TIME SERIES ANALYSIS ON CLIMATE CHANGE IN INDIA

Shrikar Jayaraman and Ananya Singh

19-02-2023

All inferences provided are for the graphs above them

Libraries

```
library(tidyverse)
```

```
## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.2 ──
## ✔ ggplot2 3.4.0      ✔ purrr   1.0.1
## ✔ tibble  3.1.8      ✔ dplyr   1.0.10
## ✔ tidyr   1.2.1      ✔ stringr 1.5.0
## ✔ readr   2.1.3      ✔ forcats 0.5.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
library(ggplot2)
library(grid)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
library(RColorBrewer)
library(ggthemes)
library(dplyr)
library(sp)
library(rworldmap)
```

```
## ### Welcome to rworldmap ###
## For a short introduction type :   vignette('rworldmap')
```

```
library(fpp2)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
## ── Attaching packages ──────────────────────────────────────────────── fpp2 2.5 ──
## ✔ forecast  8.21     ✔ expsmooth 2.3
## ✔ fma       2.5
```

```
library(tseries)
library(prophet)
```

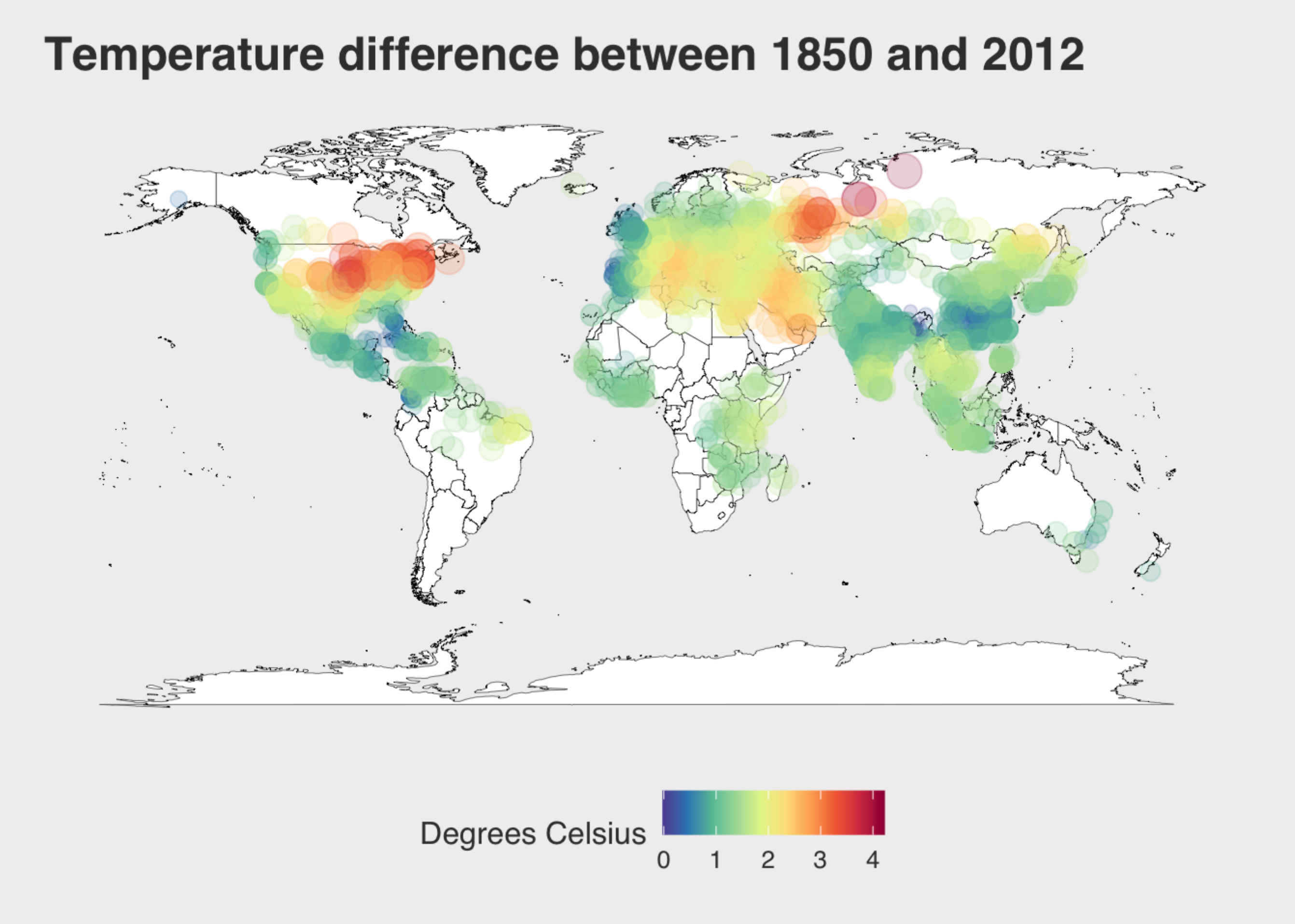
```
## Loading required package: Rcpp
## Loading required package: rlang
##
## Attaching package: 'rlang'
##
## The following objects are masked from 'package:purrr':
##
##     %@%, flatten, flatten_chr, flatten_dbl, flatten_int, flatten_lgl,
##     flatten_raw, invoke, splice
```

Global Temperatures of countries world map between 1850 - 2012

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## ℹ Please use `linewidth` instead.
```

```
## Warning in geom_map(data = worldMap, map = worldMap, aes(x = long, y = lat, :
## Ignoring unknown aesthetics: x and y
```

```
## `summarise()` has grouped output by 'Country'. You can override using the
## `.groups` argument.
## Adding missing grouping variables: `Country`
## `summarise()` has grouped output by 'Country'. You can override using the
## `.groups` argument.
```



Inference

- The darker the color of a country, the more variation in temperature was recorded. Eastern countries of US show red color indicating heavy change in temperature but India has not shown any drastic change when the temperatures were compared.

Functions

```
rm(list = ls())

mean_median = function(df){
  df %>%
    summarize(avg = mean(df$AverageTemperature,na.rm = T), md = median(df$AverageTemp
erature, na.rm= T))
} ## prints mean and median

check_na = function(df,col){
  print(paste("Col name: ", col))
  sum(is.na(df[col]))
}## checks if any na value exists
```

Data pre-processing

```
##           dt AverageTemperature AverageTemperatureUncertainty Country
## 1 1743-11-01              4.384                         2.294   Åland
## 2 1743-12-01                 NA                            NA   Åland
## 3 1744-01-01                 NA                            NA   Åland
## 4 1744-02-01                 NA                            NA   Åland
## 5 1744-03-01                 NA                            NA   Åland
## 6 1744-04-01              1.530                         4.680   Åland
```

```
##       Year          Month             Day    AverageTemperature
##  Min.   :1796   Min.   : 1.000   Min.   :1   Min.   :14.38     
##  1st Qu.:1850   1st Qu.: 3.000   1st Qu.:1   1st Qu.:19.86     
##  Median :1904   Median : 6.000   Median :1   Median :25.02     
##  Mean   :1904   Mean   : 6.495   Mean   :1   Mean   :23.87     
##  3rd Qu.:1959   3rd Qu.: 9.000   3rd Qu.:1   3rd Qu.:27.13     
##  Max.   :2013   Max.   :12.000   Max.   :1   Max.   :31.33     
##                                              NA's   :105       
##  AverageTemperatureUncertainty   Country         
##  Min.   :0.0670                Length:2613       
##  1st Qu.:0.2240                Class :character  
##  Median :0.3660                Mode  :character  
##  Mean   :0.7583                                  
##  3rd Qu.:1.2090                                  
##  Max.   :6.0890                                  
##  NA's   :105
```

```
## 'data.frame':    2613 obs. of  6 variables:
##  $ Year                         : int  1796 1796 1796 1796 1796 1796 1796 1796 179
6 1796 ...
##  $ Month                        : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Day                          : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ AverageTemperature           : num  17 19.2 22.3 27.2 30 ...
##  $ AverageTemperatureUncertainty: num  2.04 1.36 2.12 1.51 1.34 ...
##  $ Country                      : chr  "India" "India" "India" "India" ...
```

```
## [1] "Col name:  AverageTemperature"
```

```
## [1] 105
```

```
## [1] "Col name:  Year"
```

```
## [1] 0
```

```
##        avg      md
## 1 23.87379 25.0185
```

```
## [1] "Col name:  AverageTemperature"
```

```
## [1] 34
```

```
## [1] "Col name:  Year"
```

```
## [1] 0
```

```
## [1] "Col name:  Month"
```

```
## [1] 0
```

```
##        avg     md
## 1 23.99976 25.318
```

Here the column dt is seperated into year, month and date and the rows with na values in Average Temperature Column are imputated with median

```
## 'data.frame':    1965 obs. of  6 variables:
##  $ Year                         : int  1850 1850 1850 1850 1850 1850 1850 1850 185
0 1850 ...
##  $ Month                        : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ Day                          : int  1 1 1 1 1 1 1 1 1 1 ...
##  $ AverageTemperature           : num  15.9 18.7 23 26.7 28.6 ...
##  $ AverageTemperatureUncertainty: num  1.358 1.758 2.893 1.297 0.867 ...
##  $ Country                      : chr  "India" "India" "India" "India" ...
```

```
## [1] "Col name:  AverageTemperature"
```

```
## [1] 0
```

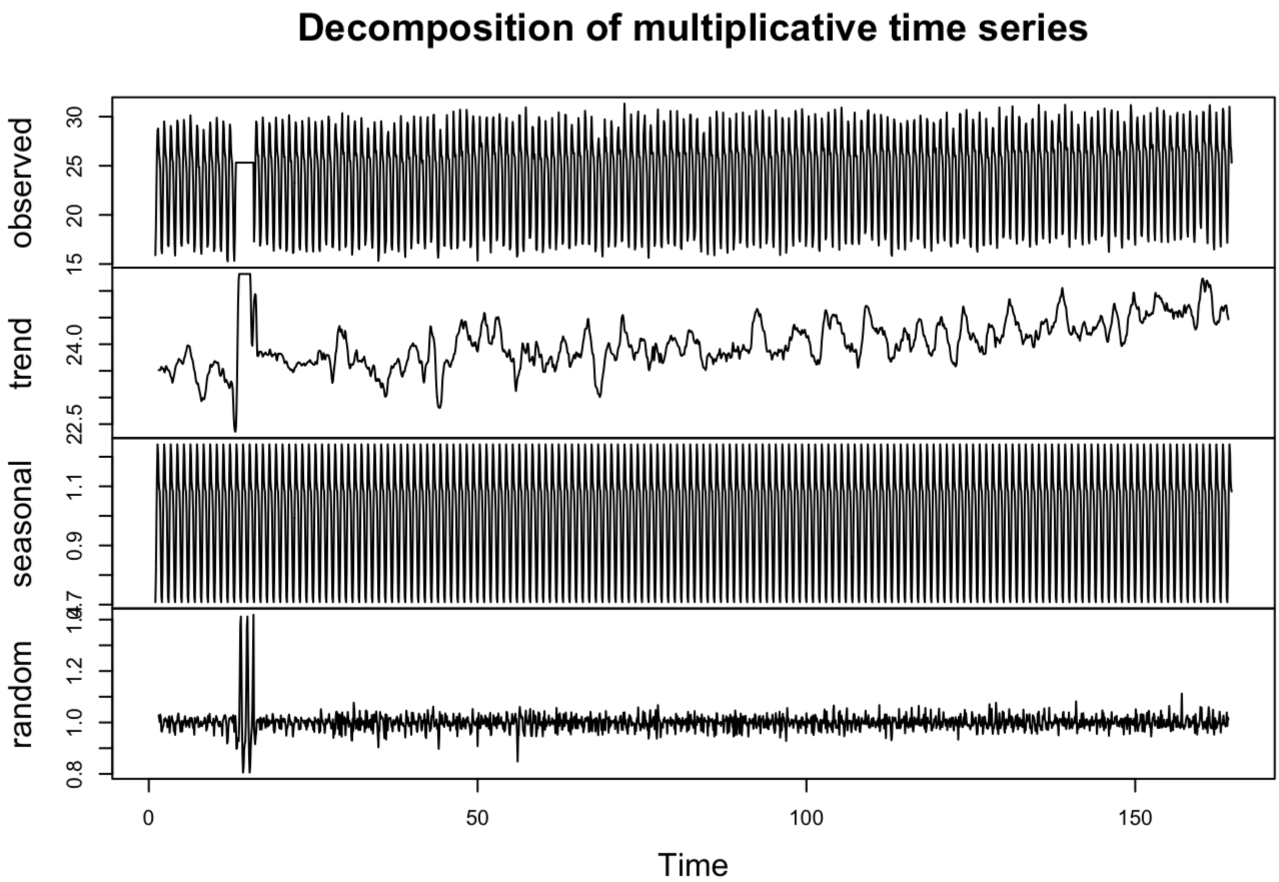
```
## [1] "Col name:  Year"
```

