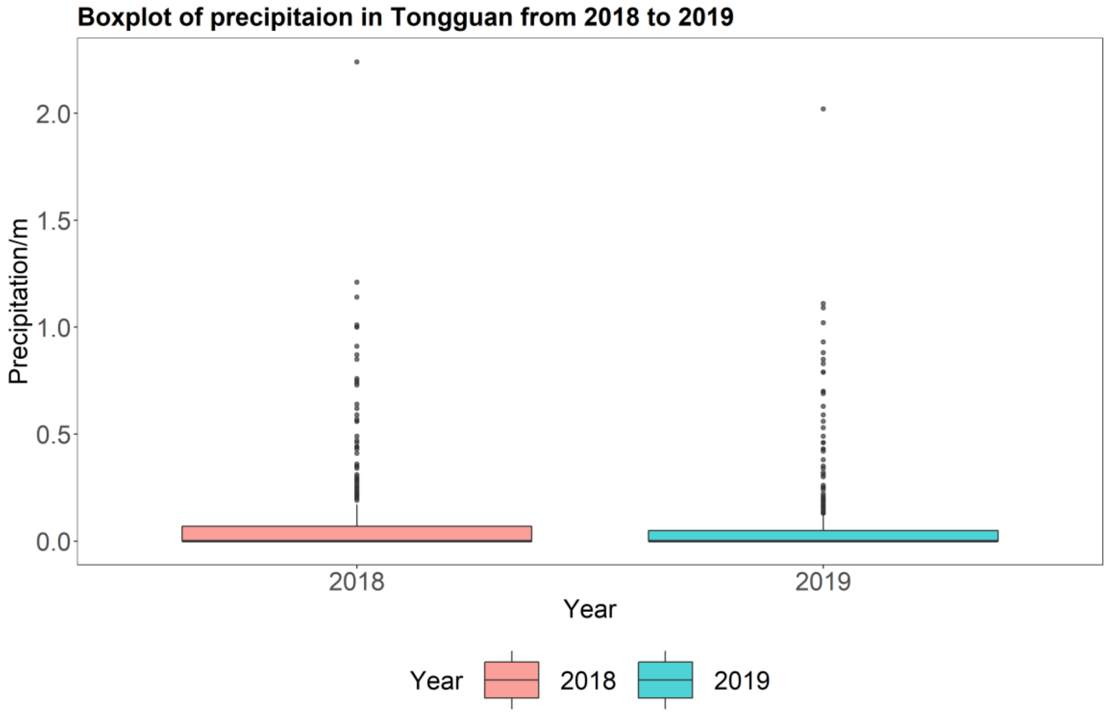
Assignment 04

1. Plotting with ggplot2

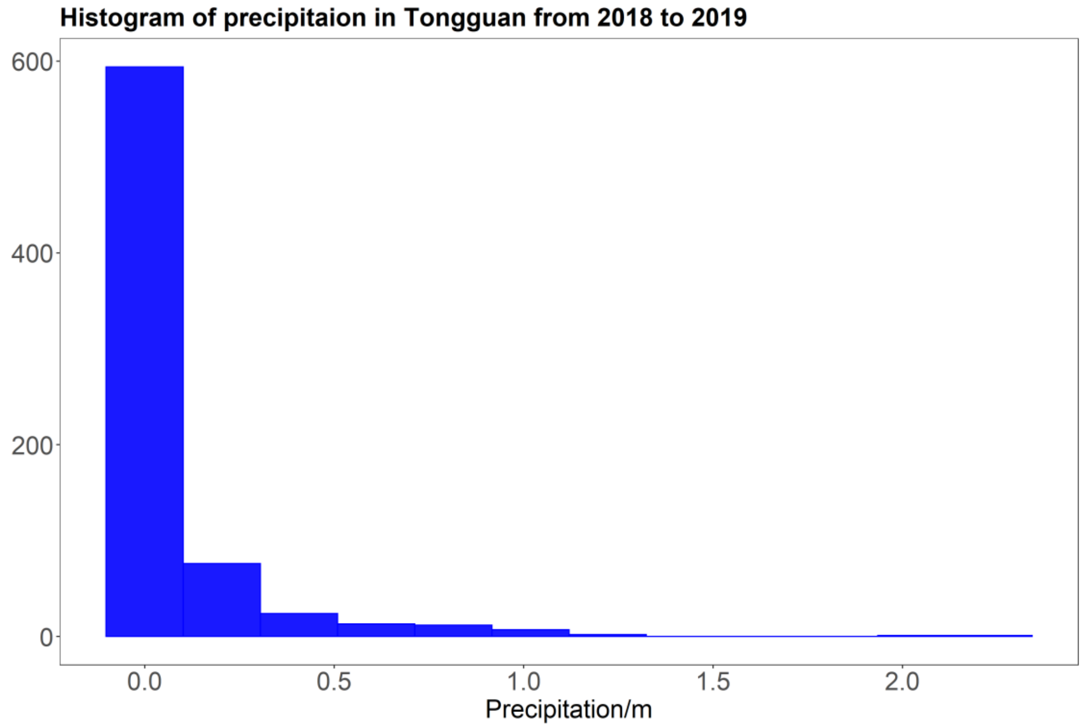
Use the precipitation data in Tongguan from 6/1/2017 to 2/29/2020

```r
#precipitaion data from 6/1/2017 to 2/29/2020 in Tongguan Country
install.packages("readxl")
library(readxl)
TG_rain <- read_excel("tongguan.xls", sheet="rain")
head(TG_rain)
```

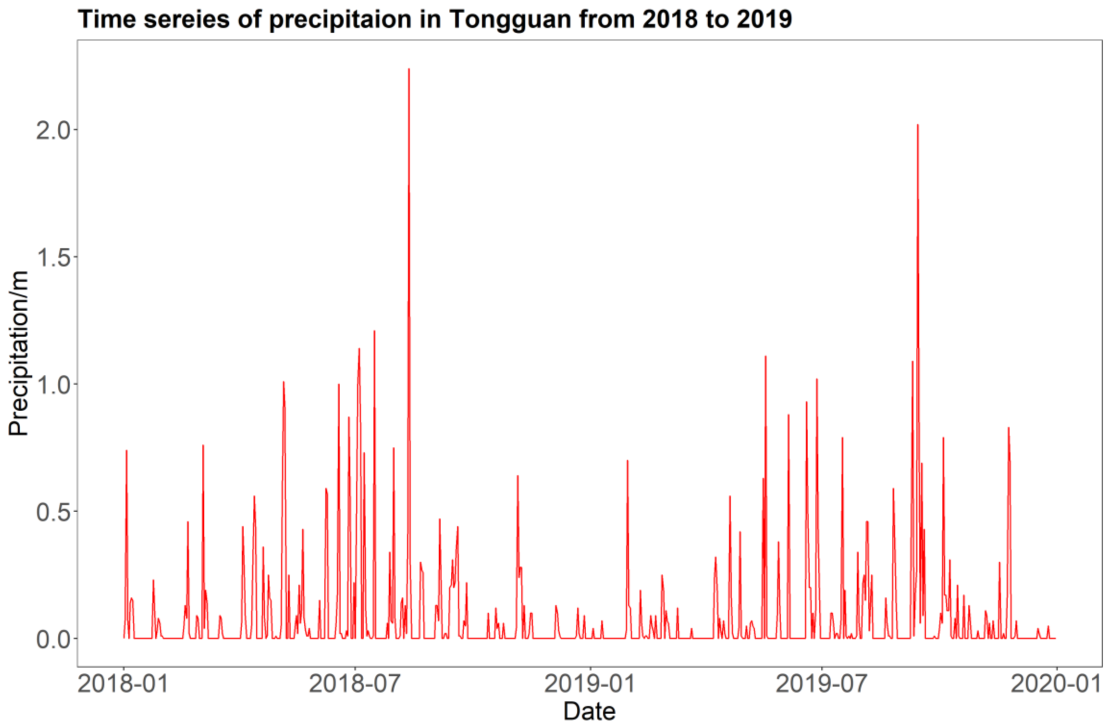
```r
#Boxplot of precipitaion in Tongguan from 2018 to 2019
TG_rain %>%
  filter(date >= "2018-01-01" & date < "2020-01-01") %>%
  mutate(year=substr(date, 1,4)) %>%
  ggplot(aes(x=year, y=precipitation,fill=year)) +
  geom_boxplot(alpha=0.7) +
  labs(title="Boxplot of precipitaion in Tongguan from 2018 to 2019", x="Year", y="Precipitation/m", fill="Year") +
  theme_bw() +
  theme(panel.grid.major =element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank()) +
  theme(plot.title=element_text(size=20, face="bold"),
        axis.text.x=element_text(size=20),
