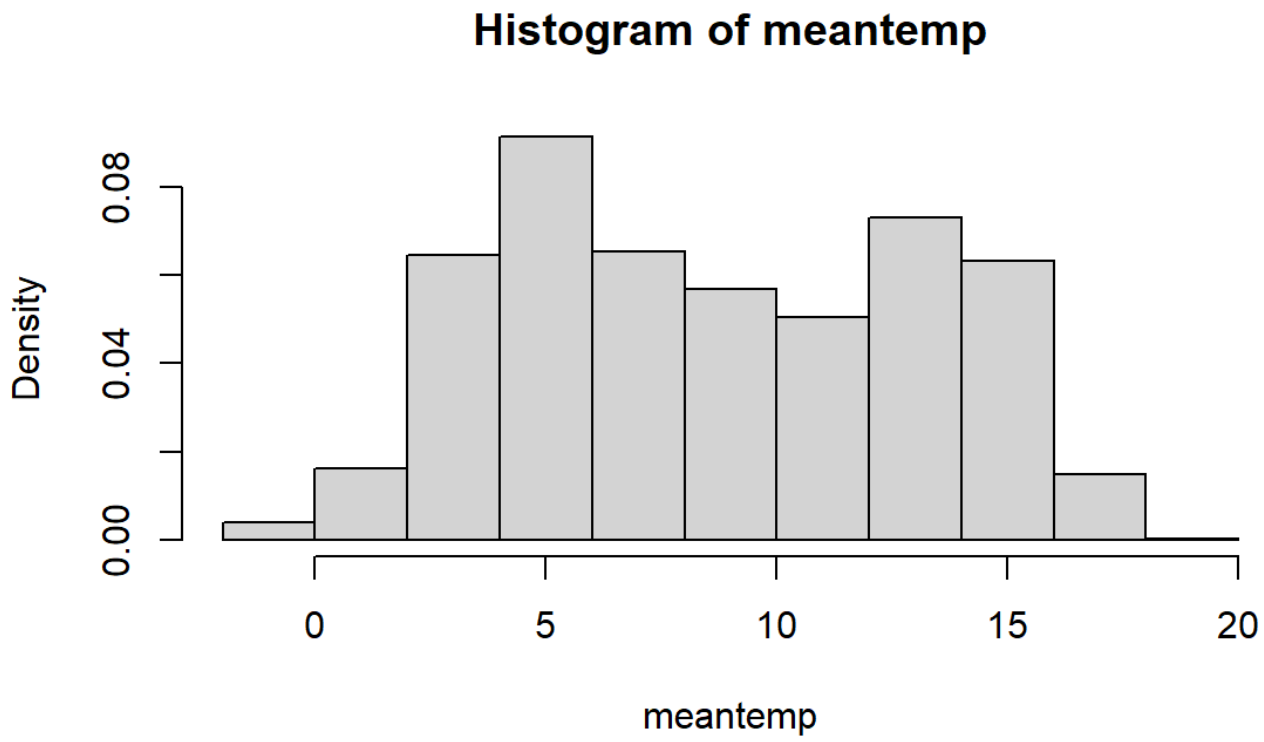
The p-value is 0.01, which I can safely reject the null hypothesis (that this process is a unit-root process).