```
## [1] 0
```

```
## [1] "Max avg temp:  31.329"
```

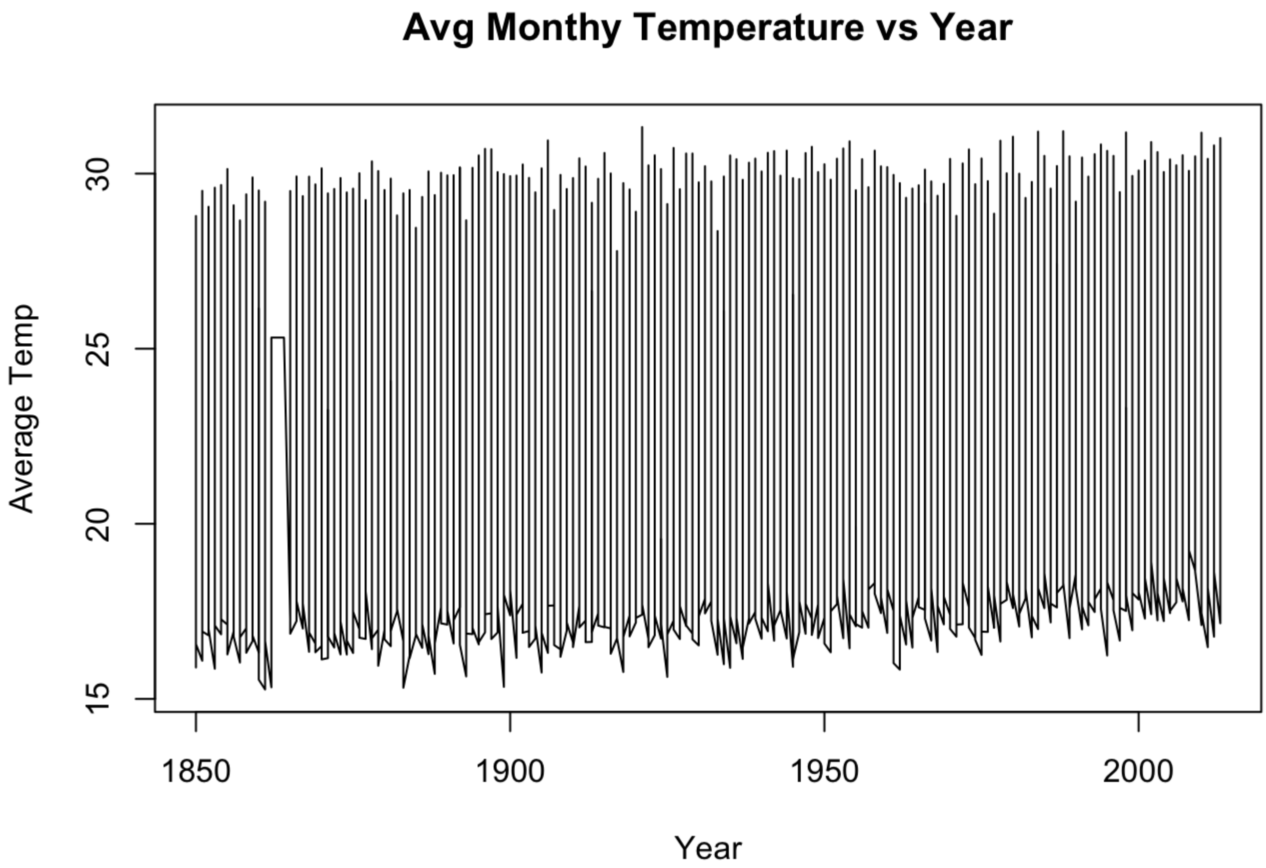
```
## [1] "Min avg temp:  15.27"
```

Plots and Analysis



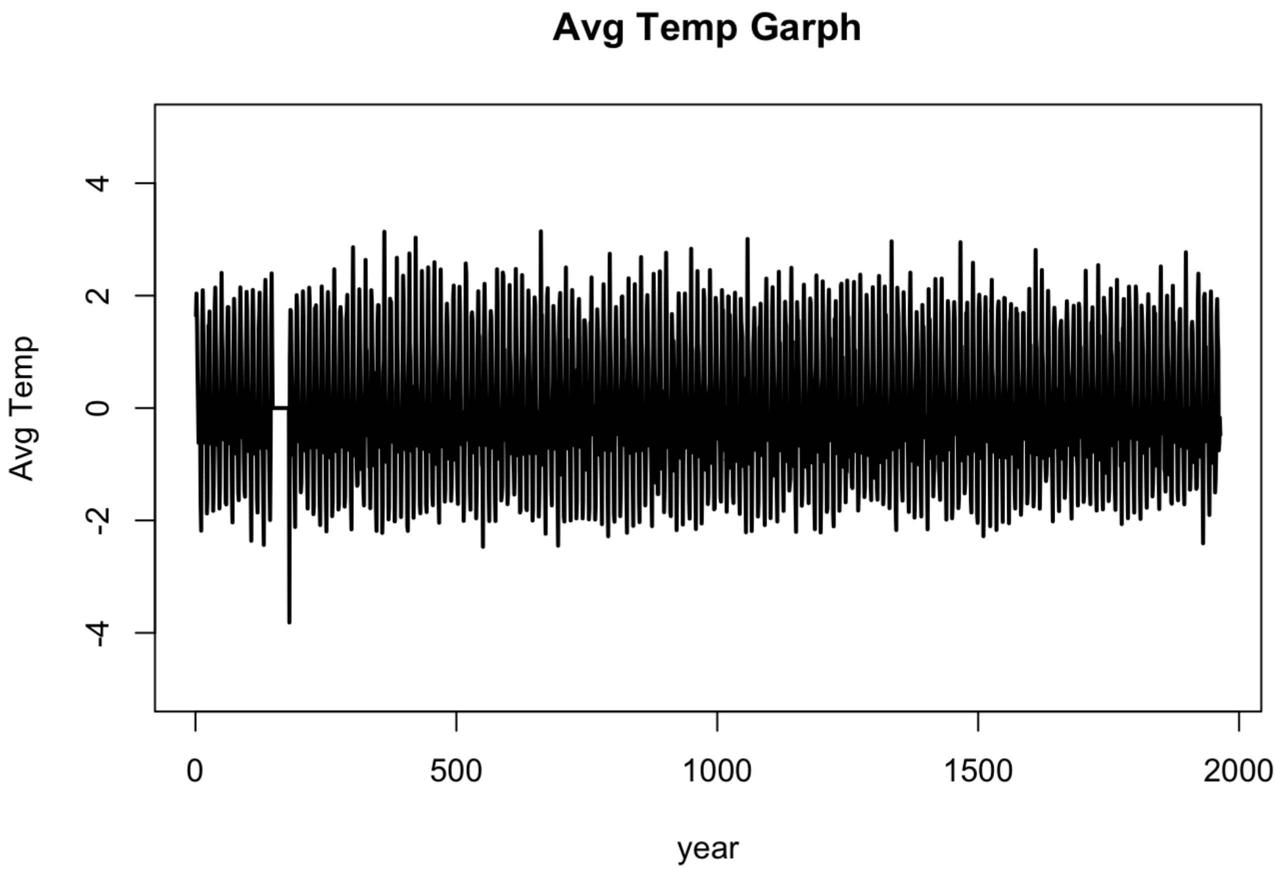
Here we can see the trend clearly but seasonality is not visible properly hence further investigation in that is required. Trend

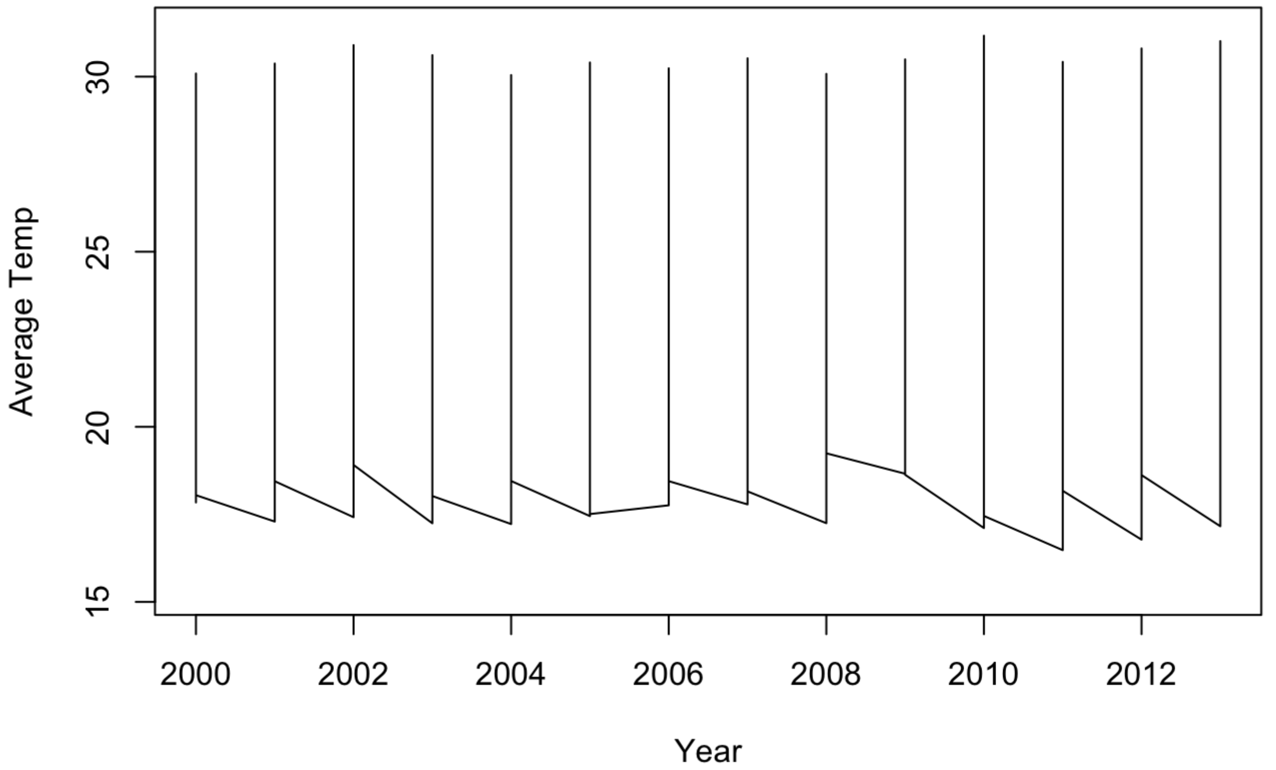
shows that the temperature is increasing



Here all the datapoints are ploted hence no seasonality or

trend is observed.

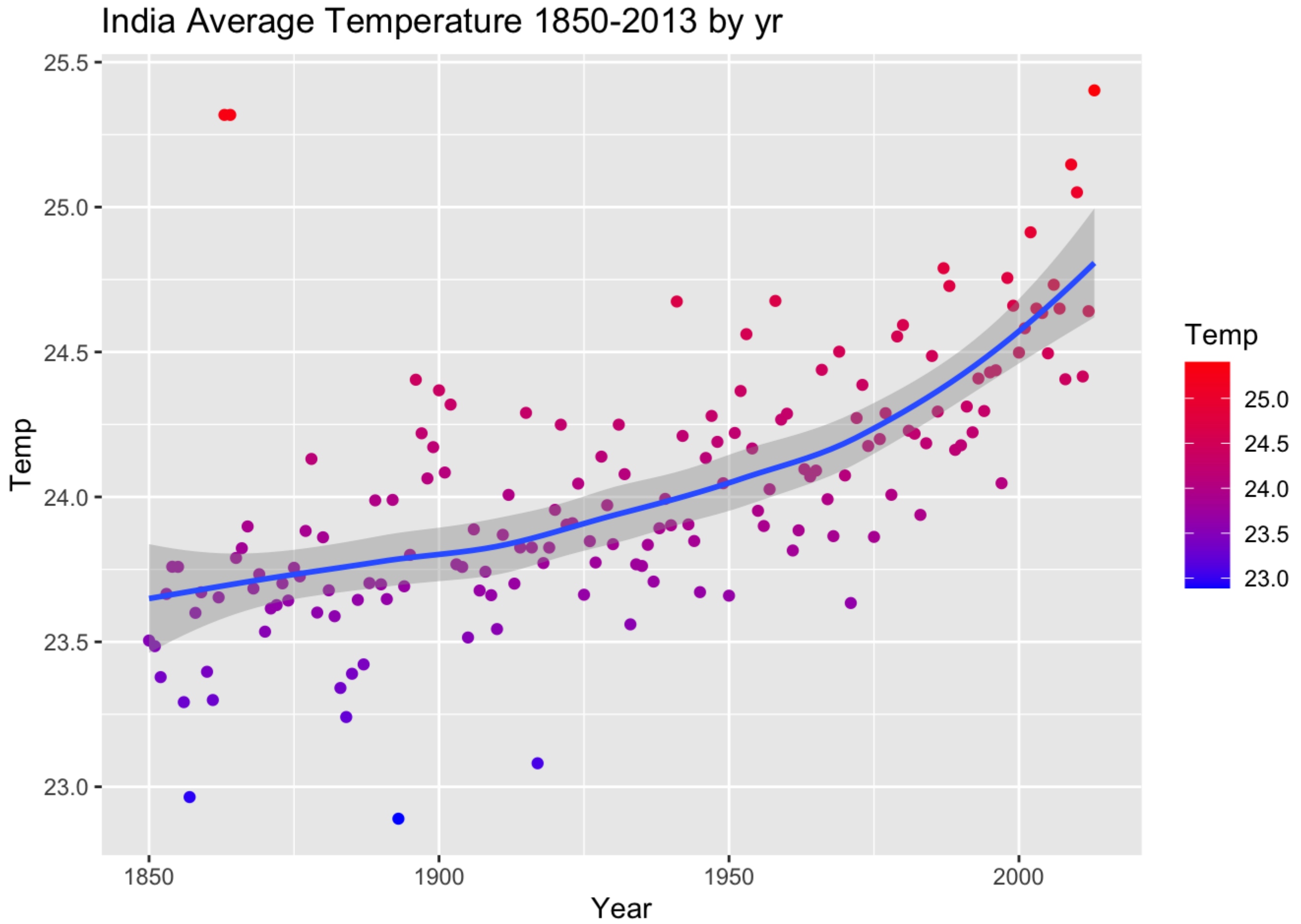