        axis.text.y=element_text(size=20),
        axis.title.x=element_text(size=20),
        axis.title.y=element_text(size=20),
        legend.position = "bottom",
        legend.text = element_text(size = 20),
        legend.title = element_text(size = 20),
        legend.key.size = unit(2,'cm'))
ggsave("boxplot.png")
```



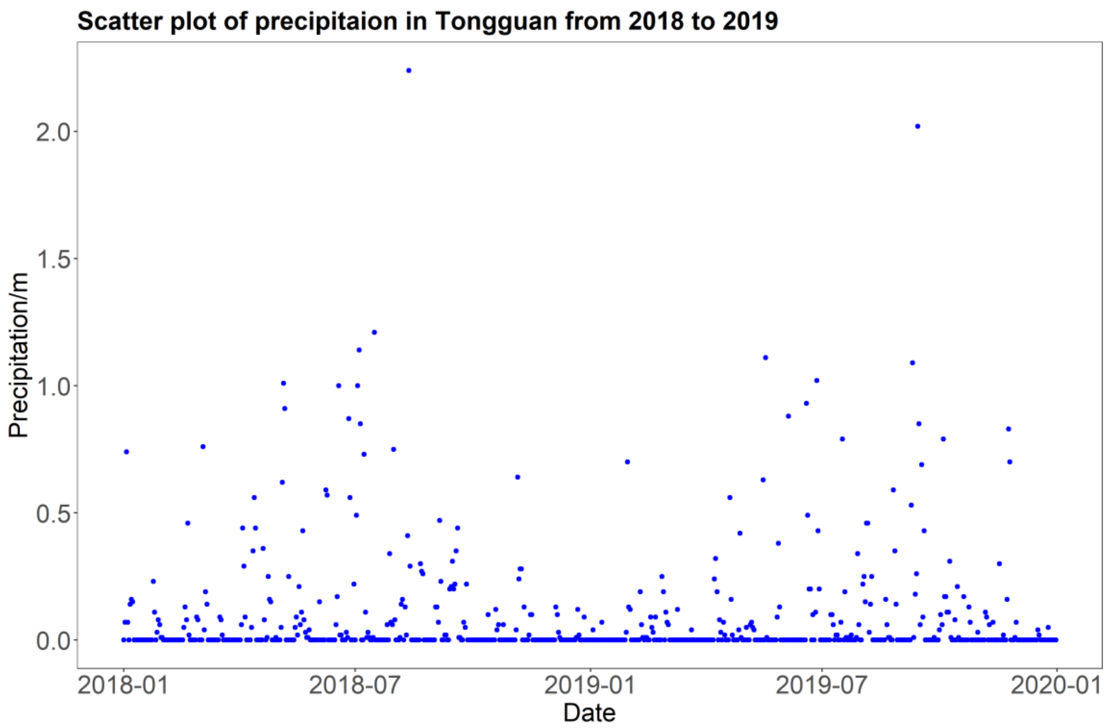
```r
#Histogram of precipitaion in Tongguan from 2018 to 2019
TG_rain %>%
  filter(date >= "2018-01-01" & date < "2020-01-01") %>%
  ggplot(aes(x=precipitation))+
  geom_histogram(bins=12, color="blue", fill="blue", alpha=0.9) +
  labs(title="Histogram of precipitaion in Tongguan from 2018 to 2019", x="Precipitation/m", y="") +
  theme_bw() +
  theme(panel.grid.major =element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank()) +
  theme(plot.title=element_text(size=20, face="bold"),
        axis.text.x=element_text(size=20),
        axis.text.y=element_text(size=20),