2.4 Time Series Diagnostics and Statistical Properties

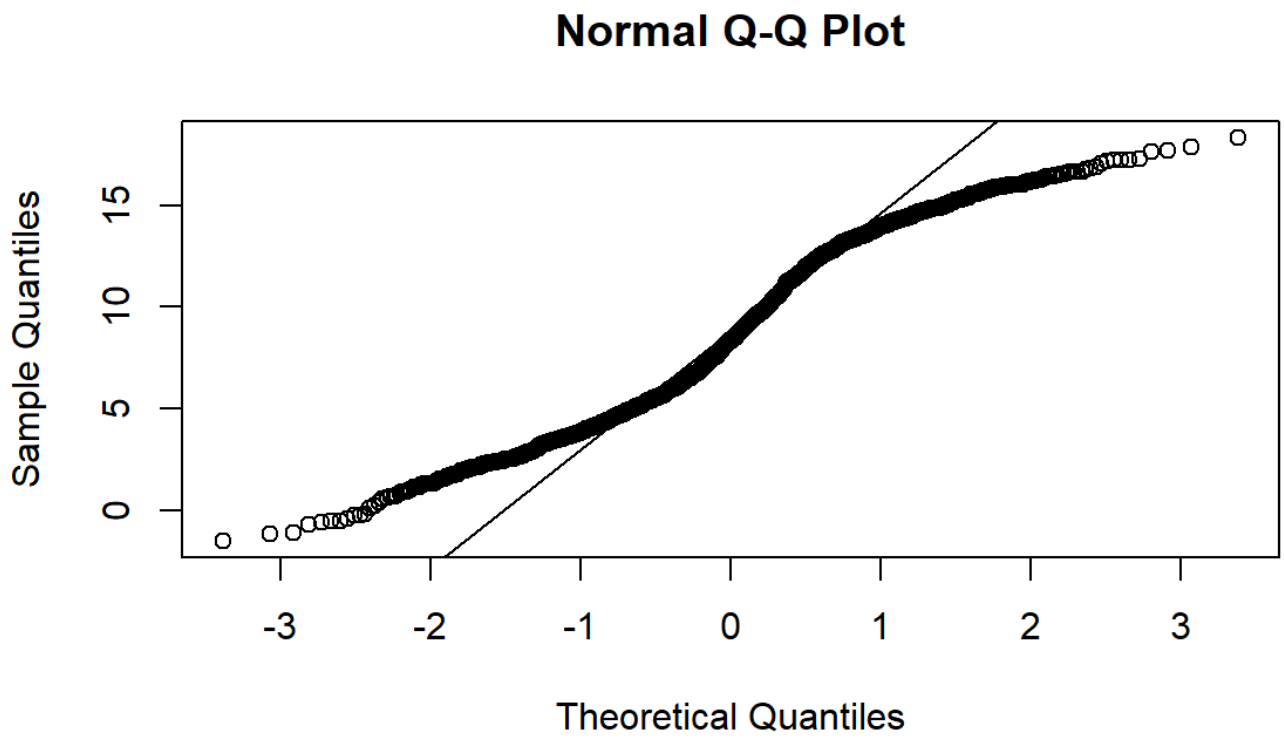
```
normalTest(meantemp,method = "jb") # Cannot reject the normality of R
```

```
Title:
 Jarque - Bera Normalality Test

Test Results:
  STATISTIC:
    X-squared: 82.7423
  P VALUE:
    Asymptotic p Value: < 2.2e-16
```

```
D=hist(meantemp, probability = TRUE)
```




```
qqnorm(meantemp); qqline(meantemp)
```



I deployed Jarque-Bera asymptotic test on the time series to show that the series is not normally distributed. This is rather clear in the histogram plot. In addition, the Quantile-Quartile plot also suggests its non-normality.

3. Model it with ARIMA model

Why I choose this model: ARIMA (Autoregressive Integrated Moving Average) is a popular time series forecasting model that is widely used for forecasting temperature data. Monthly temperature data often exhibits seasonal patterns, where the temperature varies based on the month of the year.Temperature data often shows autocorrelation.The month temperature is correlated with the that in the previous month,and the same month of last year.So the temperature data is seasonality and autocorrelation,which is suitable for ARIMA model to solve this kind of problem.Also I searched on the internet, there are a lot of examples for using ARIMA to forecast temperature.