```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
```

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

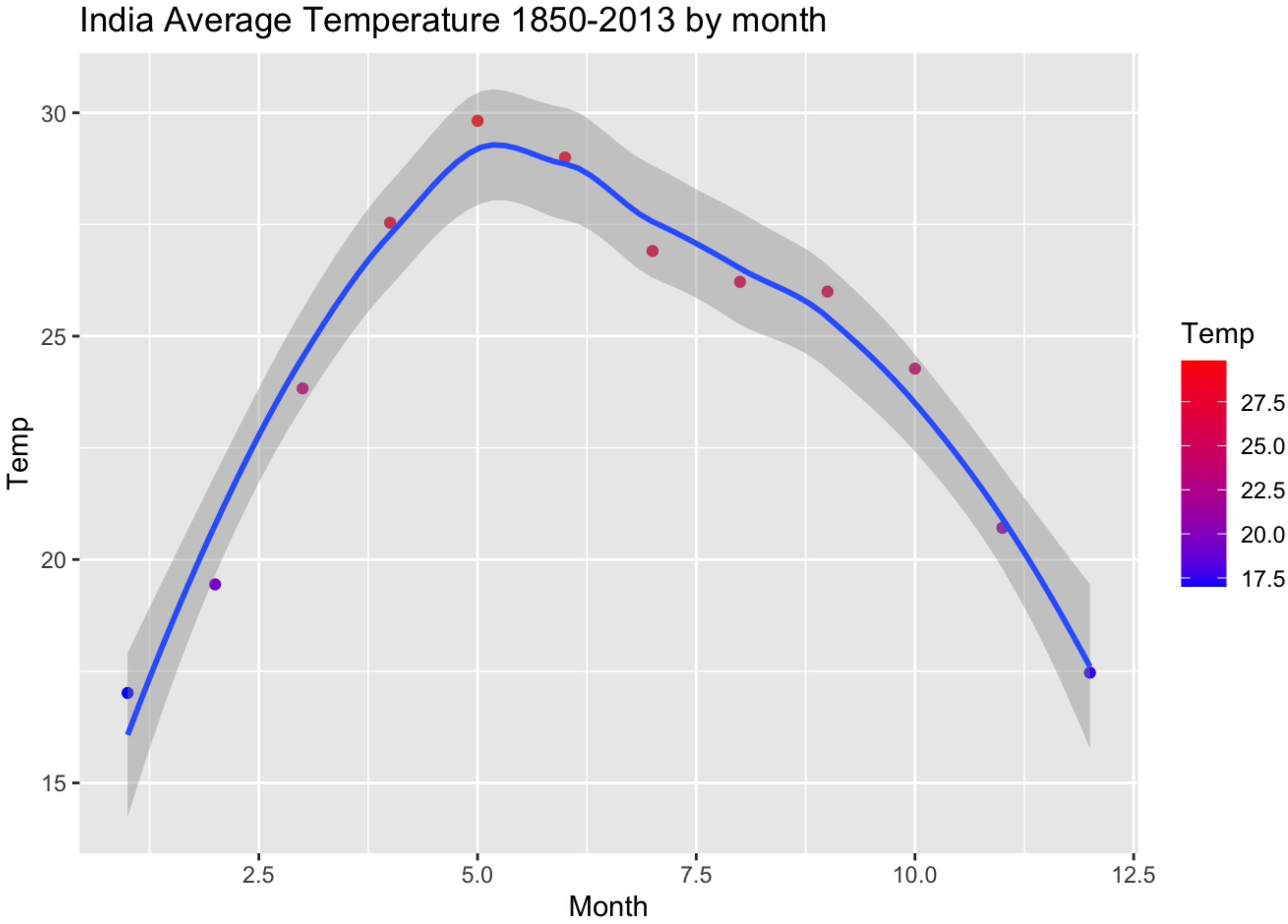
```
## Warning: The following aesthetics were dropped during statistical transformation:
colour
## ℹ This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## ℹ Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
```



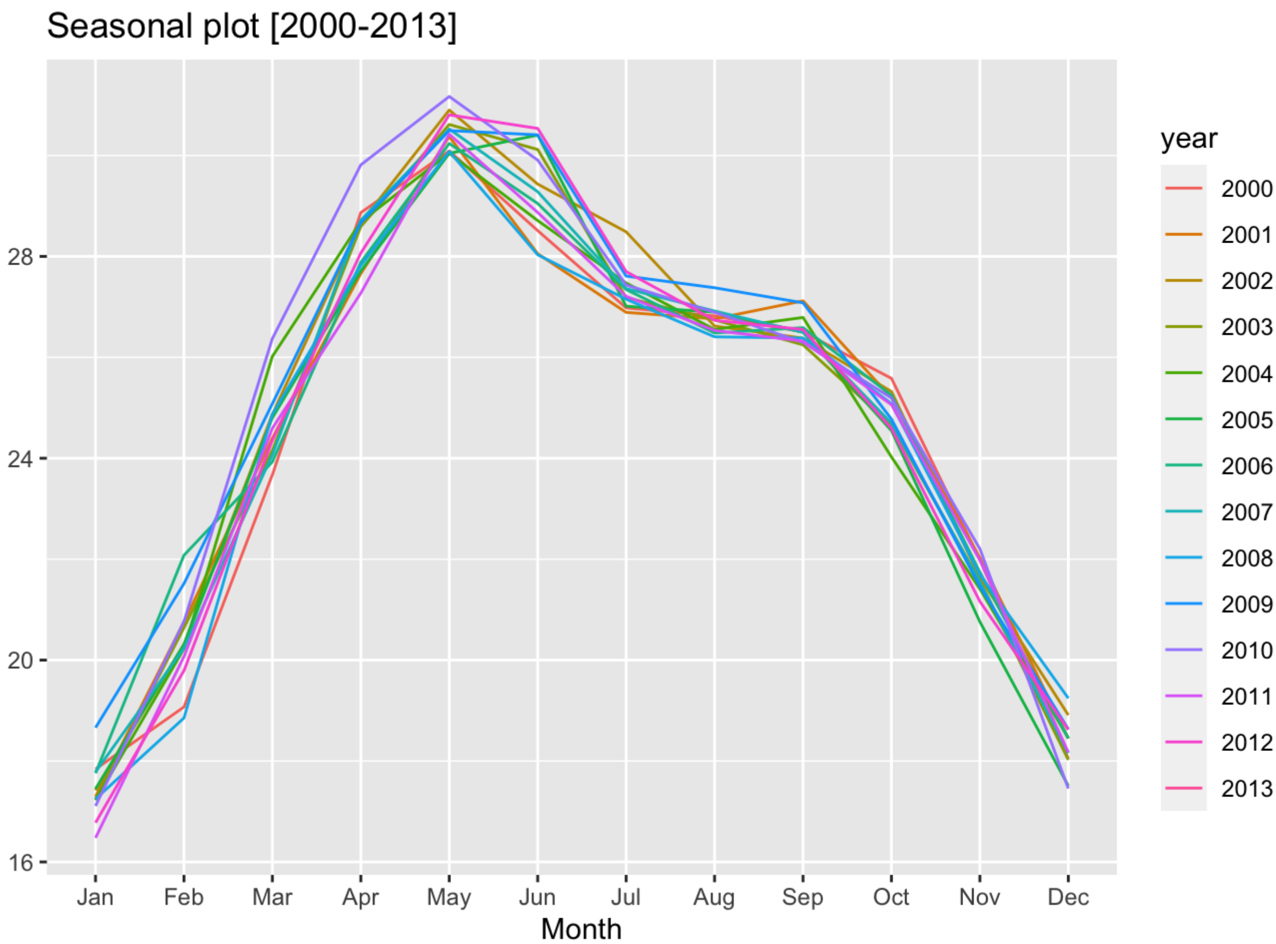
Here the avg tempearture of all the years are plotted and the trends becomes clearly visible.

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

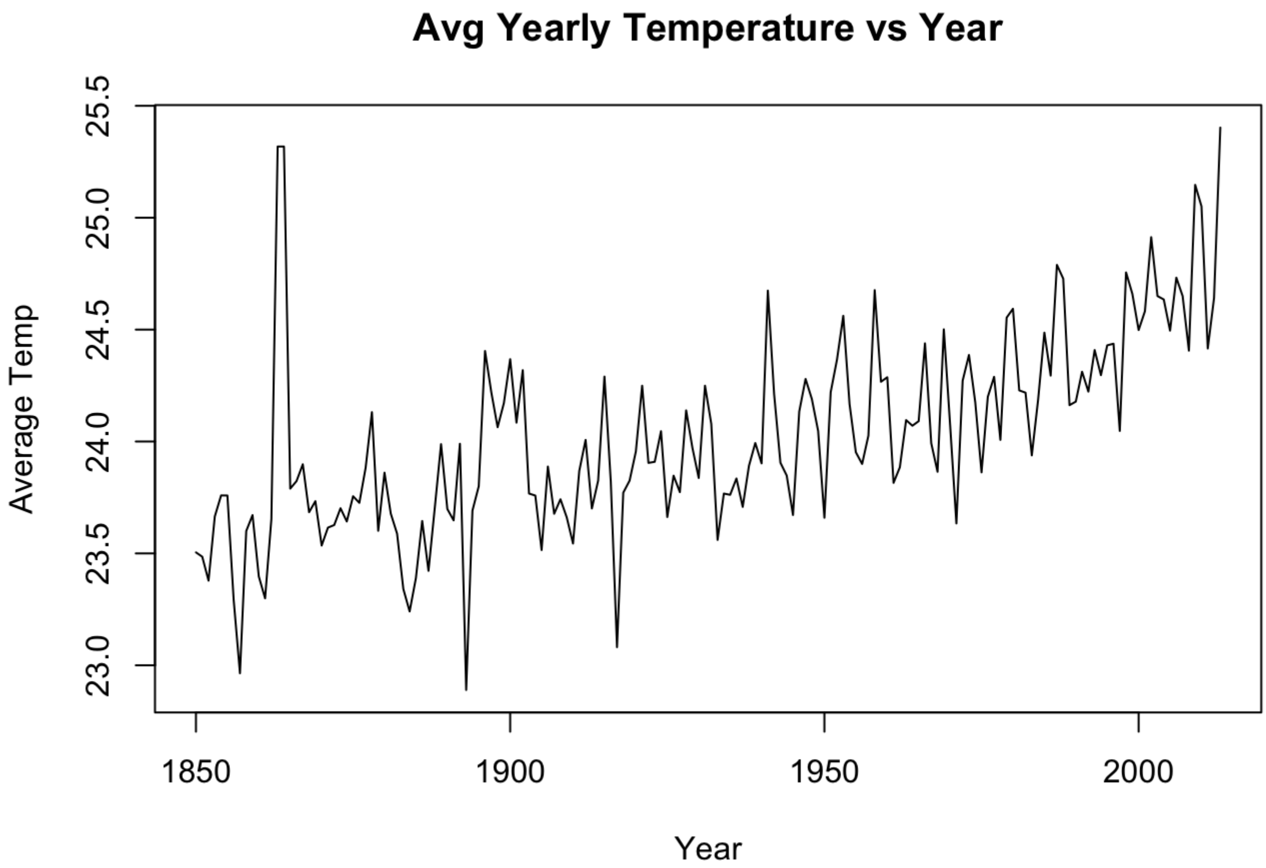
```
## Warning: The following aesthetics were dropped during statistical transformation:
colour
## ℹ This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## ℹ Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
```



Monthly seasonality where temp increases till May which is its peek and then decreases

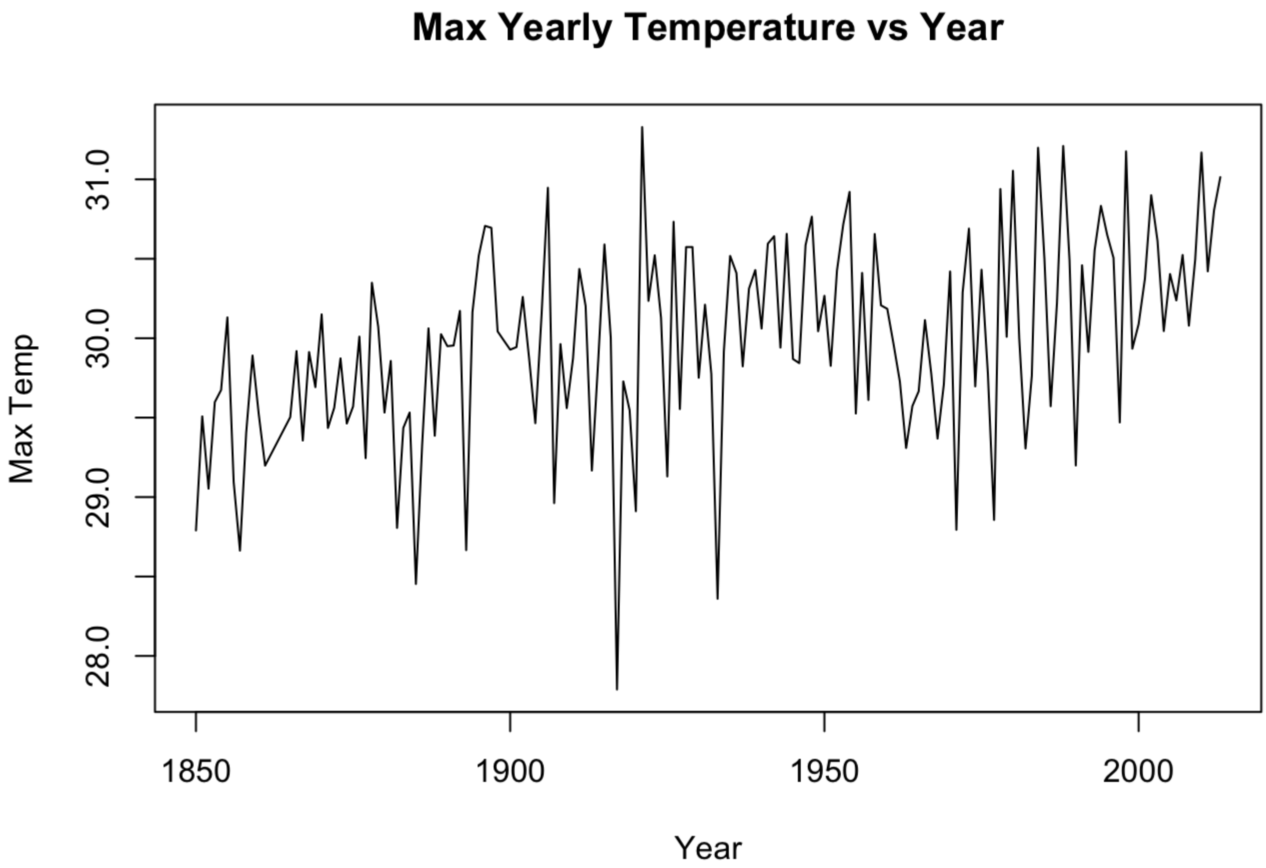


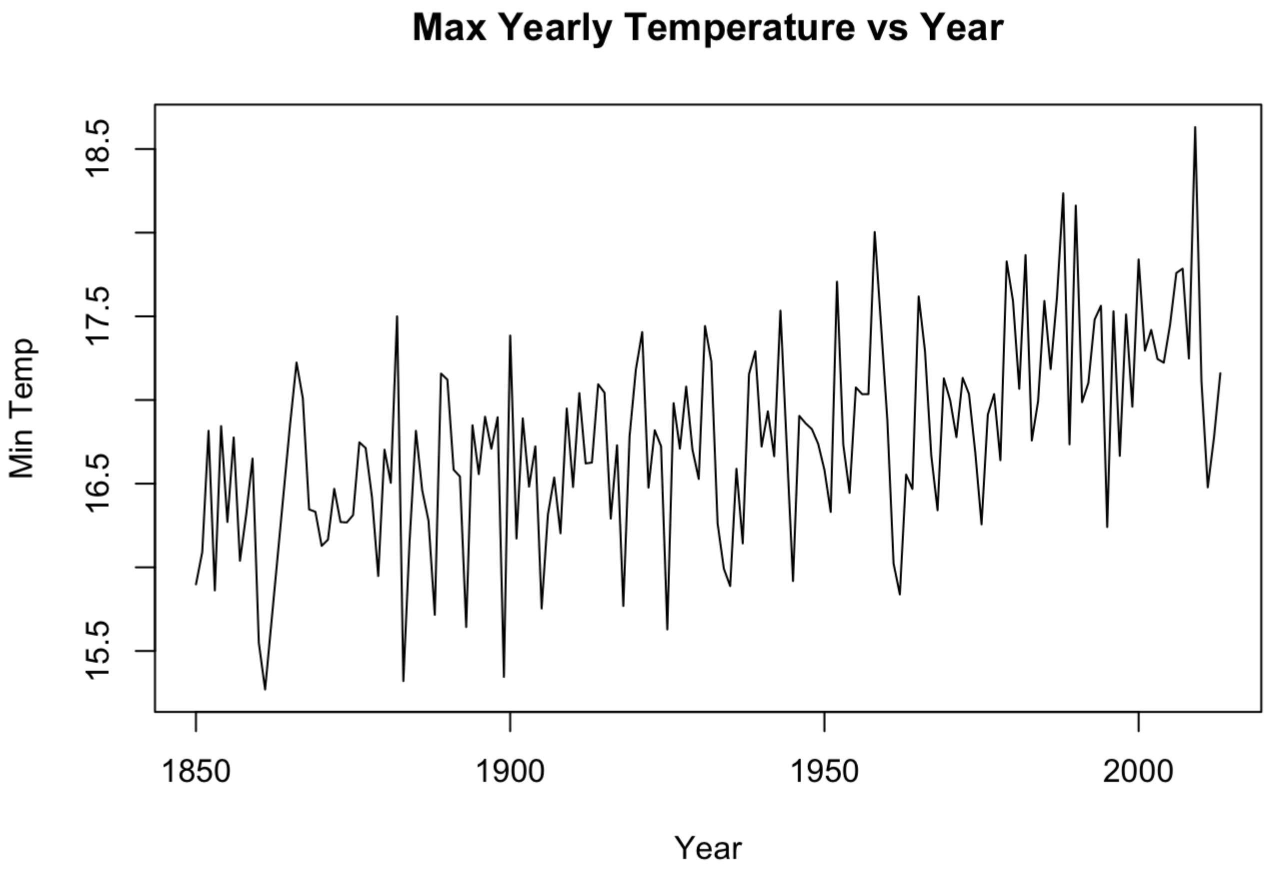
Seasonality remains same for all years from 2000 to 2013 where temp increases till may and then decreases



Here the avg tempearture of all the years are plotted and the

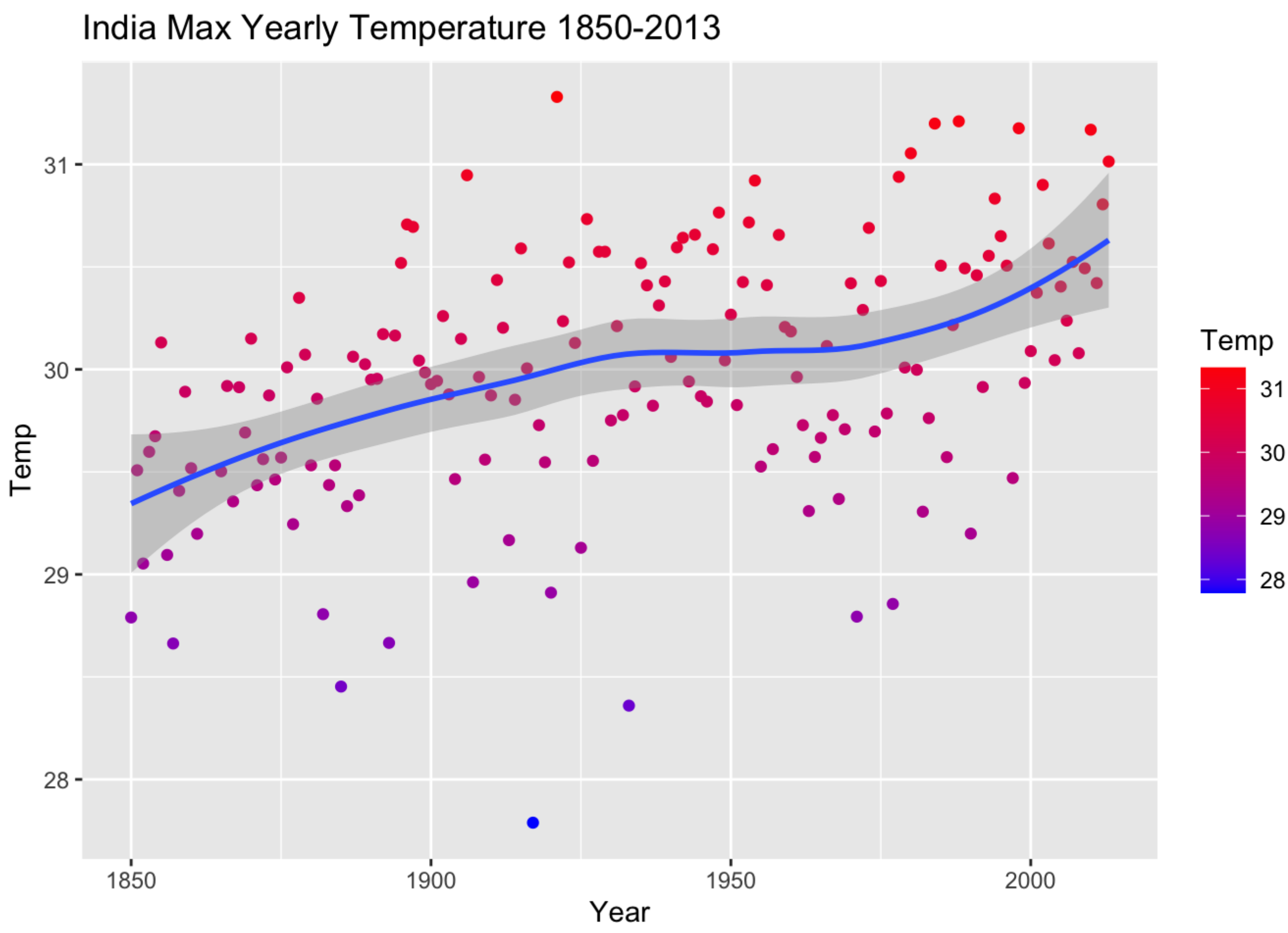
trends becomes clearly visible which is increasing trend.





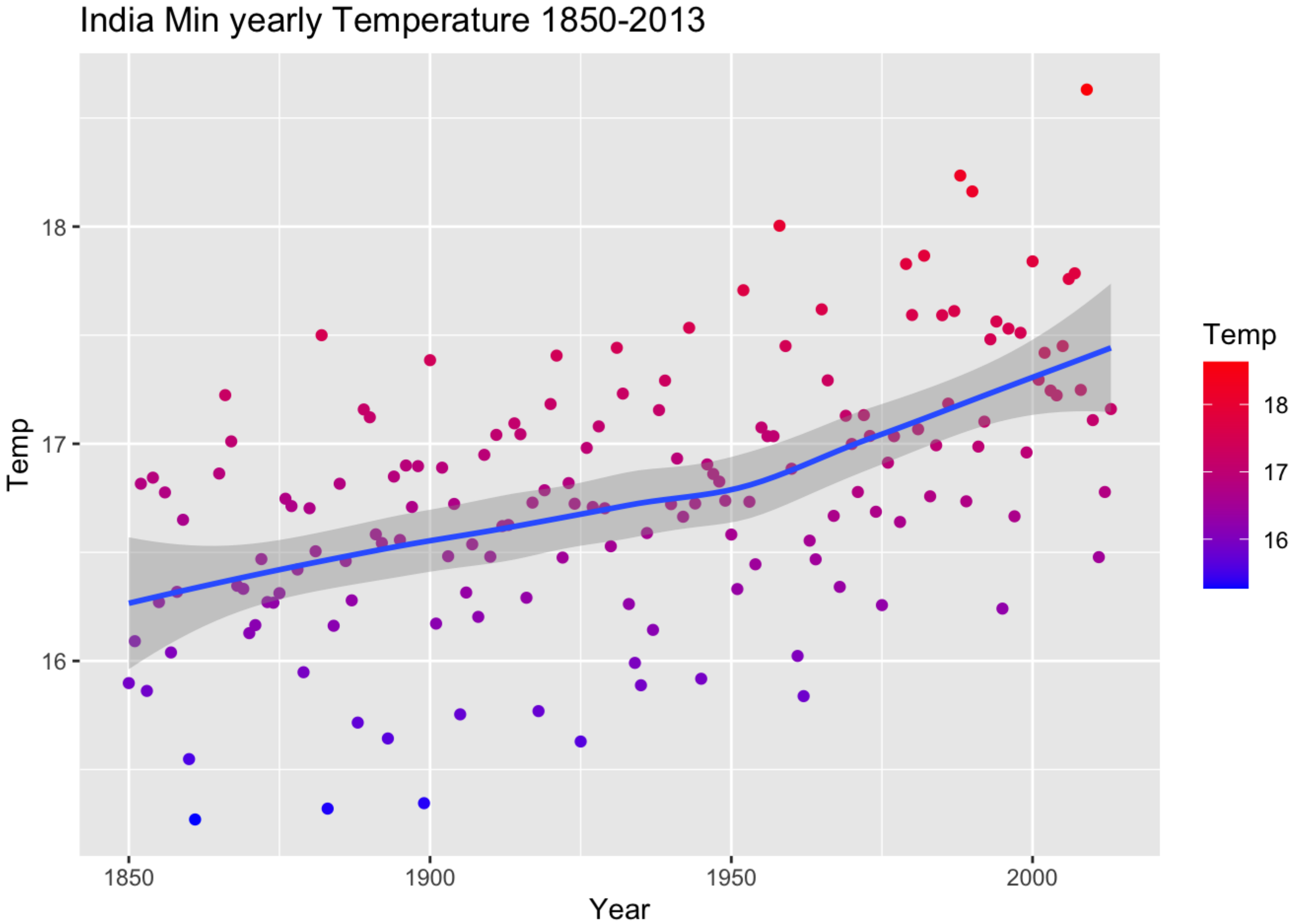

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

```
## Warning: The following aesthetics were dropped during statistical transformation:
colour
## ℹ This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## ℹ Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
```

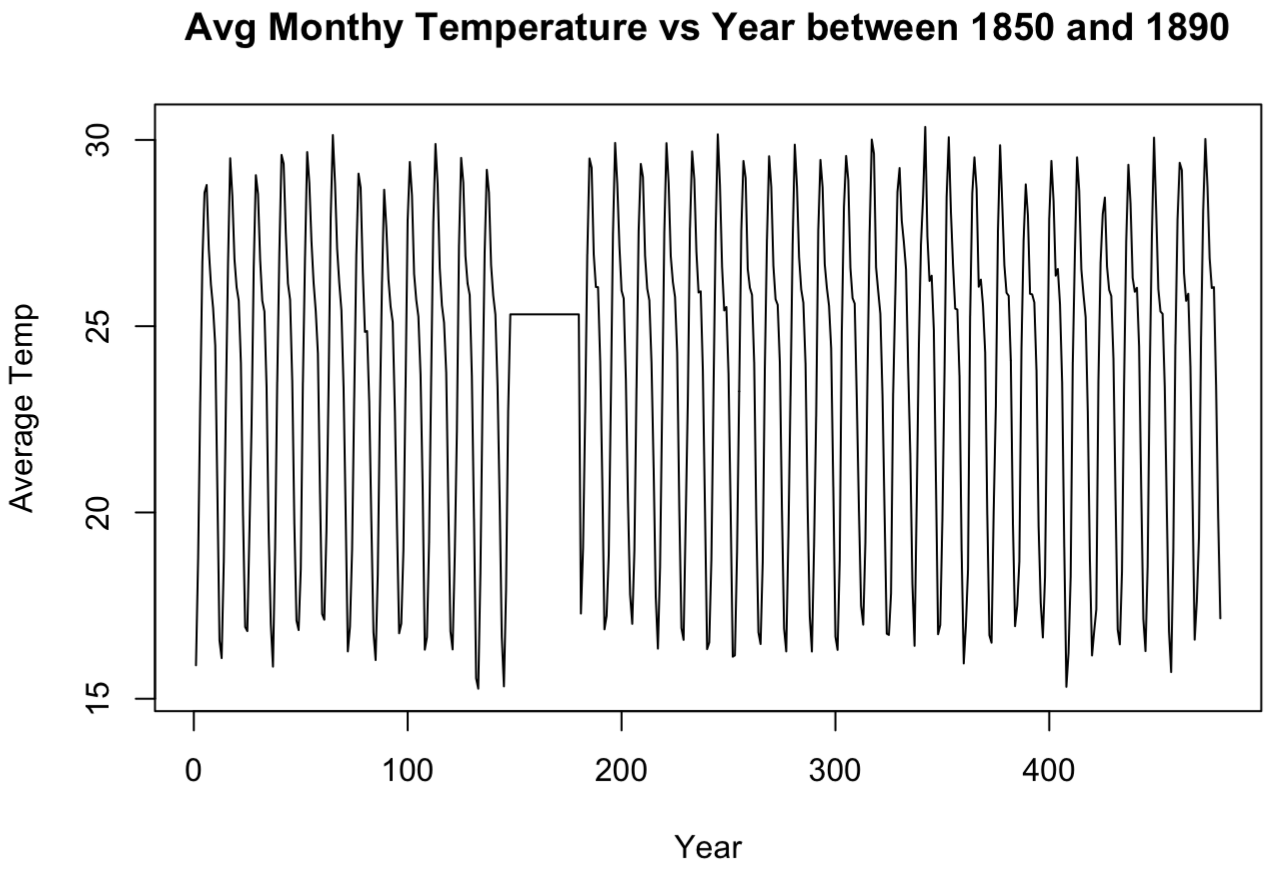


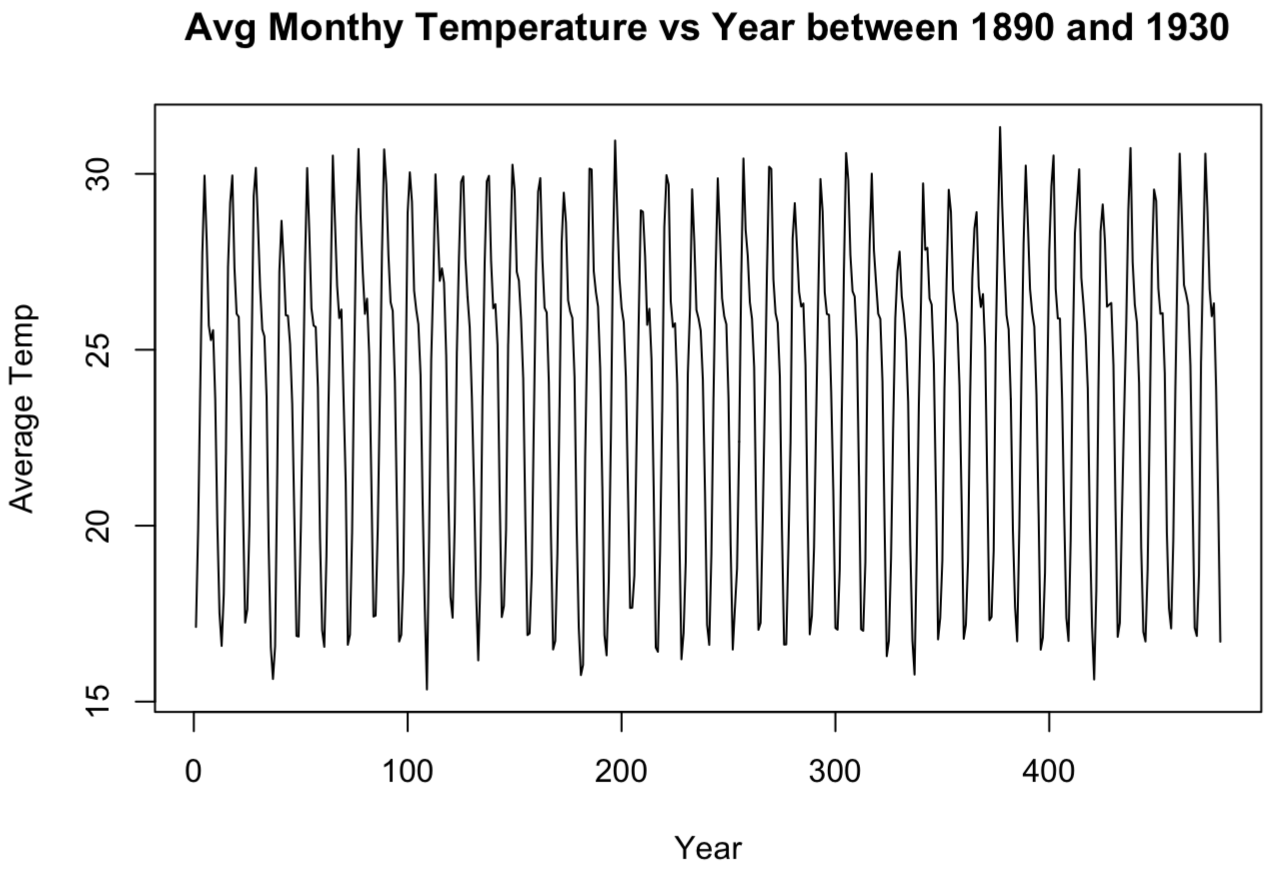