        axis.title.x=element_text(size=20),
        axis.title.y=element_text(size=20))
ggsave("histogram.png")
```




```r
#Time series of precipitaion in Tongguan from 2018 to 2019
# Apply the ts() function
Pre <- ts(TG_rain$precipitation, start=c(2018-01-01), frequency=12)

# Quick plot
plot(Pre, type="l")

TG_rain %>%
  filter(date >= "2018-01-01" & date < "2020-01-01") %>%
  ggplot(aes(x=date, y=precipitation))+
  geom_line(color = "red", size=0.5) +
  labs(title="Time sereies of precipitaion in Tongguan from 2018 to 2019", x="Date", y="Precipitation/m") +
  theme_bw() +
  theme(panel.grid.major =element_blank(),
        panel.grid.minor = element_blank(),
        panel.background = element_blank()) +
  theme(plot.title=element_text(size=20, face="bold"),
        axis.text.x=element_text(size=20),
        axis.text.y=element_text(size=20),
        axis.title.x=element_text(size=20),
        axis.title.y=element_text(size=20))
ggsave("time series.png")
```




```r
#Scatter plot of precipitaion in Tongguan from 2018 to 2019
TG_rain %>%
  filter(date >= "2018-01-01" & date < "2020-01-01") %>%
  ggplot(aes(x=date, y=precipitation)) +
  geom_point(color="blue") +
  labs(title="Scatter plot of precipitaion in Tongguan from 2018 to 2019", x="Date", y="Precipitation/m") +
  theme_bw() +
  theme(panel.grid.major =element_blank(),
              panel.grid.minor = element_blank(),
              panel.background = element_blank()) +
  theme(plot.title=element_text(size=20, face="bold"),
        axis.text.x=element_text(size=20),
        axis.text.y=element_text(size=20),
        axis.title.x=element_text(size=20),
        axis.title.y=element_text(size=20))
ggsave("Scatter plot.png")
```

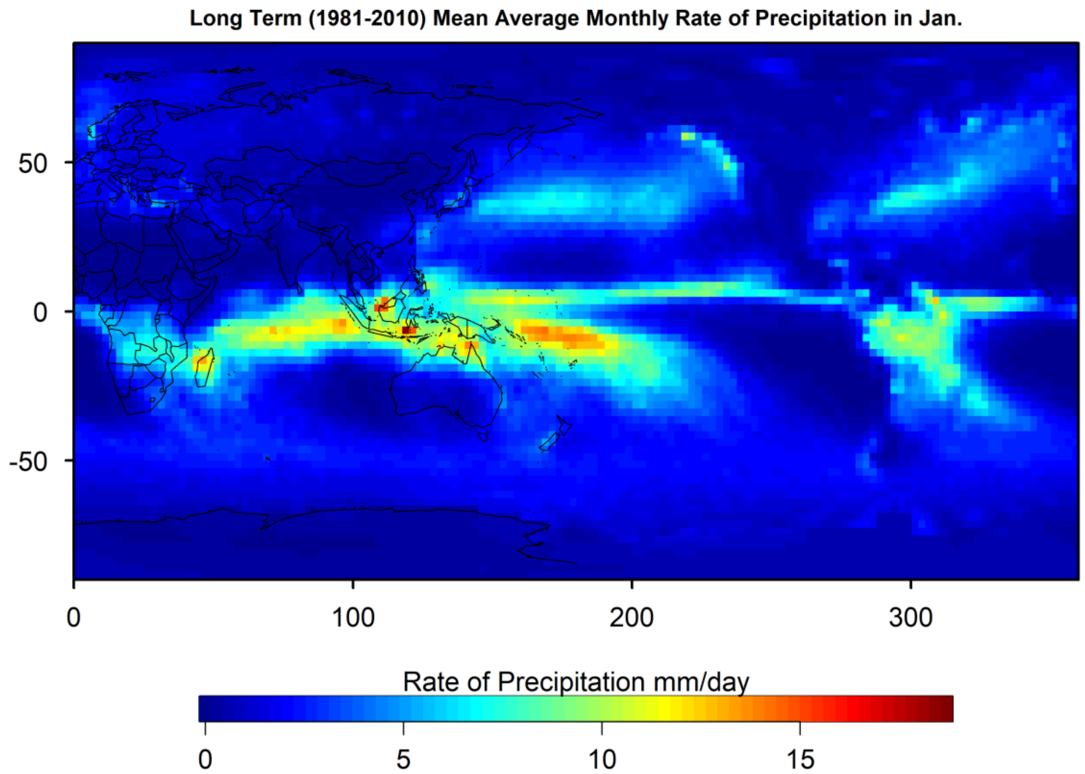


Download the long term (1981-2010) mean average monthly rate of precipitation from Jones (CRU) Air Temperature Anomalies Version 4: CRUTEM4 and move it to my working directory.