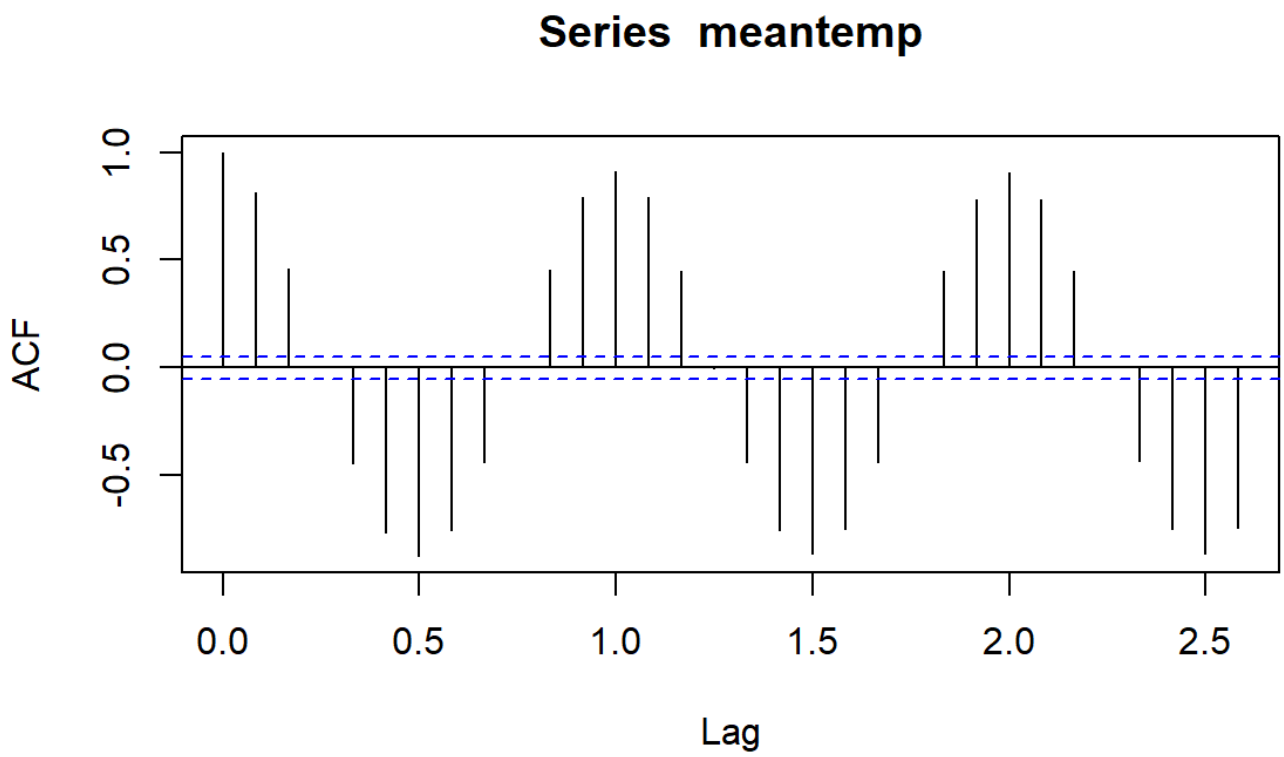
```
# Investigate the statistical dependency of the simple return series.

Box.test(meantemp, lag= log(length(meantemp)), type='Ljung')
```

```
    Box-Ljung test

data:  meantemp
X-squared = 4268, df = 7.2563, p-value < 2.2e-16
```