```
## `geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```

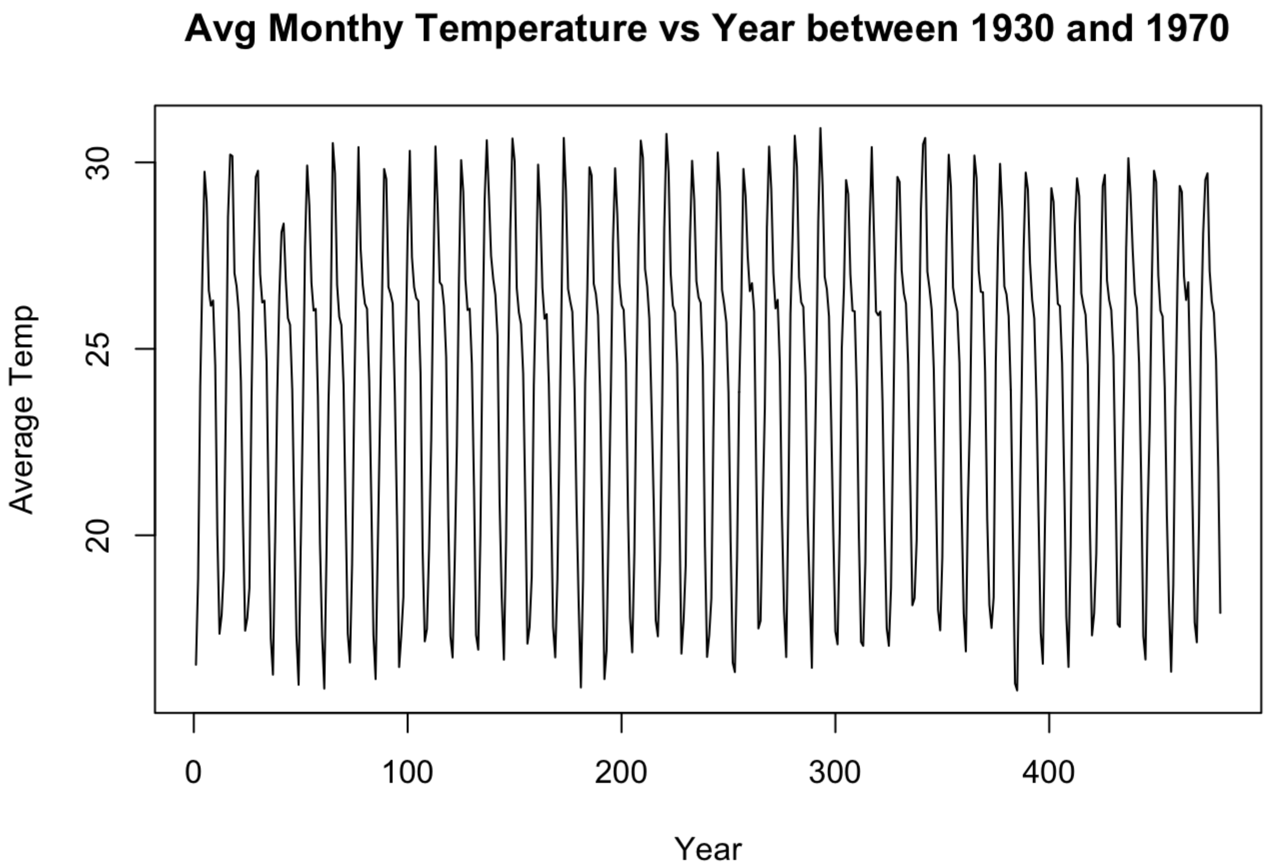
```
## Warning: The following aesthetics were dropped during statistical transformation:
colour
## ℹ This can happen when ggplot fails to infer the correct grouping structure in
##   the data.
## ℹ Did you forget to specify a `group` aesthetic or to convert a numerical
##   variable into a factor?
```

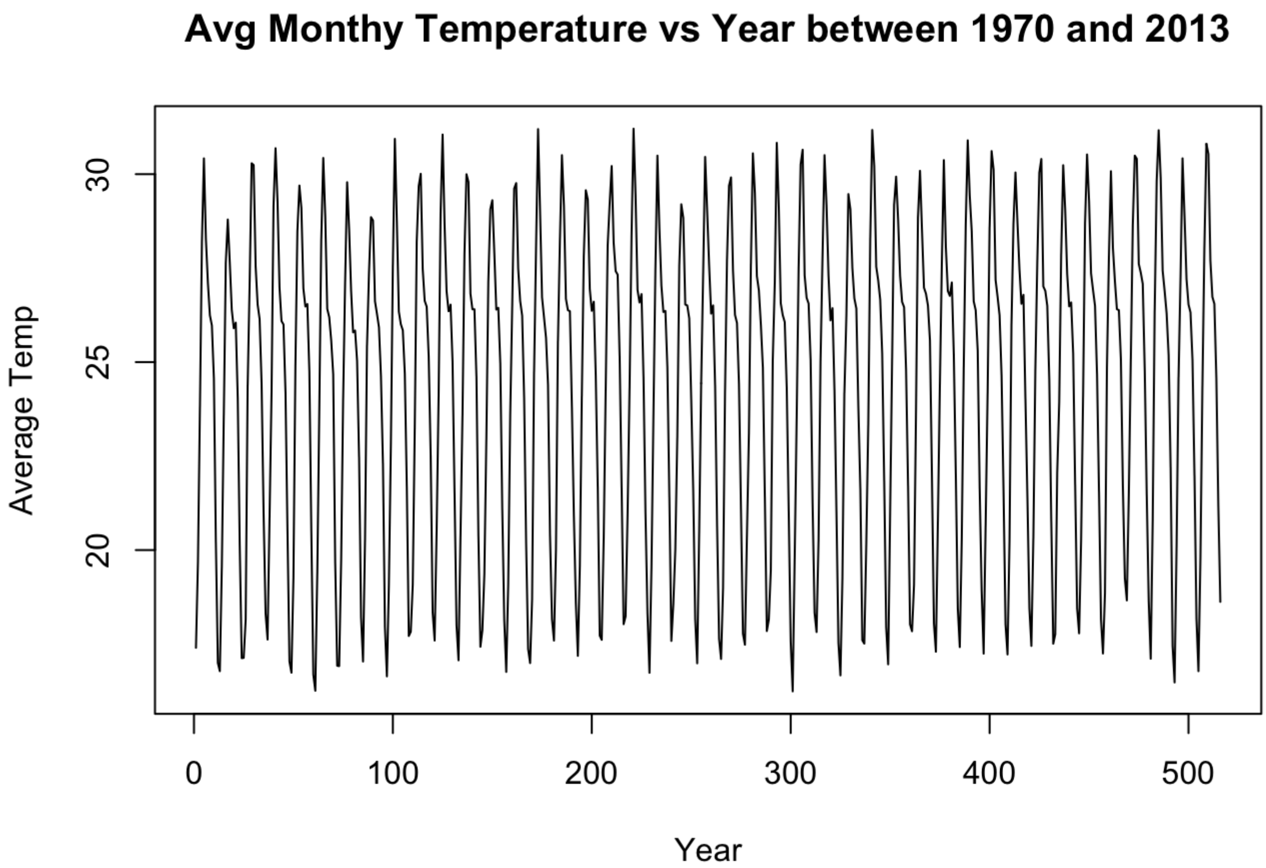


Here the yearly max and min temperatures are plotted and even they show same trend which means that our inference on the avg temperature above was correct.





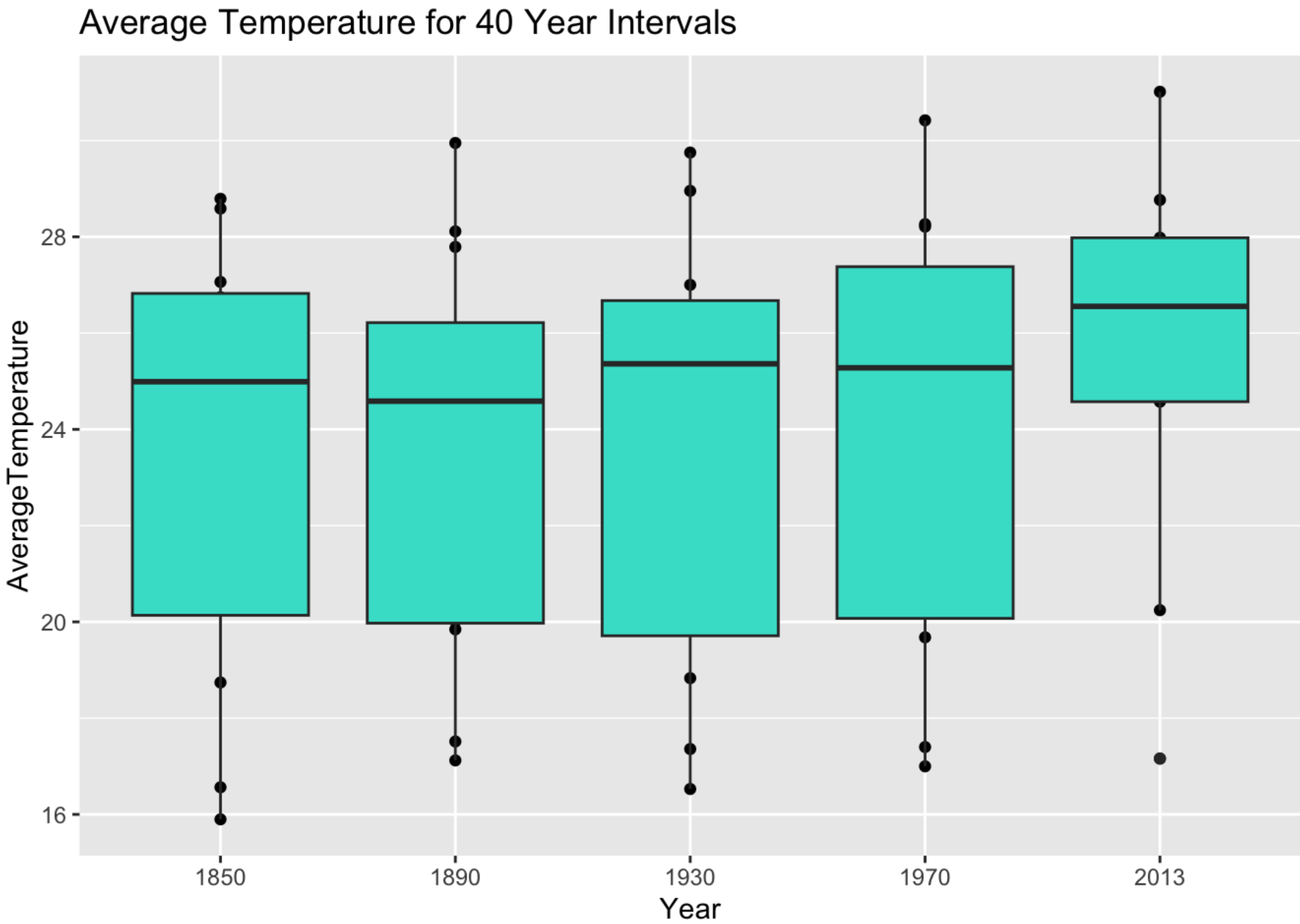




Here 40 yr splits are done

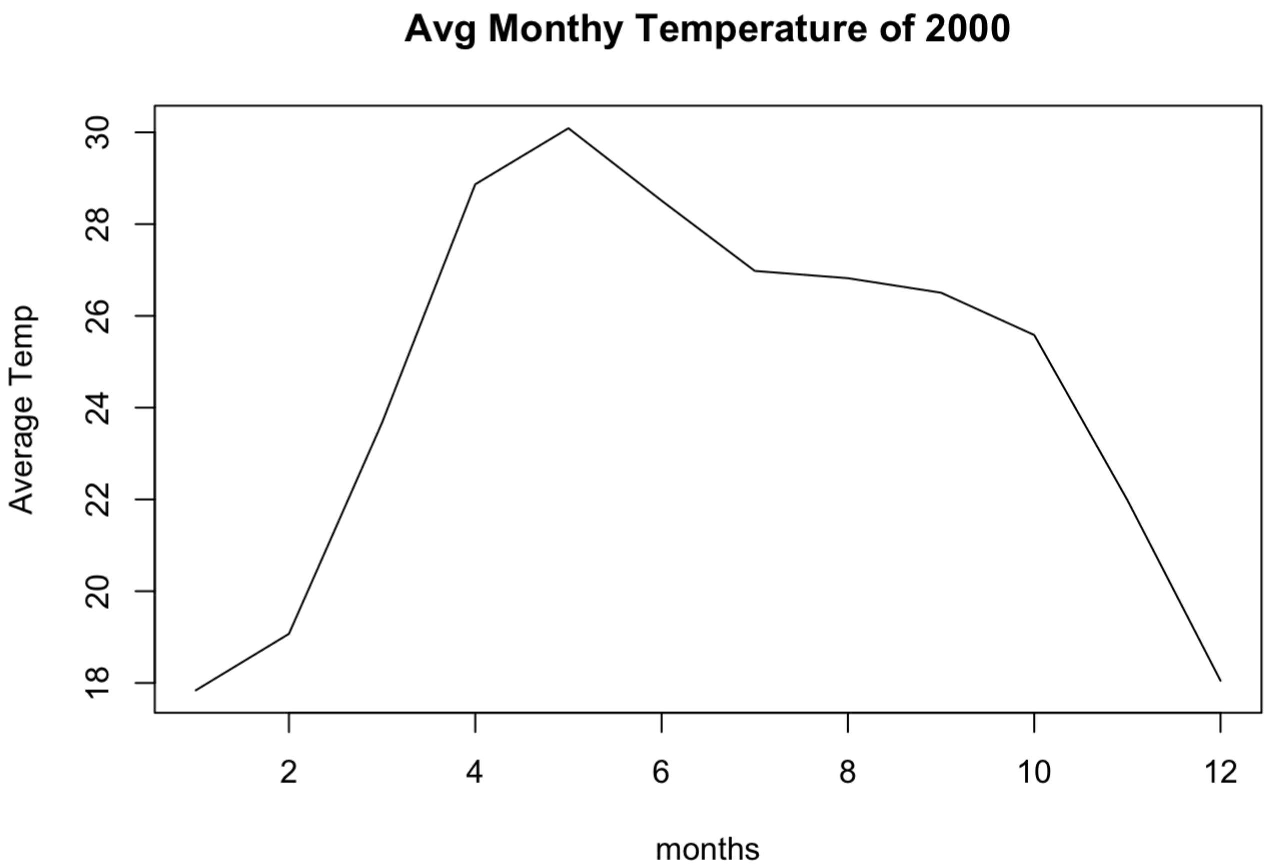
- The temperature range gradually increases as the year reaches 1930.

- This is also seen by seeing range of the graph
- Initially the lower is 15 but then it changes to 20 indicating rise in temperature.