```r
# Open the NetCDF file
ex.nc     <- open.nc("precip.mon.ltm.nc")
# Print the variables and attributes
print.nc(ex.nc)

# Read the variables
# Lat
Lat        <- var.get.nc(ex.nc, "lat")
# Lon
Lon        <- var.get.nc(ex.nc, "lon")
# Long Term Mean Average Monthly Rate of Precipitation mm/day
precip_T      <- var.get.nc(ex.nc, "precip")

# Close the NetCDF file
close.nc(ex.nc)

# Original Lat is in decreasing order, we need to reverse it
Lat <- rev(Lat)

# Data transformation of precip_T_Jan
precip_T_Jan <- array(NA,dim=c(length(Lon), length(Lat)))
for(row in 1:length(Lat)){
  precip_T_Jan[,row] <- precip_T[, (length(Lat)+1-row),1 ]
}

image.plot(Lon, Lat,precip_T_Jan)
```

```r
# Plot
image.plot(Lon, Lat, precip_T_Jan,
           horizontal=T, useRaster=T,
           legend.shrink=0.75, axis.args=list(cex.axis = 1.25),
           legend.width=1, legend.mar=2,
           legend.args=list(text="Rate of Precipitation mm/day",cex=1.25),
           xlab='',ylab='',midpoint=T, axes=F, ann=F
)
title(xlab="",cex.lab=1.25,font.lab=2)
axis(1,at=pretty(Lon),tck=-0.015,lwd=2,cex.axis=1.25,font=1)
title(ylab="",cex.lab=1.25,font.lab=2)
axis(2,at=pretty(Lat),tck=-0.015,lwd=2,cex.axis=1.25,font=1,las=1)
title(main=paste("Long Term (1981-2010) Mean Average Monthly Rate of Precipitation in Jan."),
      cex.main=1,font.main=2)