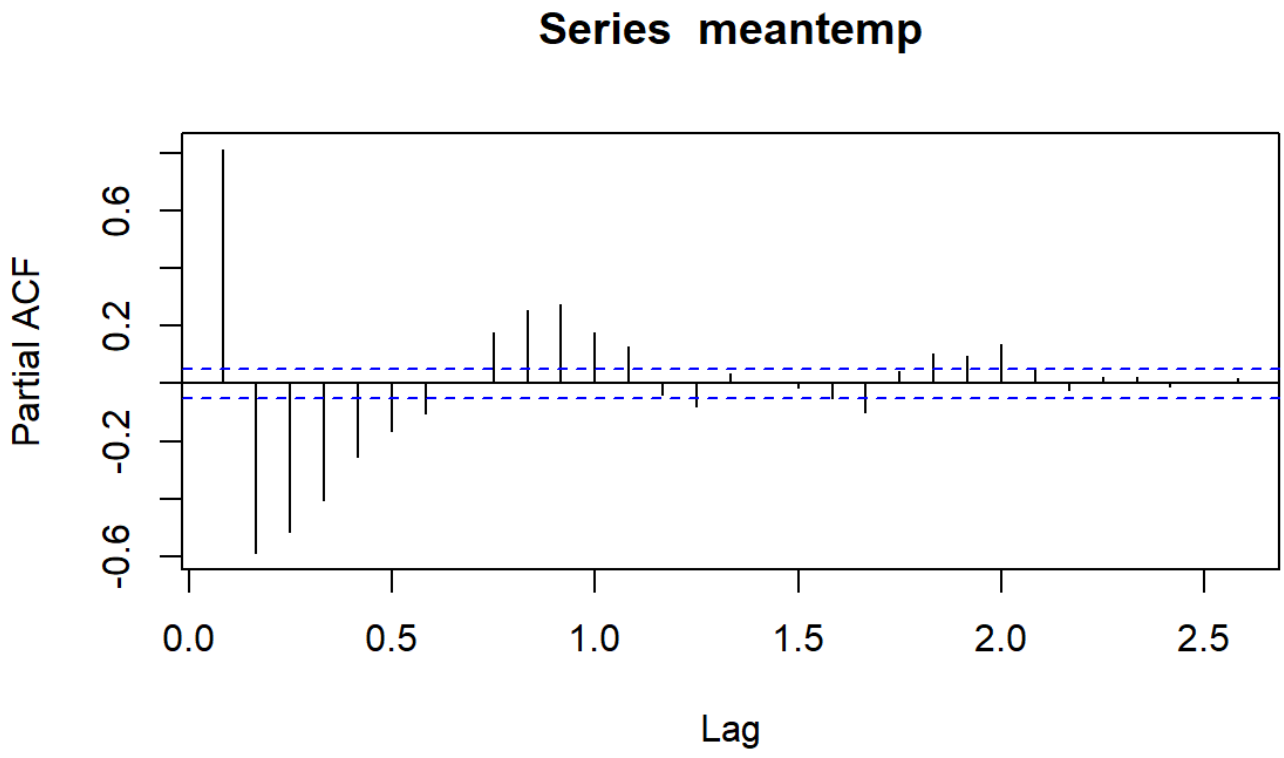
The null hypothesis of Box-Ljung test is that there is no autocorrelation. Our results of the test (p value is lesser 2.2e-16) suggests rejecting null hypothesis. This time series has autocorrelation, and therefore, I can model it using linear models such as ARIMA models.

3.1 Model Choosing Using ACF, PACF

```
acf(meantemp)
```




```
pacf(meantemp)
```



The combination of PACF and ACF suggests the series exhibit seasonality. A seasonal ARIMA model should be fitted to the series.

3.2 Seasonal ARIMA Modelling

```
model1 <- arima(meantemp,order=c(1,0,0),seasonal=list(order=c(1,0,1),period=12)) # by intuition
of weather
model1
```

```
Call:
arima(x = meantemp, order = c(1, 0, 0), seasonal = list(order = c(1, 0, 1),
    period = 12))

Coefficients:
         ar1    sar1     sma1  intercept
      0.2718  0.9998  -0.9434     8.6985
s.e.  0.0263  0.0001   0.0103     1.2929

sigma^2 estimated as 1.576:  log likelihood = -2357.64,  aic = 4725.27
```

```
model2 <- auto.arima(meantemp)
model2
```

```
Series: meantemp
ARIMA(1,0,0)(2,1,0)[12] with drift

Coefficients:
         ar1     sar1     sar2   drift
      0.2545  -0.6898  -0.3259  0.0011
s.e.  0.0259   0.0255   0.0255  0.0021

sigma^2 = 1.991:  log likelihood = -2478.5
AIC=4967.01   AICc=4967.05   BIC=4993.25
```

```
model3 <- arima(meantemp,order=c(0,0,0),seasonal=list(order=c(1,0,0),period=12))
model3
```

```
Call:
arima(x = meantemp, order = c(0, 0, 0), seasonal = list(order = c(1, 0, 0),
    period = 12))

Coefficients:
        sar1  intercept
      0.9215     8.7436
s.e.  0.0100     0.5394

sigma^2 estimated as 3.041:  log likelihood = -2809.99,  aic = 5625.99
```