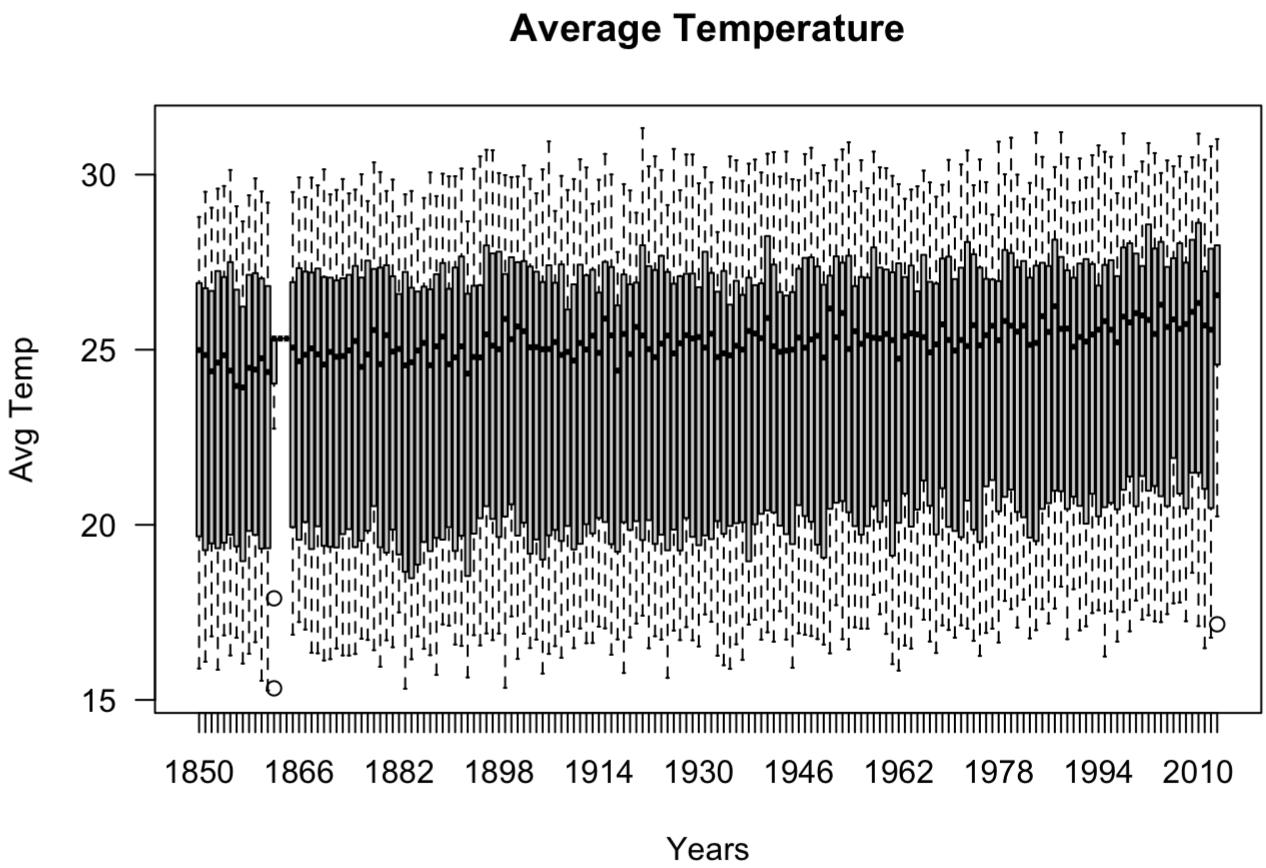
here 5 years are taken as a difference of 40 years

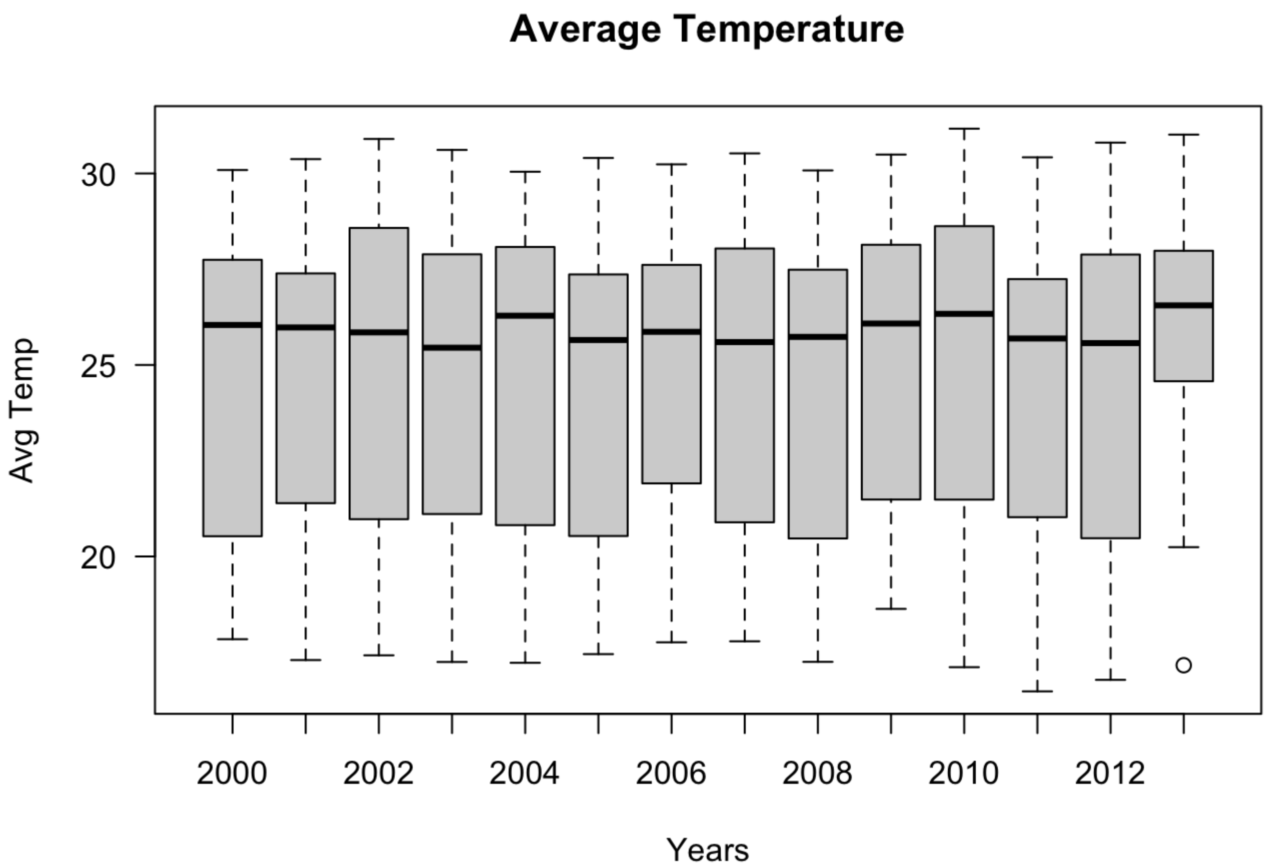
- It is observable that that both the range and median increases
- This further confirms the trend



Here seasonality is checked on the yr 2000 and same thing is observed.

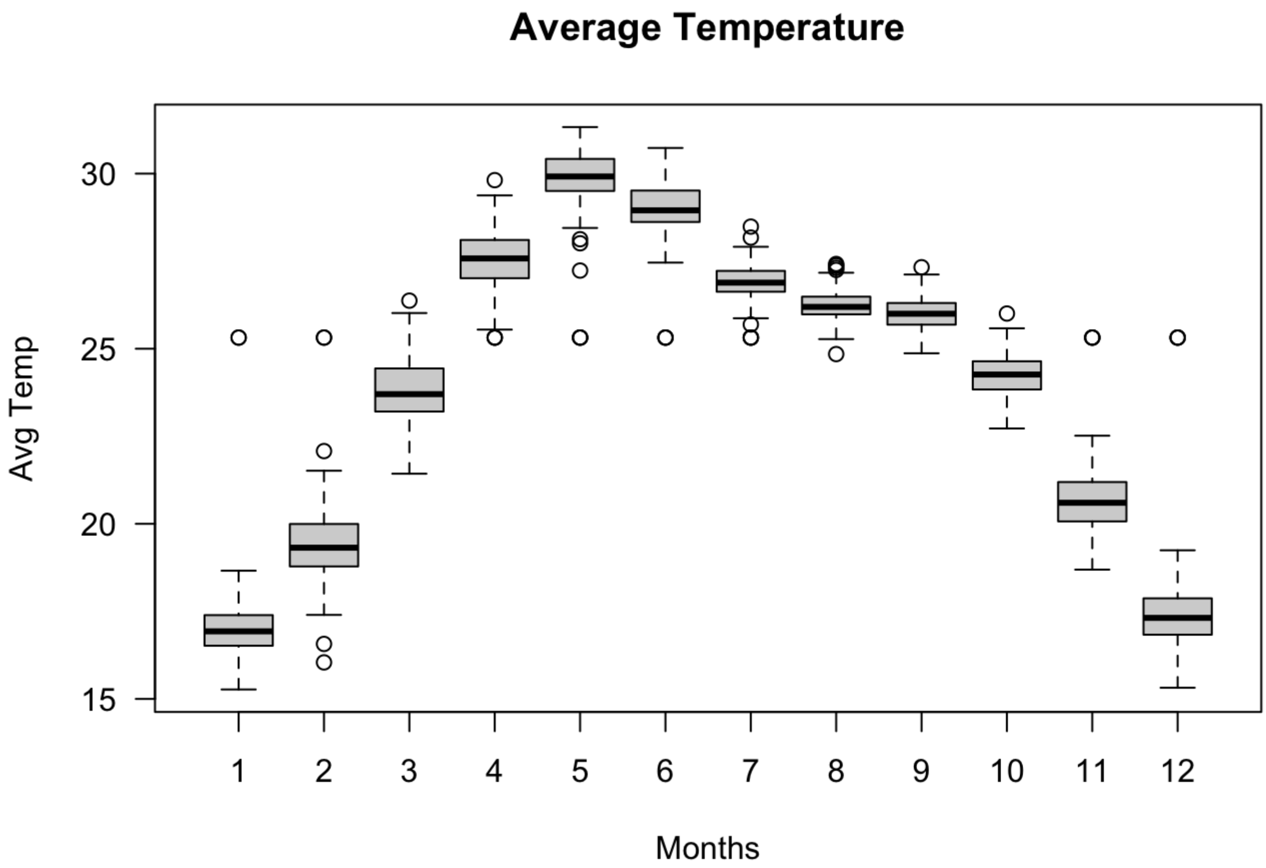
```
##     0%    25%    50%    75%   100%
## 15.270 20.070 25.318 27.148 31.329
```



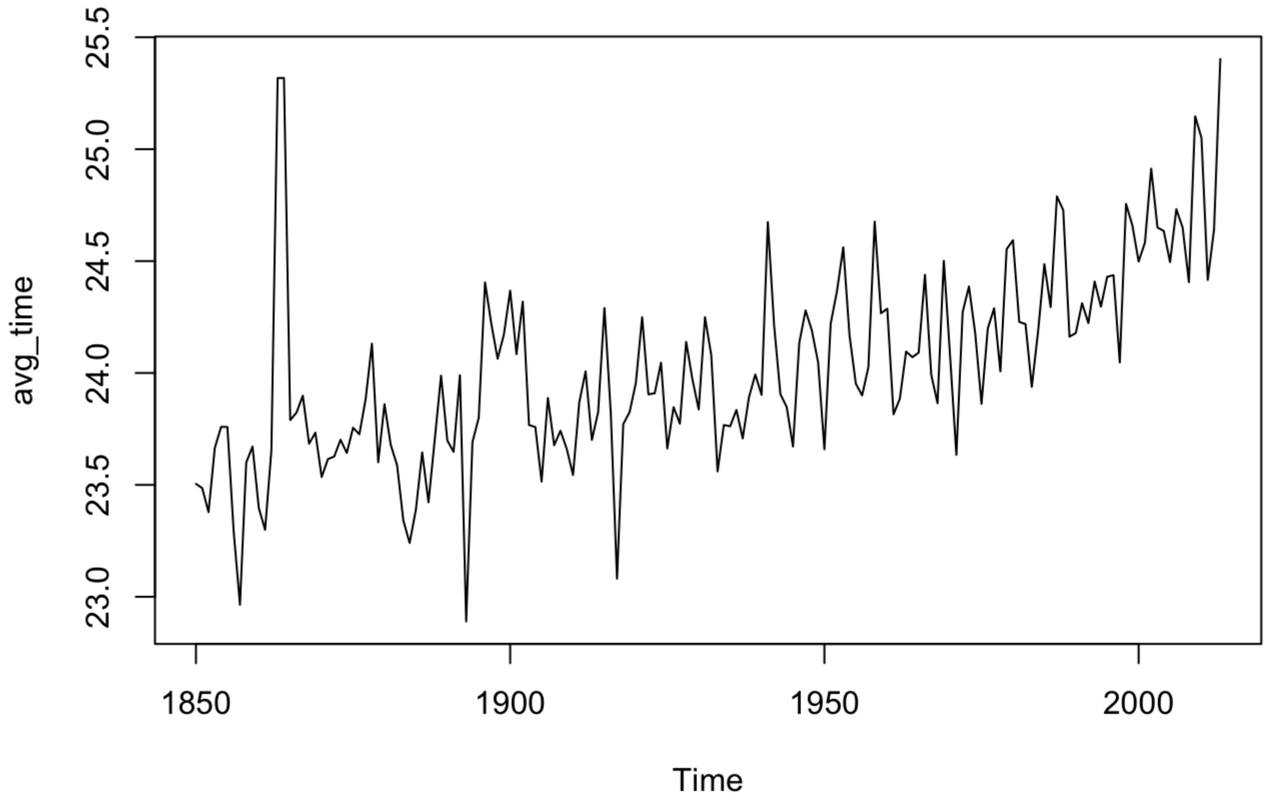


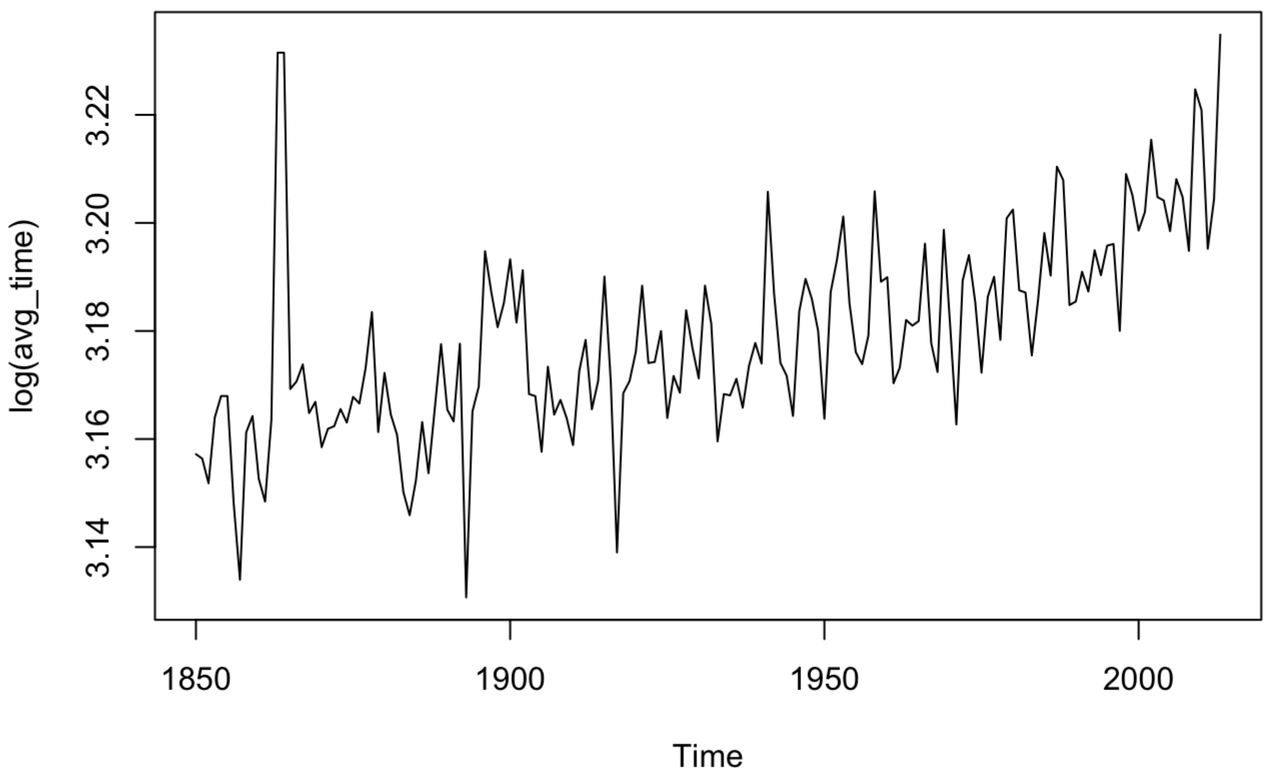
Boxplots are made on Yearly Average Temperatures and the

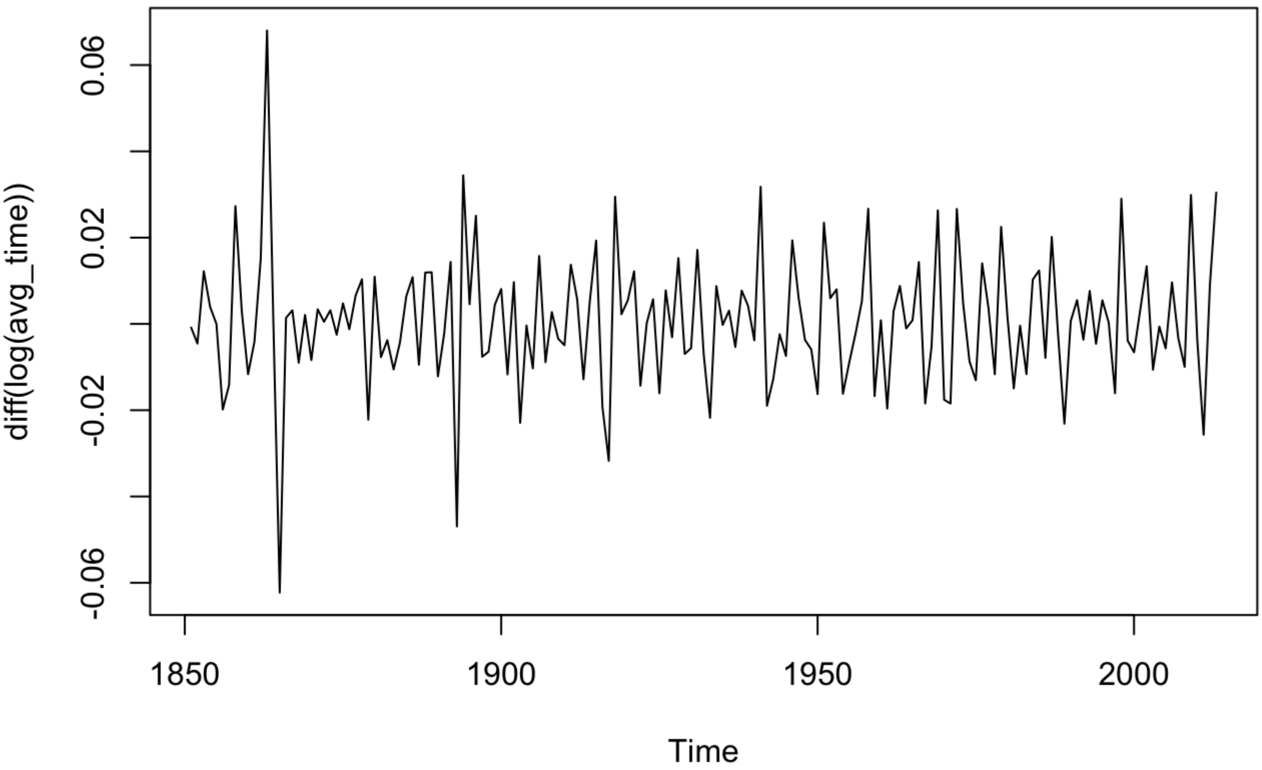
median increases over the years



Monthly boxplots







The initial time series is not stationary meaning the mean and variance any 2 time periods is not same.