# Add map
map('world',add=T,lwd=0.75,col="black")

# Add a box
box(lwd=2)

# Set the png format
png("Precip_T.png", width=8.5, height=6, units="in", res=400)

# Set margins on bottom, left, top, right
par(mar=c(4.5,3,2,1))
```

```r
# Plot
image.plot(Lon, Lat, precip_T_Jan,
           horizontal=T, useRaster=T,
           legend.shrink=0.75, axis.args=list(cex.axis = 1.25),
           legend.width=1, legend.mar=2,
           legend.args=list(text="Rate of Precipitation mm/day",cex=1.25),
           xlab='',ylab='',midpoint=T, axes=F, ann=F
)
title(xlab="",cex.lab=1.25,font.lab=2)
axis(1,at=pretty(Lon),tck=-0.015,lwd=2,cex.axis=1.25,font=1)
title(ylab="",cex.lab=1.25,font.lab=2)
axis(2,at=pretty(Lat),tck=-0.015,lwd=2,cex.axis=1.25,font=1,las=1)
title(main=paste("Long Term (1981-2010) Mean Average Monthly Rate of Precipitation in Jan."),
      cex.main=1,font.main=2)

# Add map
map('world',add=T,lwd=0.75,col="black")

# Add a box
box(lwd=2)

# Close the png file to save the file
dev.off()
```



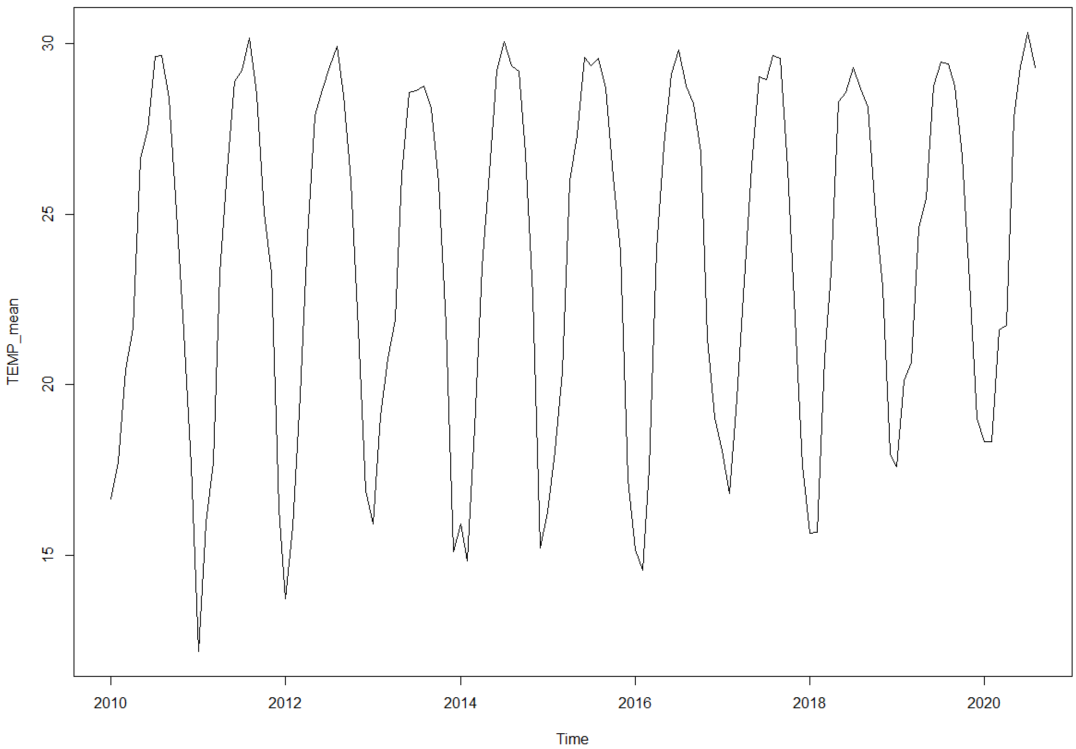
2. Analysis of the time series of monthly temperature

2.1 Construct a time series of monthly-averaged temperature from 2010 Jan. to 2020 Aug.

Read the data from 2281305.csv.

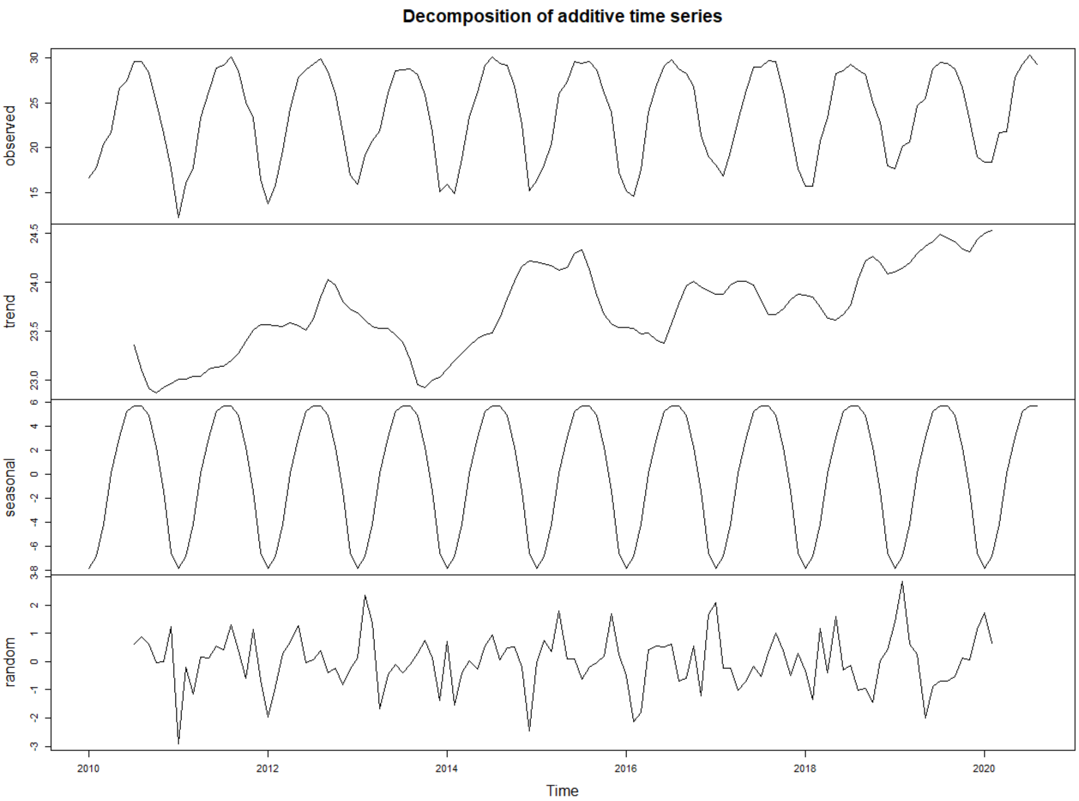
```r
Baoan_Data <- read.csv("2281305.csv", head= TRUE)