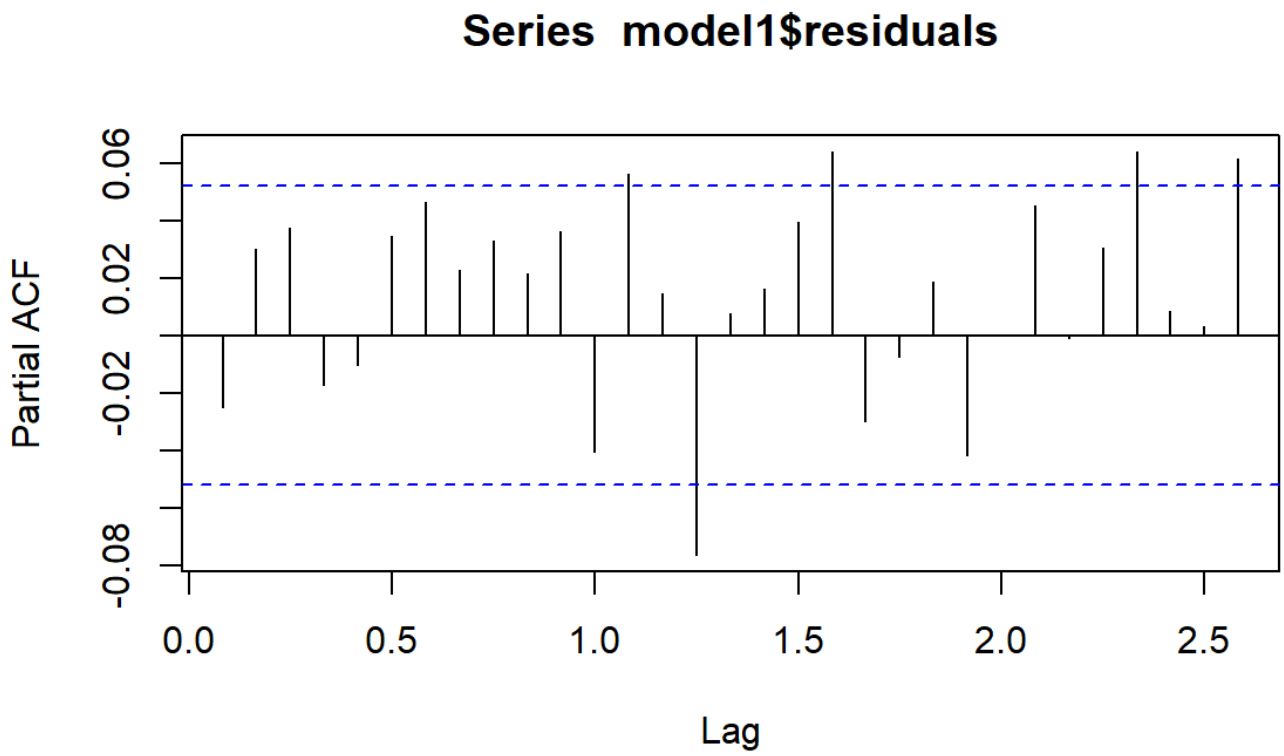
I compare 3 models in ARIMAs. I first begin with the intuition that monthly average temperature is yearly dependent. Hence, it will be conducive to try a seasonal model that gives a 12-month lag. Then, I use Auto ARIMA from the R package to pick a model. I then rely AIC figure to decide which model to go with. The

winning model is model 1 because of the smallest value of AIC.