We convert it into stationary time series by differencing the log values.

We will use Yearly average tempearature to make the model and forecast values

ARIMA Model

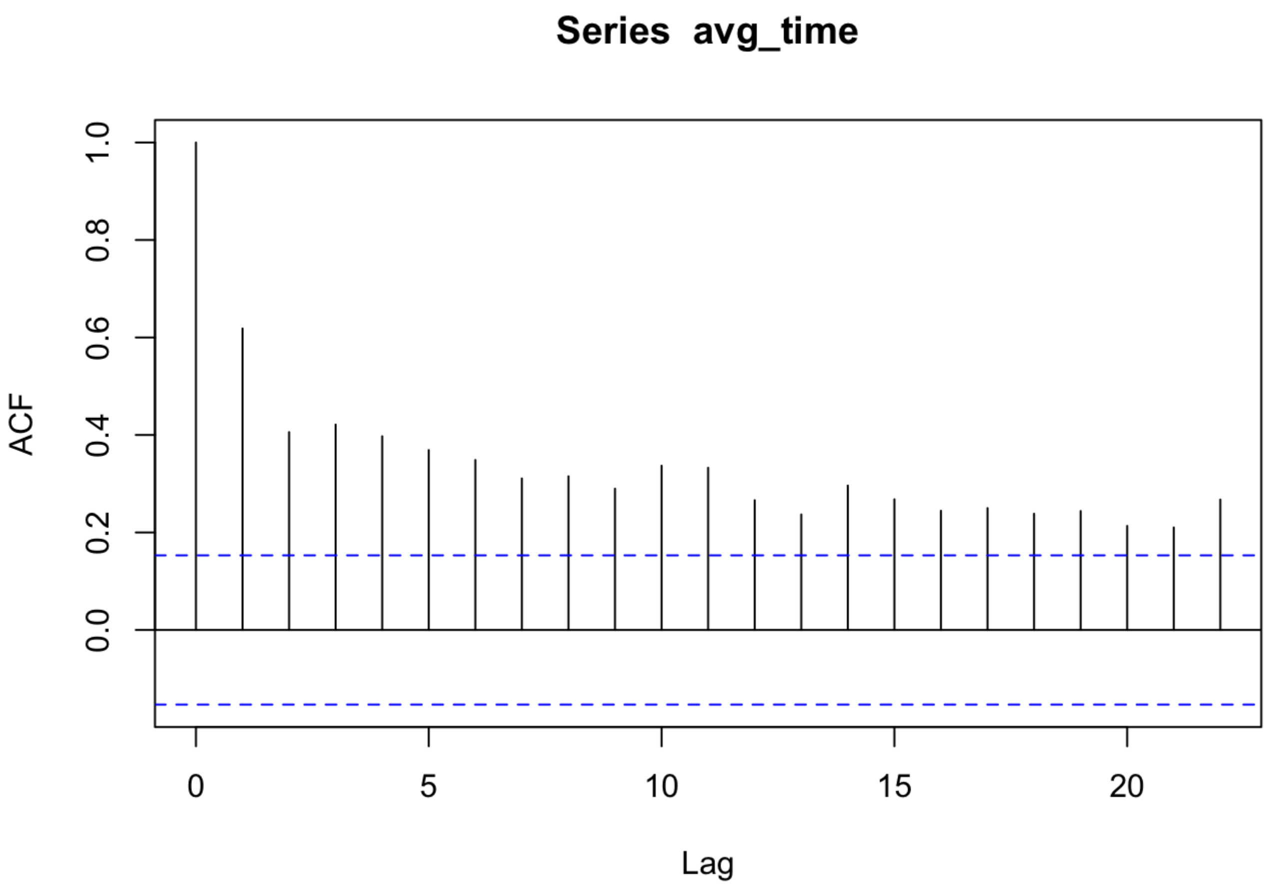
AR I MA

q d p

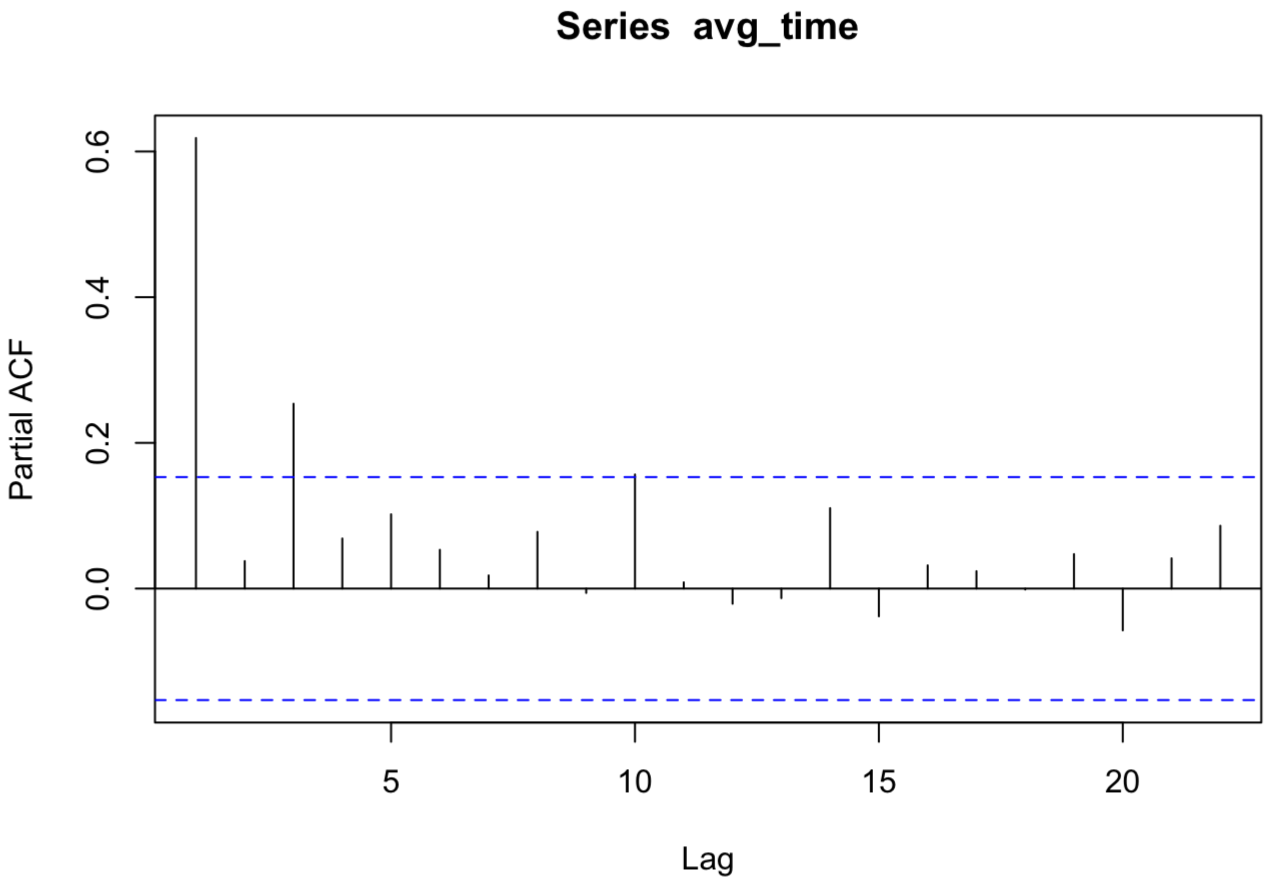
p = acf

q = pacf

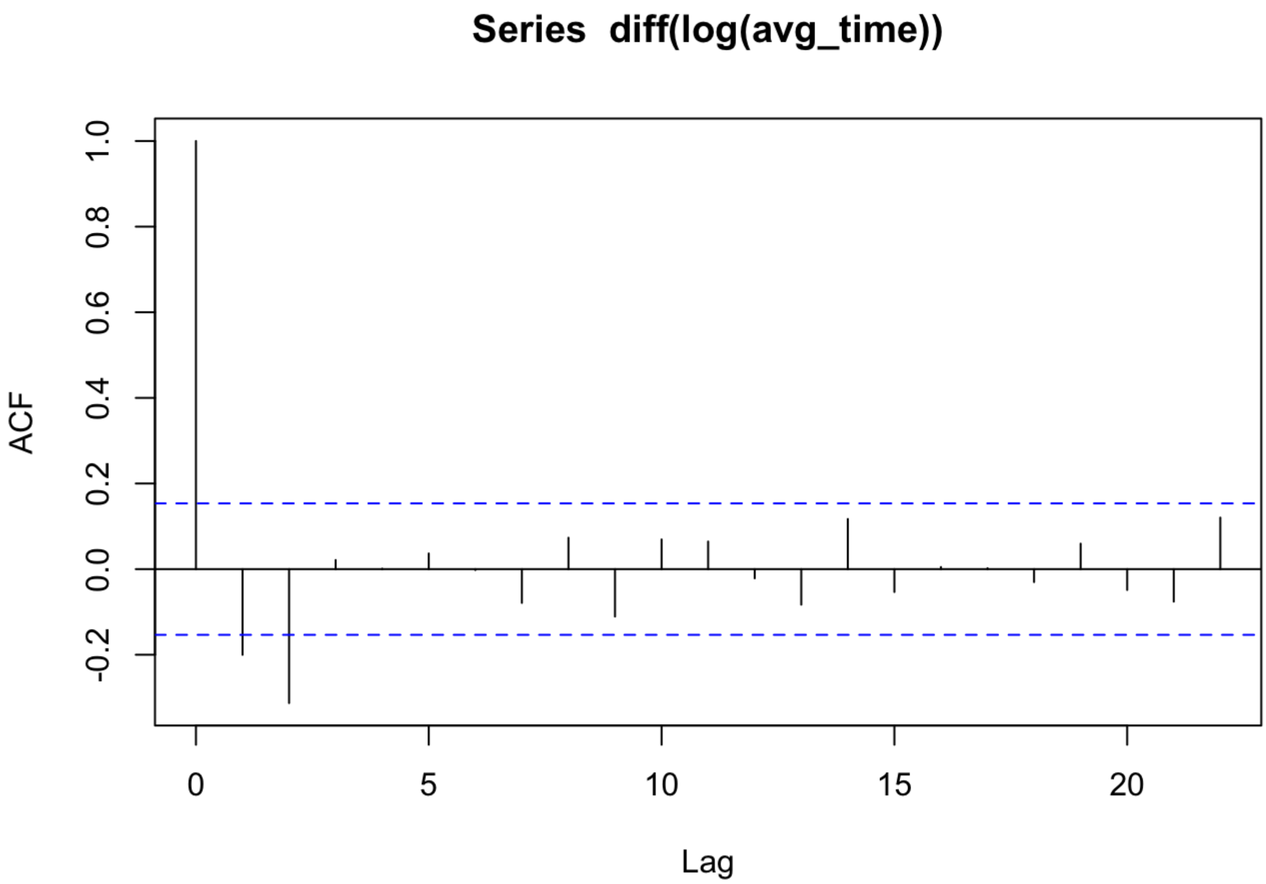
```
acf(avg_time) ## As non stationary time series, all lines are above the blue limit li
ne
```



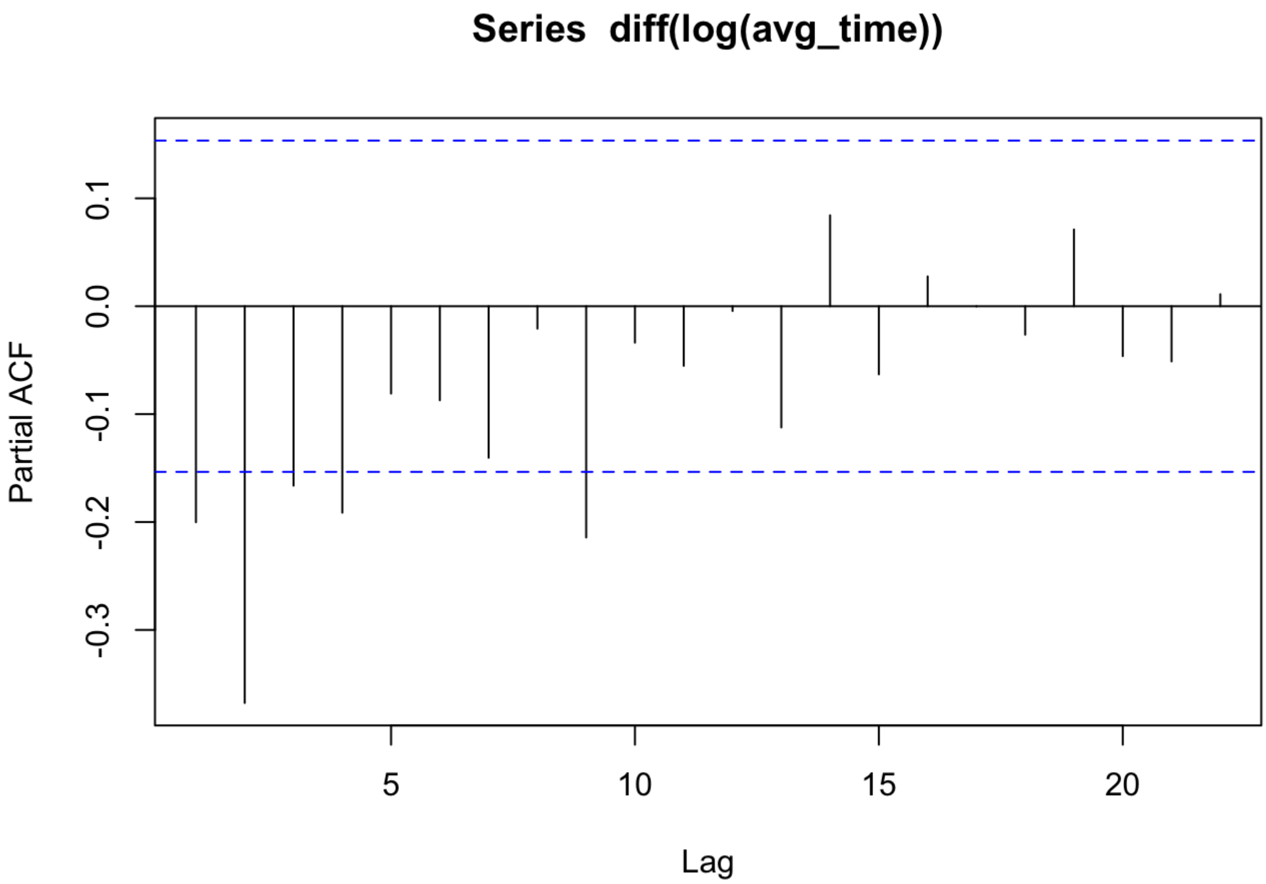

```
pacf(avg_time)
```




```
acf(diff(log(avg_time))) ## Most of the lines between the line after converting to st
ationary
```



```
pacf(diff(log(avg_time)))
```