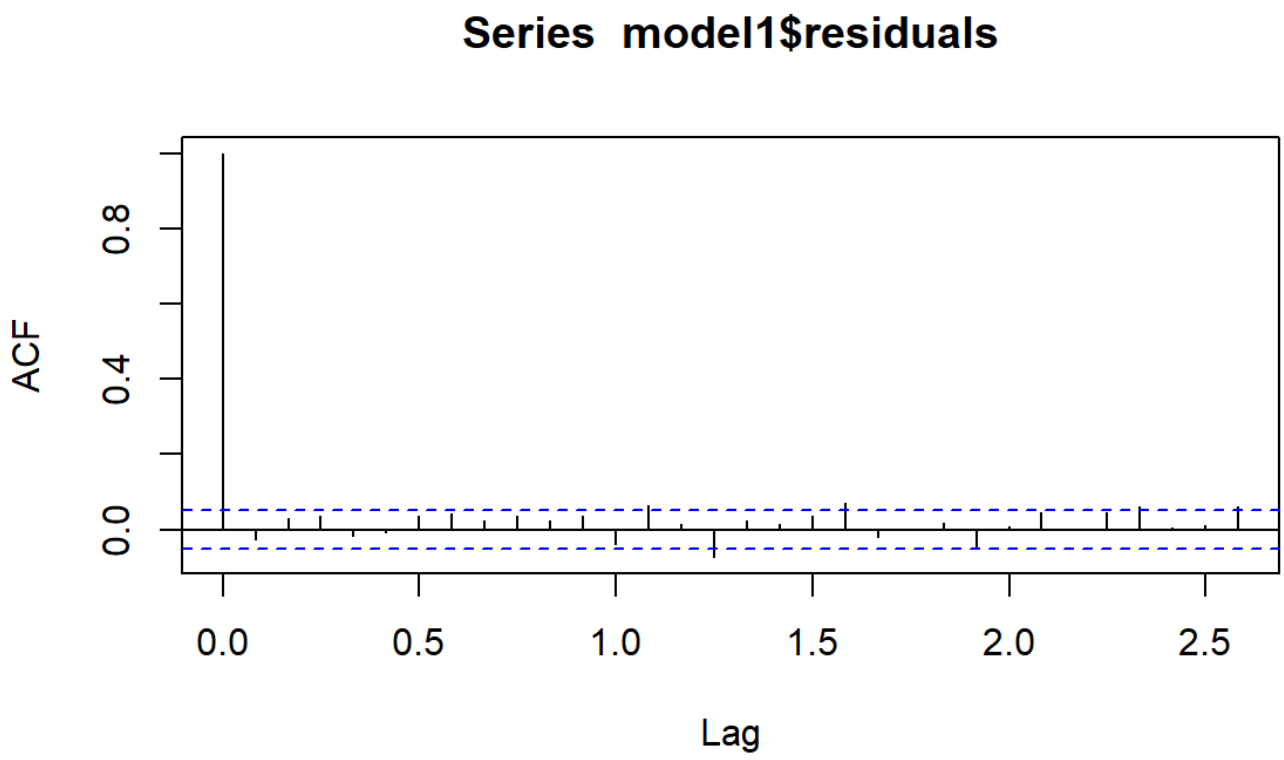
3.3 Validation of the Model

It is crucial to check that the residual series should exhibit no linear dependencies.

```
pacf(model1$residuals)
```




```
acf(model1$residuals)
```




```
Box.test(model1$residuals, lag= 10, type='Ljung')
```

```
    Box-Ljung test

data:  model1$residuals
X-squared = 12.339, df = 10, p-value = 0.263
```