For non stationary time series-

- All the lines in acf graph are above the blue line

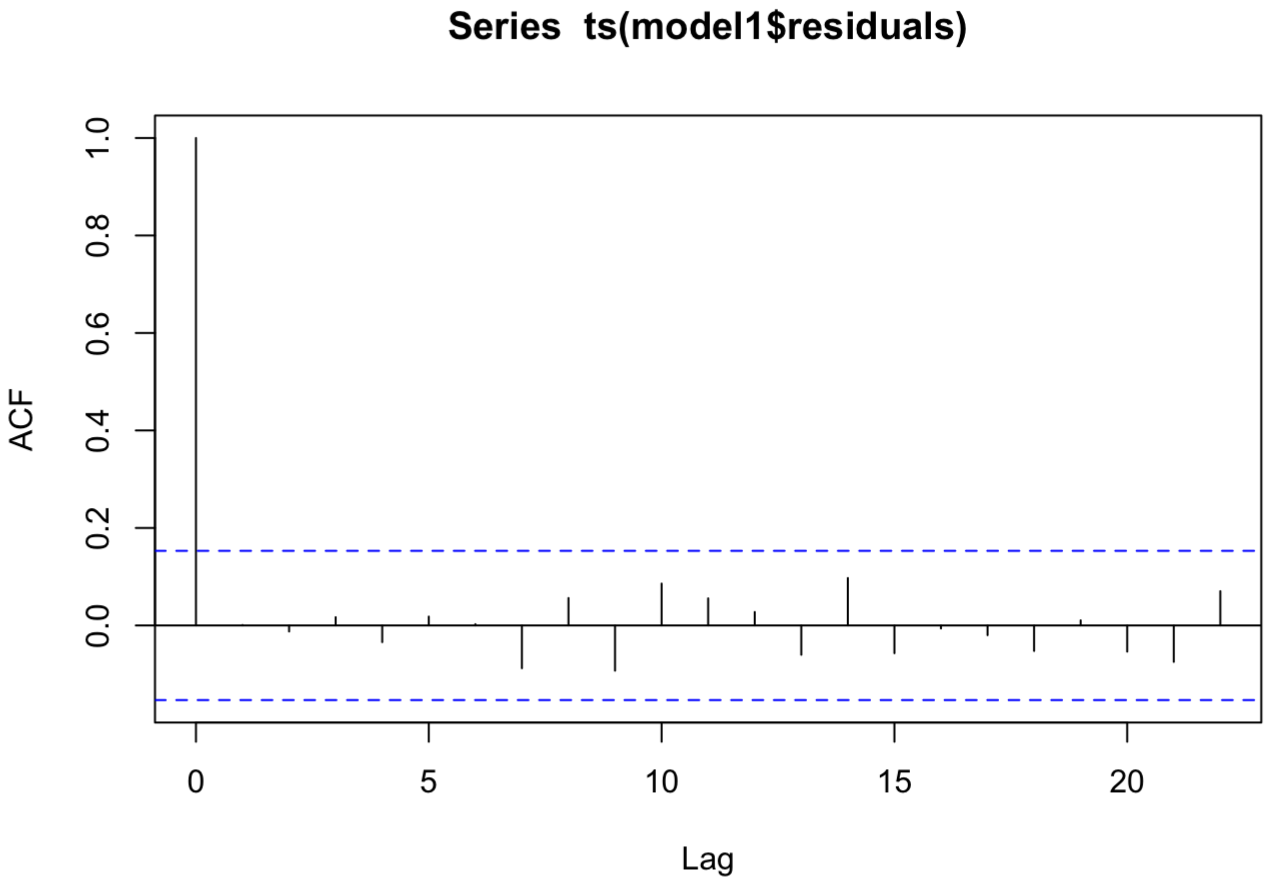
For stationary time series-

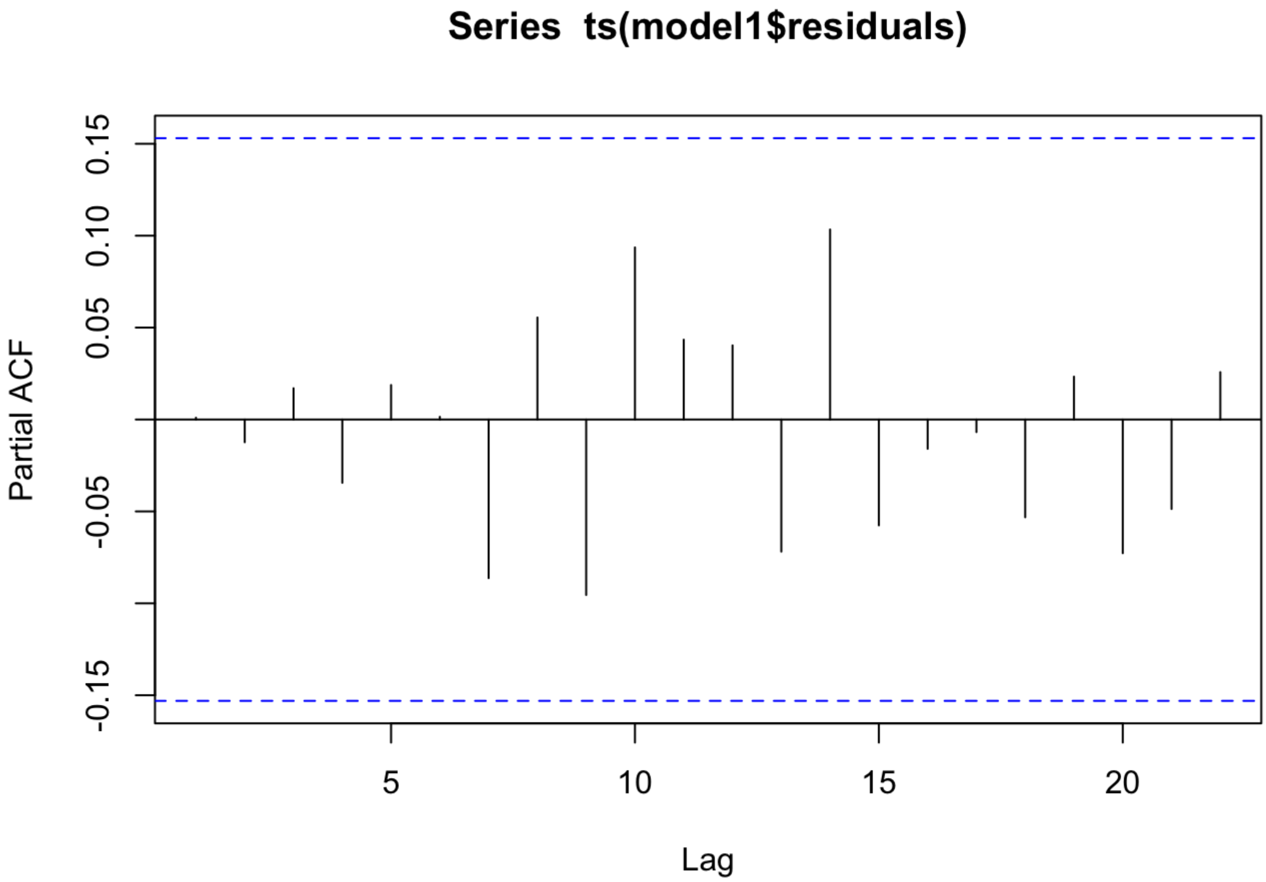
- Most of the lines fit between the blue lines in both acf and padf graph

Model

```
##
##  Fitting models using approximations to speed things up...
##
##  ARIMA(2,1,2) with drift         : 92.91879
##  ARIMA(0,1,0) with drift         : 141.4078
##  ARIMA(1,1,0) with drift         : 137.8059
##  ARIMA(0,1,1) with drift         : 109.9328
##  ARIMA(0,1,0)                    : 139.5722
##  ARIMA(1,1,2) with drift         : Inf
##  ARIMA(2,1,1) with drift         : 94.04269
##  ARIMA(3,1,2) with drift         : 92.89836
##  ARIMA(3,1,1) with drift         : 92.87764
##  ARIMA(3,1,0) with drift         : 112.9978
##  ARIMA(4,1,1) with drift         : 94.49213
##  ARIMA(2,1,0) with drift         : 115.2427
##  ARIMA(4,1,0) with drift         : 109.5969
##  ARIMA(4,1,2) with drift         : Inf
##  ARIMA(3,1,1)                    : 95.74711
##
##  Now re-fitting the best model(s) without approximations...
##
##  ARIMA(3,1,1) with drift         : 88.61981
##
##  Best model: ARIMA(3,1,1) with drift
```

```
## Series: avg_time
## ARIMA(3,1,1) with drift
##
## Coefficients:
##          ar1      ar2     ar3      ma1   drift
##       0.4537  -0.2098  0.1233  -0.9692  0.0064
## s.e.  0.0896   0.0863  0.0883   0.0493  0.0016
##
## sigma^2 = 0.09537:  log likelihood = -38.31
## AIC=88.62   AICc=89.16   BIC=107.18
```

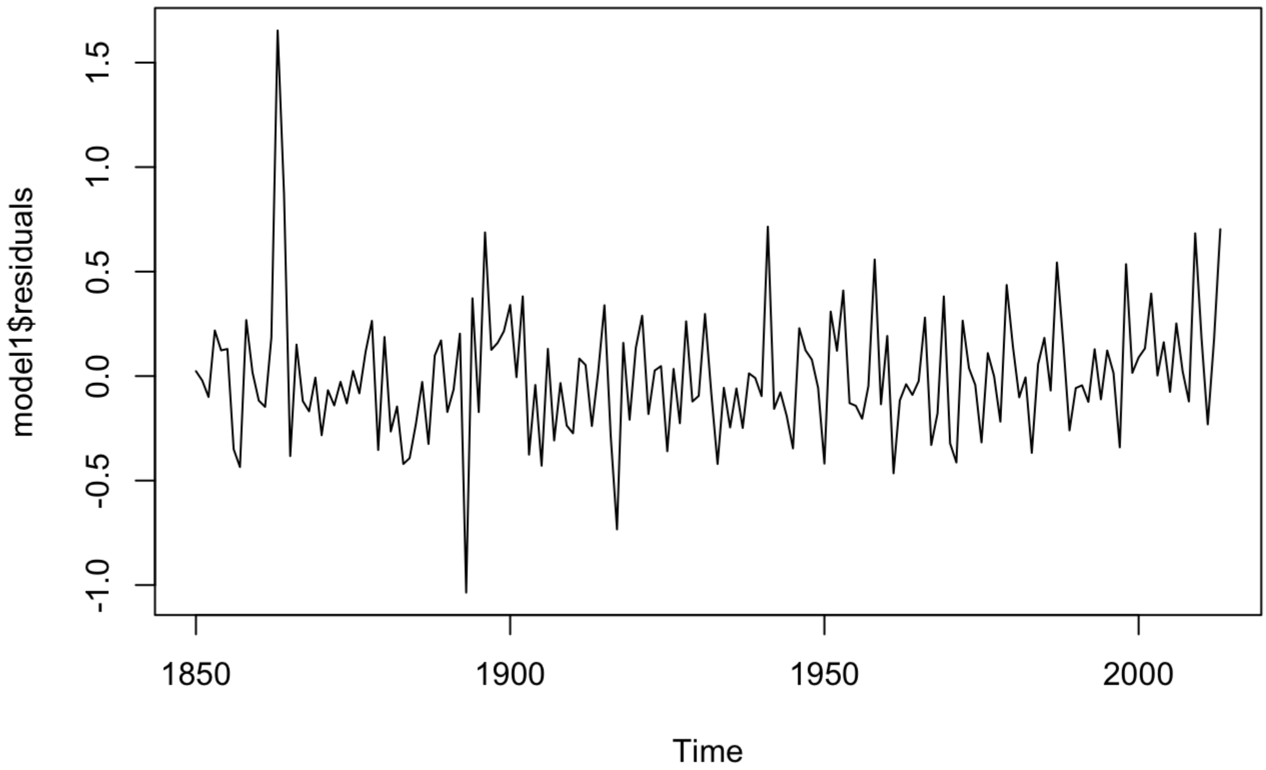

Series ts(model1$residuals)
ACF
1.0
0.8
0.6
0.4
0.2
0.0
0
5
10
15
20
Lag


Series ts(model1$residuals)
Partial ACF
0.15
0.10
0.05
-0.05
-0.15
5
10
15
20
Lag

```
##
## Fitting models using approximations to speed things up...
##
## ARIMA(2,1,2) with drift         : 7637.401
## ARIMA(0,1,0) with drift         : 9373.59
## ARIMA(1,1,0) with drift         : 8295.127
## ARIMA(0,1,1) with drift         : 8288.465
## ARIMA(0,1,0)                    : 9371.597
## ARIMA(1,1,2) with drift         : 7842.876
## ARIMA(2,1,1) with drift         : Inf
## ARIMA(3,1,2) with drift         : Inf
## ARIMA(2,1,3) with drift         : 7484.975
## ARIMA(1,1,3) with drift         : 7793.5
## ARIMA(3,1,3) with drift         : Inf
## ARIMA(2,1,4) with drift         : 6549.608
## ARIMA(1,1,4) with drift         : Inf
## ARIMA(3,1,4) with drift         : Inf
## ARIMA(2,1,5) with drift         : Inf
## ARIMA(1,1,5) with drift         : Inf
## ARIMA(3,1,5) with drift         : 6545.77
## ARIMA(4,1,5) with drift         : Inf
## ARIMA(4,1,4) with drift         : Inf
## ARIMA(3,1,5)                    : 6544.004
## ARIMA(2,1,5)                    : Inf
## ARIMA(3,1,4)                    : 6539.207
## ARIMA(2,1,4)                    : 6547.617
## ARIMA(3,1,3)                    : Inf
## ARIMA(4,1,4)                    : Inf
## ARIMA(2,1,3)                    : 7483.011
## ARIMA(4,1,3)                    : Inf
## ARIMA(4,1,5)                    : Inf
##
## Now re-fitting the best model(s) without approximations...
##
## ARIMA(3,1,4)                    : 6547.374
##
## Best model: ARIMA(3,1,4)
```

```
##
##  Fitting models using approximations to speed things up...
##
##  ARIMA(2,1,2) with drift         : 92.91879
##  ARIMA(0,1,0) with drift         : 141.4078
##  ARIMA(1,1,0) with drift         : 137.8059
##  ARIMA(0,1,1) with drift         : 109.9328
##  ARIMA(0,1,0)                    : 139.5722
##  ARIMA(1,1,2) with drift         : Inf
##  ARIMA(2,1,1) with drift         : 94.04269
##  ARIMA(3,1,2) with drift         : 92.89836
##  ARIMA(3,1,1) with drift         : 92.87764
##  ARIMA(3,1,0) with drift         : 112.9978
##  ARIMA(4,1,1) with drift         : 94.49213
##  ARIMA(2,1,0) with drift         : 115.2427
##  ARIMA(4,1,0) with drift         : 109.5969
##  ARIMA(4,1,2) with drift         : Inf
##  ARIMA(3,1,1)                    : 95.74711
##
##  Now re-fitting the best model(s) without approximations...
##
##  ARIMA(3,1,1) with drift         : 88.61981
##
##  Best model: ARIMA(3,1,1) with drift
```

```
## Series: Ind_avgYr$Temp
## ARIMA(3,1,1) with drift
##
## Coefficients:
##          ar1      ar2     ar3      ma1   drift
##       0.4537  -0.2098  0.1233  -0.9692  0.0064
## s.e.  0.0896   0.0863  0.0883   0.0493  0.0016
##
## sigma^2 = 0.09537:  log likelihood = -38.31
## AIC=88.62   AICc=89.16   BIC=107.18
```

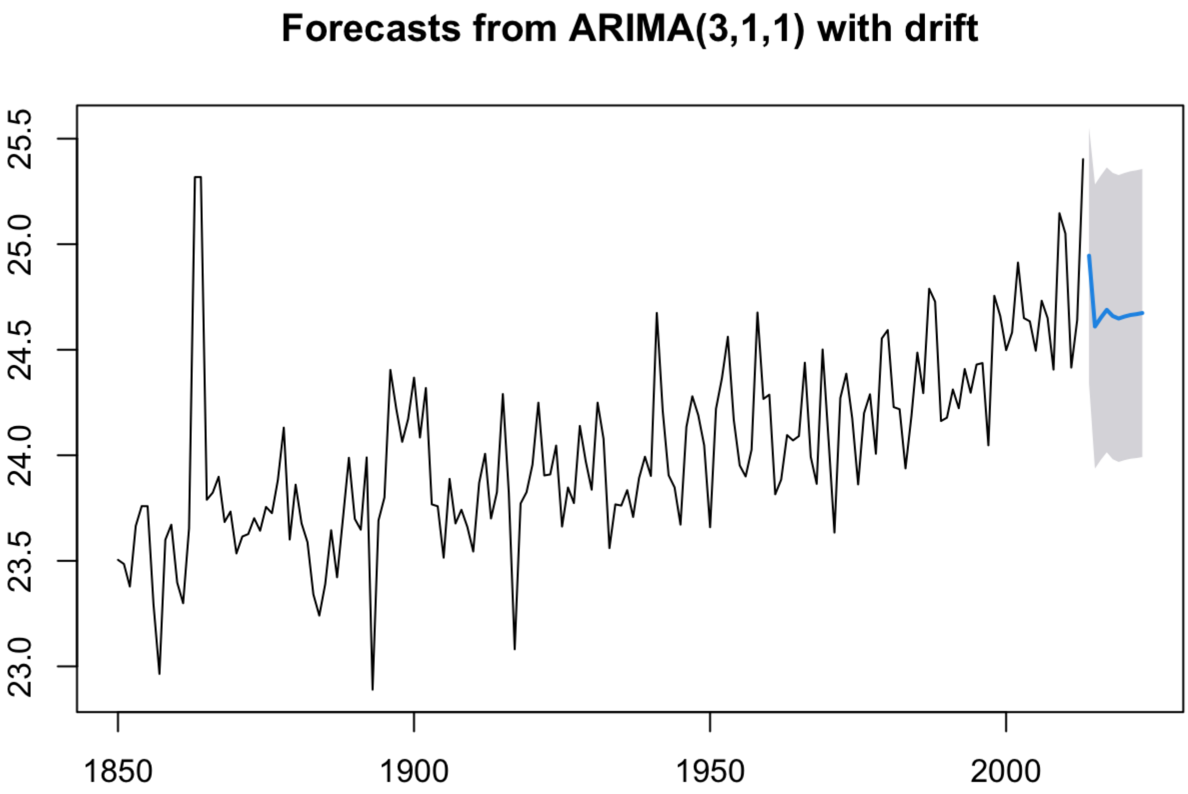


Here using the auto.arima model, t

The residuals when plotted by acf and pacf show very less auto-correlation betwenn them.

Forecast next 10 years

```
forecast1 = forecast(model1, level=c(95),h = 10)
plot(forecast1) ## the trend continues as the avg temperature continues to increase o
n yearly basis
```




```
print(forecast1)
```

```
##      Point Forecast    Lo 95    Hi 95
## 2014       24.94554 24.34025 25.55082
## 2015       24.61017 23.93759 25.28275
## 2016       24.65194 23.97891 25.32497
## 2017       24.68896 24.01456 25.36336
## 2018       24.65972 23.98179 25.33764
## 2019       24.64791 23.96855 25.32726
## 2020       24.65733 23.97733 25.33732
## 2021       24.66455 23.98384 25.34525
## 2022       24.66846 23.98700 25.34993
## 2023       24.67397 23.99183 25.35610
```

```
f_df = as.data.frame(forecast1)
print(paste("Avg Temperature in 2020: ",f_df$`Point Forecast`[7]))
```

```
## [1] "Avg Temperature in 2020:  24.6573257385717"
```

```
print(paste("Avg Temperature in 2021: ",f_df$`Point Forecast`[8]))
```

```
## [1] "Avg Temperature in 2021:  24.6645457634773"
```

```
print(paste("Avg Temperature in 2022: ",f_df$`Point Forecast`[9]))
```

```
## [1] "Avg Temperature in 2022:  24.668464926972"
```

```
forecasted_values = c(24.95,24.61,24.65,24.68,24.65,24.64,24.65,24.66)
observed_values = c(24.79,24.91,25.27,25.16,25.01,24.95,24.8,24.99)
df2= data.frame(Year=c(2014,2015,2016,2017,2018,2019,2020,2021),Forecast=forecasted_v
alues,Actual=observed_values)
df2
```

```
##   Year Forecast Actual
## 1 2014    24.95  24.79
## 2 2015    24.61  24.91
## 3 2016    24.65  25.27
## 4 2017    24.68  25.16
## 5 2018    24.65  25.01
## 6 2019    24.64  24.95
## 7 2020    24.65  24.80
## 8 2021    24.66  24.99
```

```
deviation=observed_values-forecasted_values
mean(deviation)
```

```
## [1] 0.29875
```

This shows that the Average deviation is a mere 0.3 degrees and thus shows this model is accurate.

The forecasts although not 100% accurate, are pretty close to real values.

The forecast for 2021 only varies by 1 degree.

The forecasted values in plot staify the trend that the temperature increases steadily.