4 Forecast

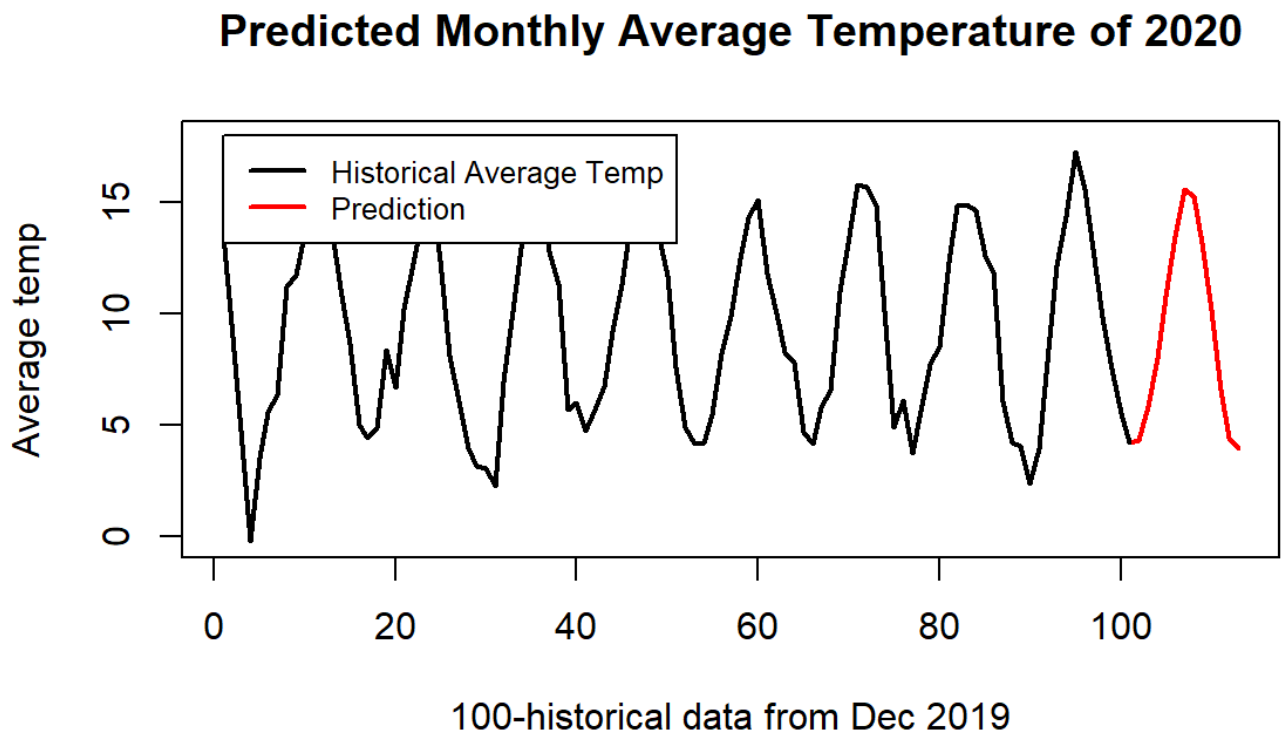
4.1 Out-of-Sample Forecast

```
pred <- predict(arima(meantemp,order=c(1,0,0),seasonal=list(order=c(1,0,1),period=12))  ,n.ahead=12)
as.numeric(pred$pred)
```

```
 [1]  4.309003  5.817022  8.012109 10.784478 13.473062 15.552103
 [7] 15.221649 13.142762 10.135824  6.554738  4.440497  3.924444
```

```
plot(c(meantemp[1317:1417],pred$pred), type="l", lwd=2,col="red", main = 'Predicted Monthly Average Temperature of 2020', xlab="100-historical data from Dec 2019", ylab="Average temp")
lines( meantemp[1317:1417], lwd=2)
```

```
legend(1, 18, legend=c("Historical Average Temp", "Prediction"),
       col=c("black", "red"), lty=c(1,1), lwd=2 ,cex=0.8)
```