Validation

```
Box.test(model1$residuals, lag=5, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  model1$residuals
## X-squared = 0.33079, df = 5, p-value = 0.997
```

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

```
##
##  Box-Ljung test
##
## data:  model1$residuals
## X-squared = 5.0375, df = 10, p-value = 0.8887
```

```
Box.test(model1$residuals, lag=15, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  model1$residuals
## X-squared = 8.6888, df = 15, p-value = 0.8932
```

```
## p-values here are all above 0.05 hence good model
```

p-values here are all above 0.05 hence good model

FBs Prophet Model

```
# column : ds,y

df_y_ts = data.frame(ds=IND_data$Year,y=IND_data$AverageTemperature)

df_y_ts$ds = as.character(df_y_ts$ds)
df_y_ts$ds <- as.Date(df_y_ts$ds, format="%Y")

prop_fit <- prophet(df_y_ts)
```

```
## Disabling weekly seasonality. Run prophet with weekly.seasonality=TRUE to override
this.
```

```
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override t
his.
```

```
Future1 = make_future_dataframe(prop_fit,periods =5, freq = "year")
tail(Future1)
```

```
##             ds
## 164 2013-04-04
## 165 2014-04-04
## 166 2015-04-04
## 167 2016-04-04
## 168 2017-04-04
## 169 2018-04-04
```

```
forecast1 = predict(prop_fit,Future1)
tail(forecast1[c('ds','yhat')])
```

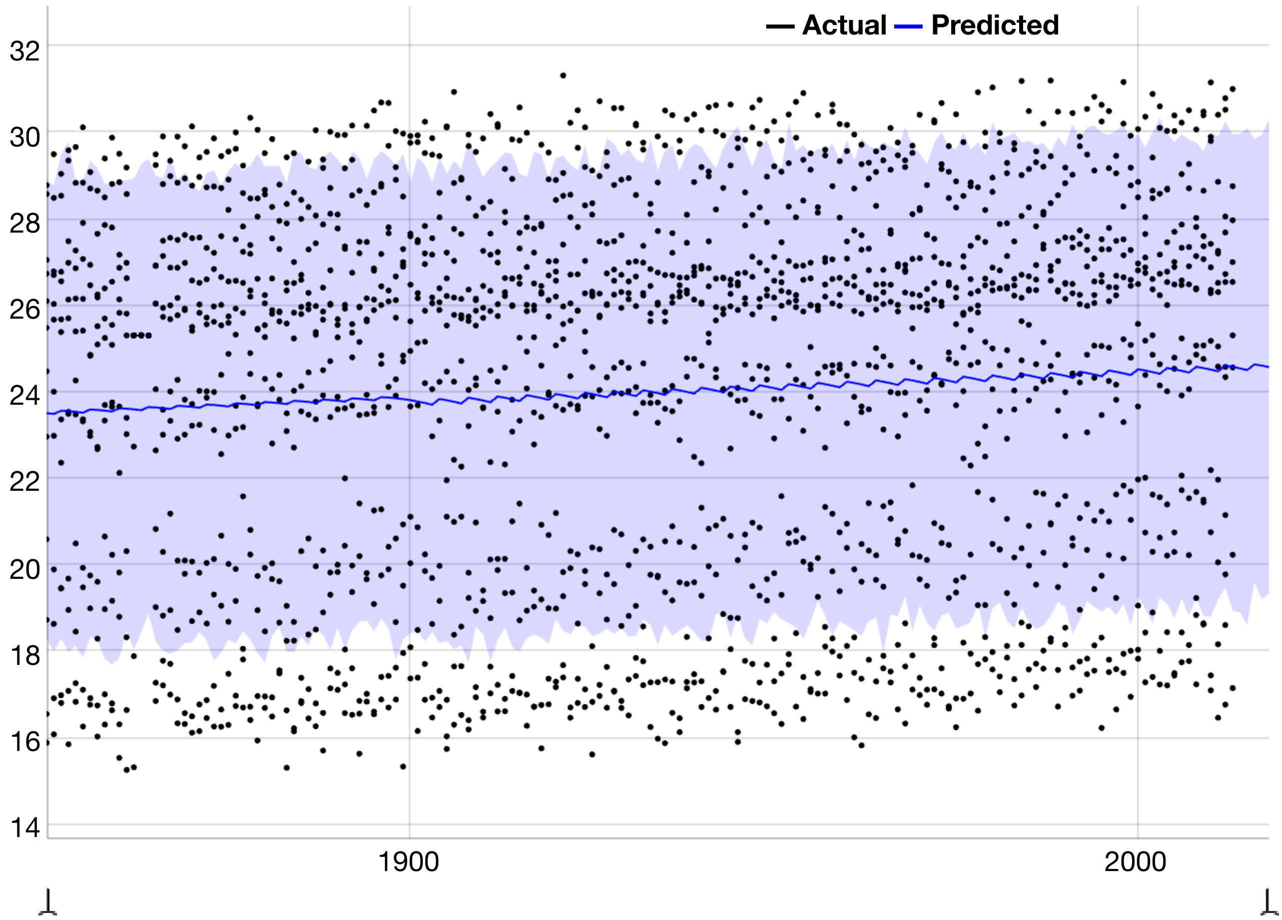
```
##             ds     yhat
## 164 2013-04-04 24.58965
## 165 2014-04-04 24.55488
## 166 2015-04-04 24.51533
## 167 2016-04-04 24.64819
## 168 2017-04-04 24.61824
## 169 2018-04-04 24.58347
```

```
prop_fit$component.modes
```

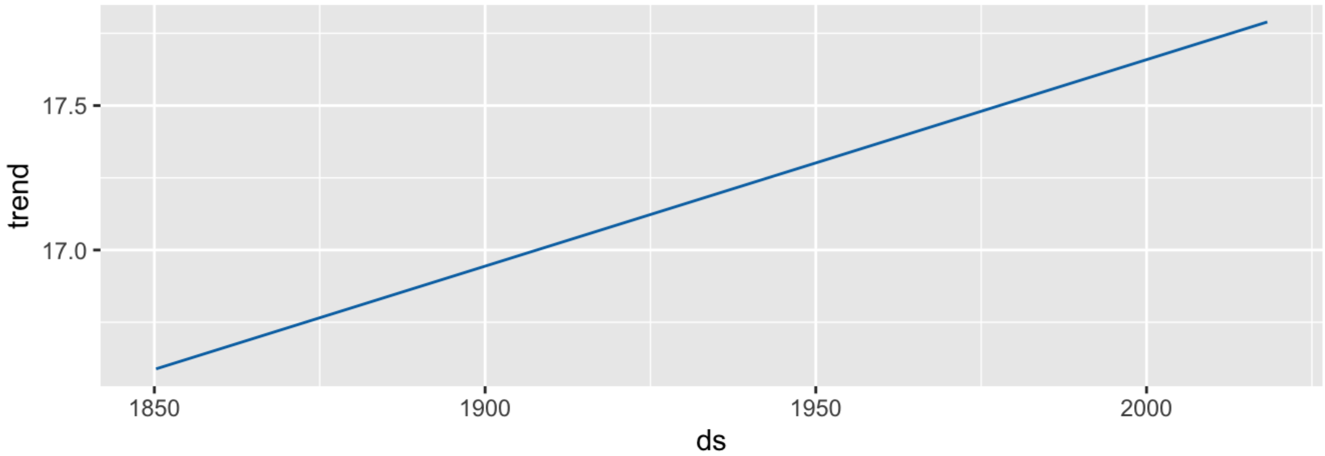
```
## $additive
## [1] "yearly"                      "additive_terms"
## [3] "extra_regressors_additive" "holidays"
##
## $multiplicative
## [1] "multiplicative_terms"            "extra_regressors_multiplicative"
```

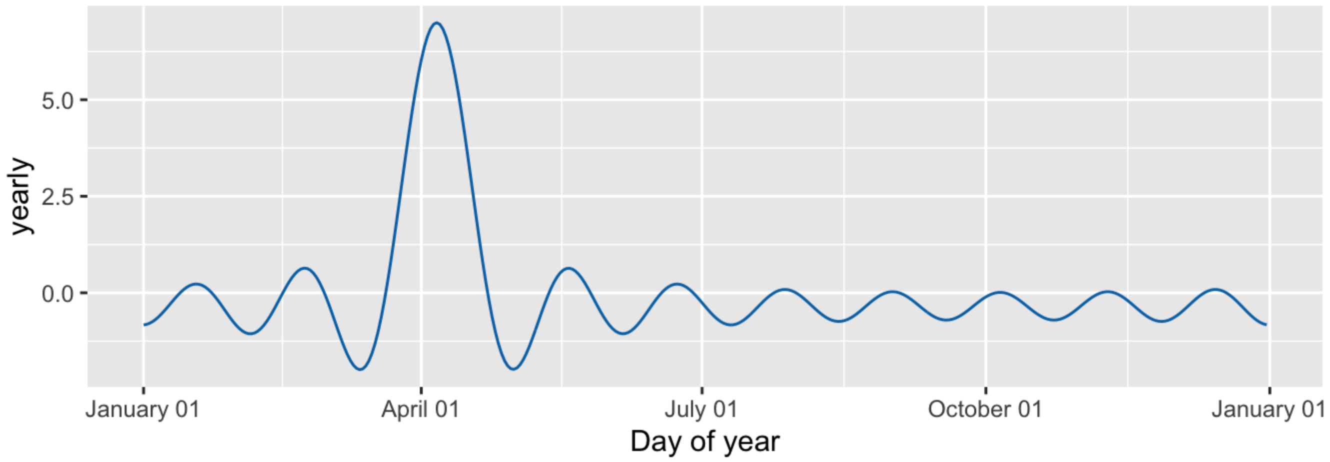
```
dyplot.prophet(prop_fit,forecast1)
```

```
## Warning: `select_()` was deprecated in dplyr 0.7.0.
## ℹ Please use `select()` instead.
## ℹ The deprecated feature was likely used in the dplyr package.
##   Please report the issue at < ]8;;https://github.com/tidyverse/dplyr/issues http
s://github.com/tidyverse/dplyr/issues ]8;; >.
```



```
prophet_plot_components(prop_fit,forecast1)
```






```
df_y_ts = data.frame(ds=Ind_avgYr$Year,y=Ind_avgYr$Temp)

df_y_ts$ds = as.character(df_y_ts$ds)
df_y_ts$ds <- as.Date(df_y_ts$ds, format="%Y")

prop_fit <- prophet(df_y_ts)
```

```
## Disabling weekly seasonality. Run prophet with weekly.seasonality=TRUE to override
this.
```

```
## Disabling daily seasonality. Run prophet with daily.seasonality=TRUE to override t
his.
```

```
Future1 = make_future_dataframe(prop_fit,periods =10, freq = "year")
tail(Future1,10)
```

```
##             ds
## 165 2014-04-04
## 166 2015-04-04
## 167 2016-04-04
## 168 2017-04-04
## 169 2018-04-04
## 170 2019-04-04
## 171 2020-04-04
## 172 2021-04-04
## 173 2022-04-04
## 174 2023-04-04
```

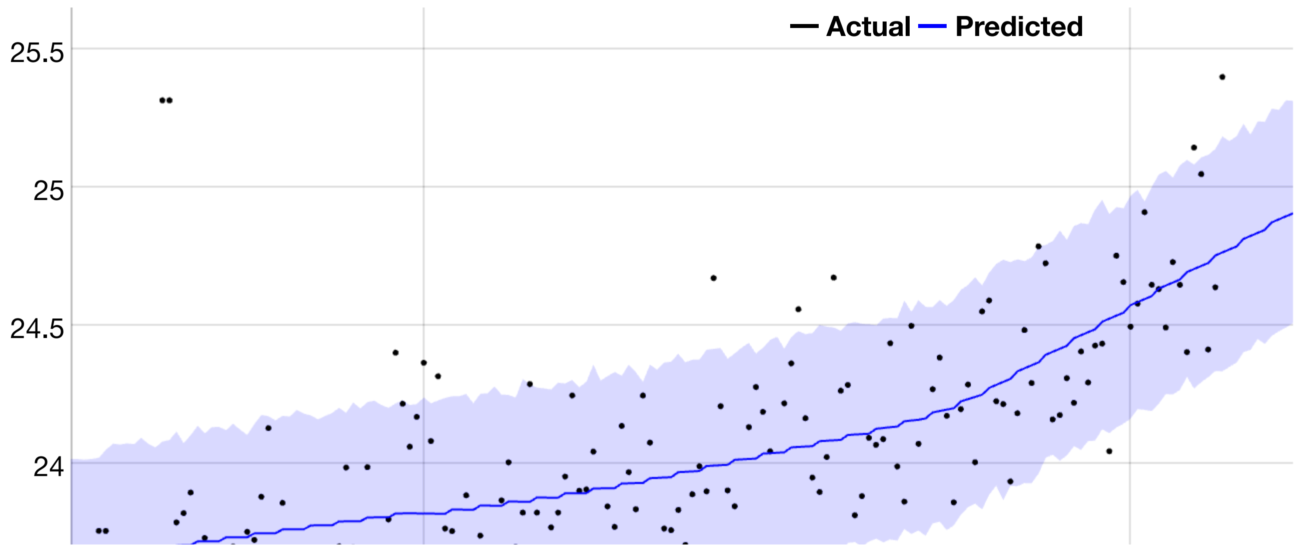
```
forecast1 = predict(prop_fit,Future1)
tail(forecast1[c('ds','yhat')],10)
```

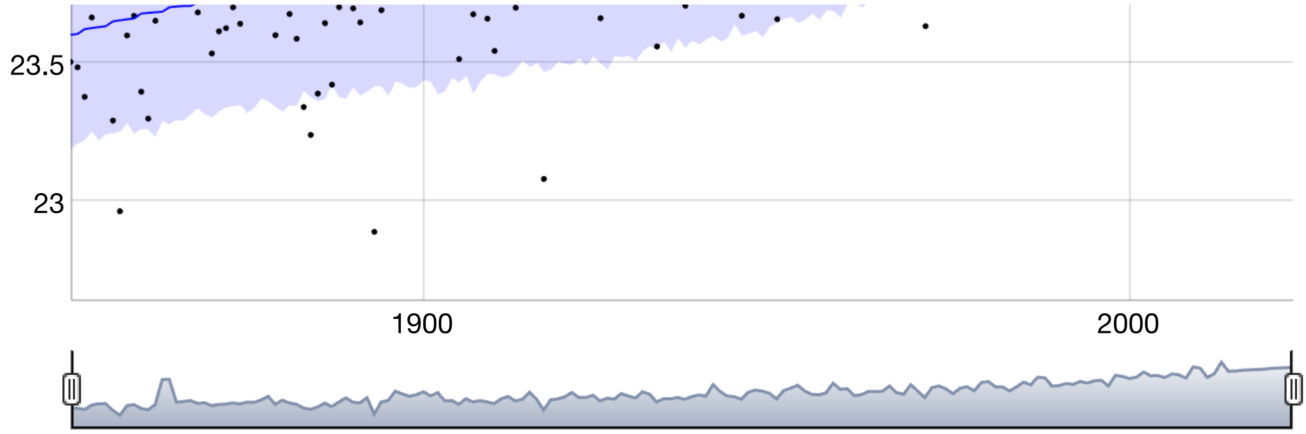
```
##             ds     yhat
## 165 2014-04-04 24.77794
## 166 2015-04-04 24.78866
## 167 2016-04-04 24.81604
## 168 2017-04-04 24.82703
## 169 2018-04-04 24.83788
## 170 2019-04-04 24.84860
## 171 2020-04-04 24.87597
## 172 2021-04-04 24.88697
## 173 2022-04-04 24.89782
## 174 2023-04-04 24.90854
```

```
prop_fit$component.modes
```

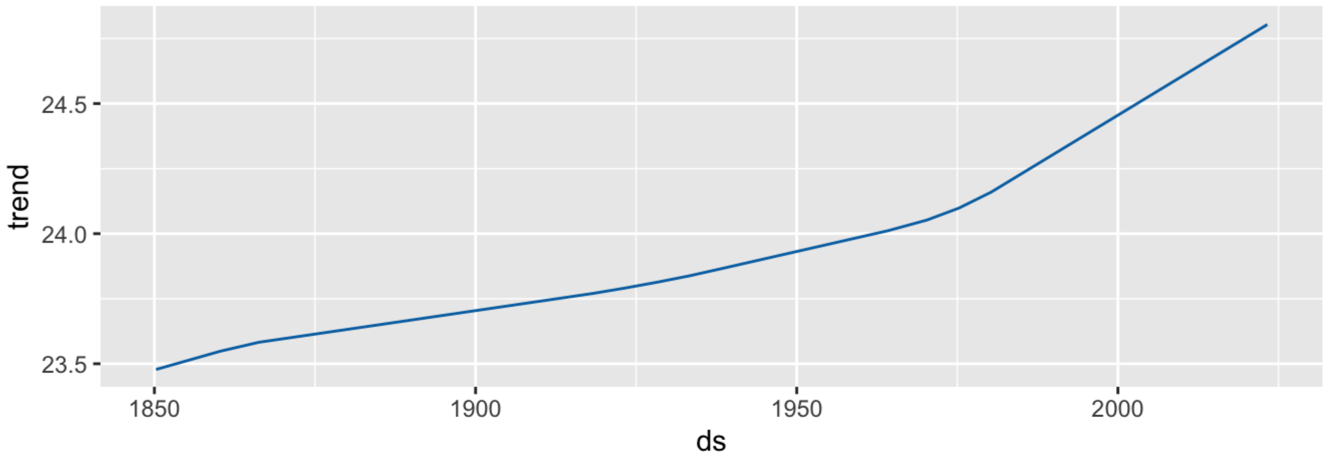
```
## $additive
## [1] "yearly"                      "additive_terms"
## [3] "extra_regressors_additive" "holidays"
##
## $multiplicative
## [1] "multiplicative_terms"             "extra_regressors_multiplicative"
```

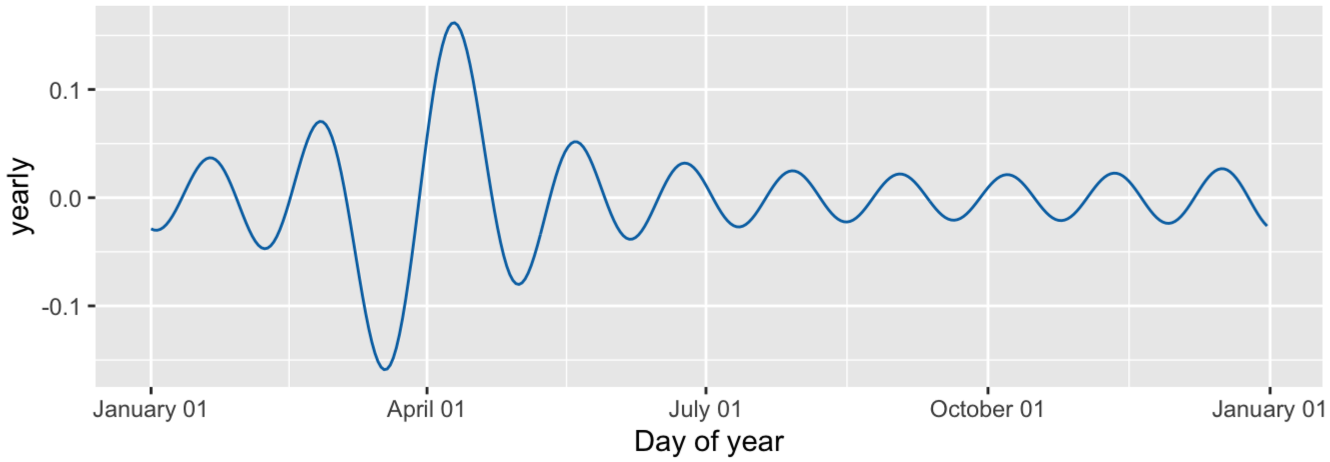
```
dyplot.prophet(prop_fit,forecast1)
```






```
prophet_plot_components(prop_fit,forecast1)
```






```
forecasted_pvalues=c(24.77,24.78,24.81,24.82,24.83,24.84,24.87,24.88)
df1= data.frame(Year=c(2014,2015,2016,2017,2018,2019,2020,2021),Forecast=forecasted_p
values,Actual=observed_values)
df1
```

```
##   Year Forecast Actual
## 1 2014    24.77  24.79
## 2 2015    24.78  24.91
## 3 2016    24.81  25.27
## 4 2017    24.82  25.16
## 5 2018    24.83  25.01
## 6 2019    24.84  24.95
## 7 2020    24.87  24.80
## 8 2021    24.88  24.99
```

```
proph_deviation=observed_values-forecasted_pvalues
mean(proph_deviation)
```

```
## [1] 0.16
```

Which shows a mean deviation of only 0.16 degrees compared to the actual temperature.

CONCLUSION: #Prophet model gives a forecast which is almost similar to ARIMA model’s forecast. # The interactive graph plot can be used to see the forecast values and trend for the future. # Plot component function showcases the trend and yearly additive component of the time series.