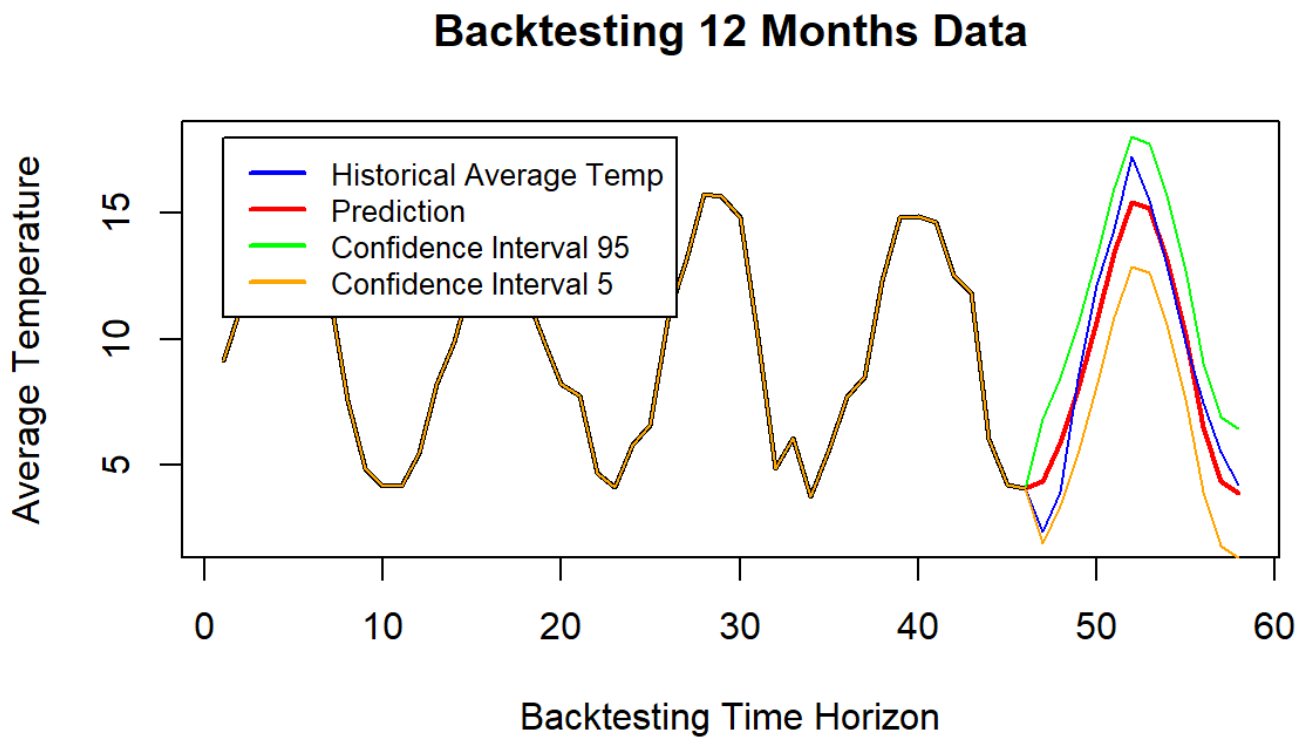


4.2 Backtesting

```
pred_train <- predict(arima(meantemp[1:as.numeric(length(meantemp)-12)],order=c(1,0,0),seasonal
=list(order=c(1,0,1),period=12))  ,n.ahead=12)
# construct the plot
plot(c(meantemp[1360:1405],pred_train$pred), type="l", lwd=2,col="red", ylim=c(2,18) ,  main=
"Backtesting 12 Months Data", ylab="Average Temperature", xlab="Backtesting Time Horizon")
lines(meantemp[1360:1405],lwd=2)
```

```
lines(c(meantemp[1360:1405],meantemp[1406:1417]), lwd=1,col="blue")

lines(c( meantemp[1360:1405], as.numeric(pred_train$pred+1.96*pred_train$se)),col='green', lwd=
1, type = "l")
```

```
lines(c( meantemp[1360:1405], as.numeric(pred_train$pred-1.96*pred_train$se)),col='orange', lwd
=1, type = "l")
legend(1, 18, legend=c("Historical Average Temp", "Prediction", "Confidence Interval 95", "Conf
idence Interval 5" ),
       col=c("blue", "red", "green","orange"), lty=c(1,1), lwd=2 ,cex=0.8)
```



It is important to note that the AIC does not tell us whether a model is perfect or not, but only provides a guideline for selecting the best model. So I do a backtest to check the result of the prediction. We can see that the prediction line is in the range of confidence interval and it is also similar to the historical value.