names(Baoan_Data)
head(Baoan_Data)
Met_Data <- Baoan_Data %>%
  mutate(temp=substr(TMP, 1,5), temp_quality=substr(TMP, 7,7)) %>%
  mutate(temp_new = as.numeric(temp), temp_quality_new = as.logical(as.numeric(temp_quality)))
# Handel missing values
Met_Data$temp_new[which(Met_Data$temp_new==9999)]<- NA
for(i in 1:length(Met_Data$temp_new)){
  if( is.na(Met_Data$temp_new[i])){
    Met_Data$temp_new[i] <- mean(Met_Data$temp_new[(i-2):(i+2)],na.rm=T )
  }
}
Met_Data$temp_quality_new[!which(Met_Data$temp_quality_new)]       <- NA
Met_Data1 <- Met_Data %>%
  mutate(temp_new=temp_new*0.1)
```

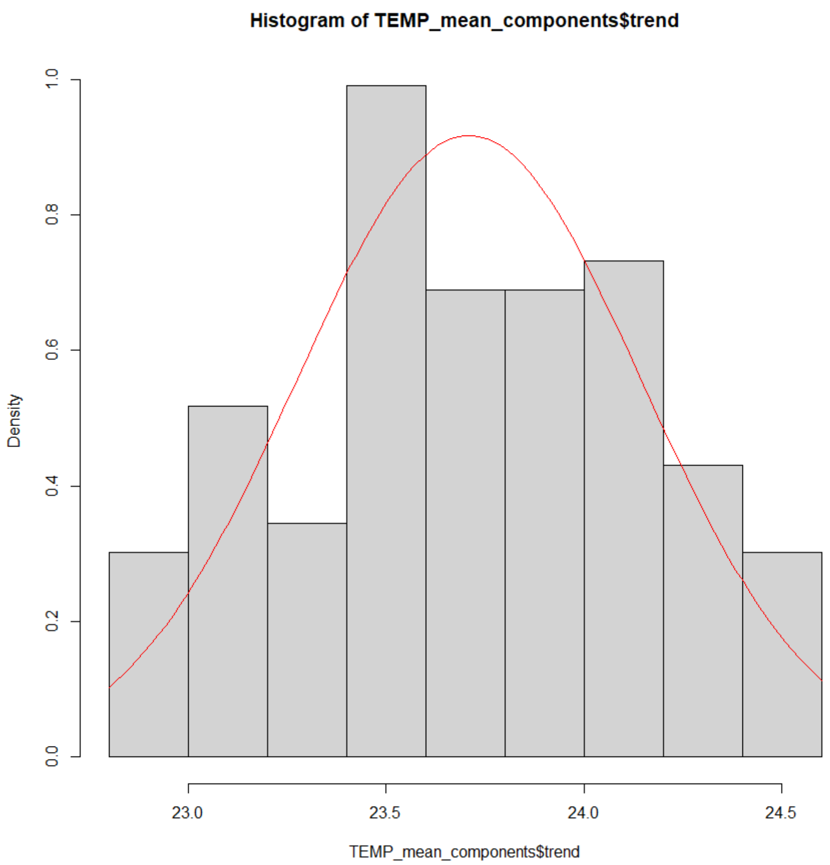
```r
#2.1 Construct a time series of monthly-averaged temperature from 2010 Jan. to 2020 Aug.
montly_mean <- c()
for (i in 2010:2020) {
  Thisyear_mean <- Met_Data1 %>%
    select(DATE, temp_new, temp_quality_new) %>%
    mutate(month=substr(DATE, 1, 7)) %>%
    mutate(year=substr(DATE,1,4)) %>%
    mutate(year2=as.numeric(year)) %>%
    filter(year2==i) %>%
    group_by(month) %>%
    summarise(mean=mean(temp_new, na.rm = T))
  montly_mean <- rbind(montly_mean, Thisyear_mean)
}

Monthly_mean1 <- montly_mean %>%
  filter(month <= "2020-08" & month >= "2010-01")
# Apply the ts() function
TEMP_mean <- ts(Monthly_mean1$mean, start=c(2010,1), frequency=12)

plot(TEMP_mean, type="l")
```



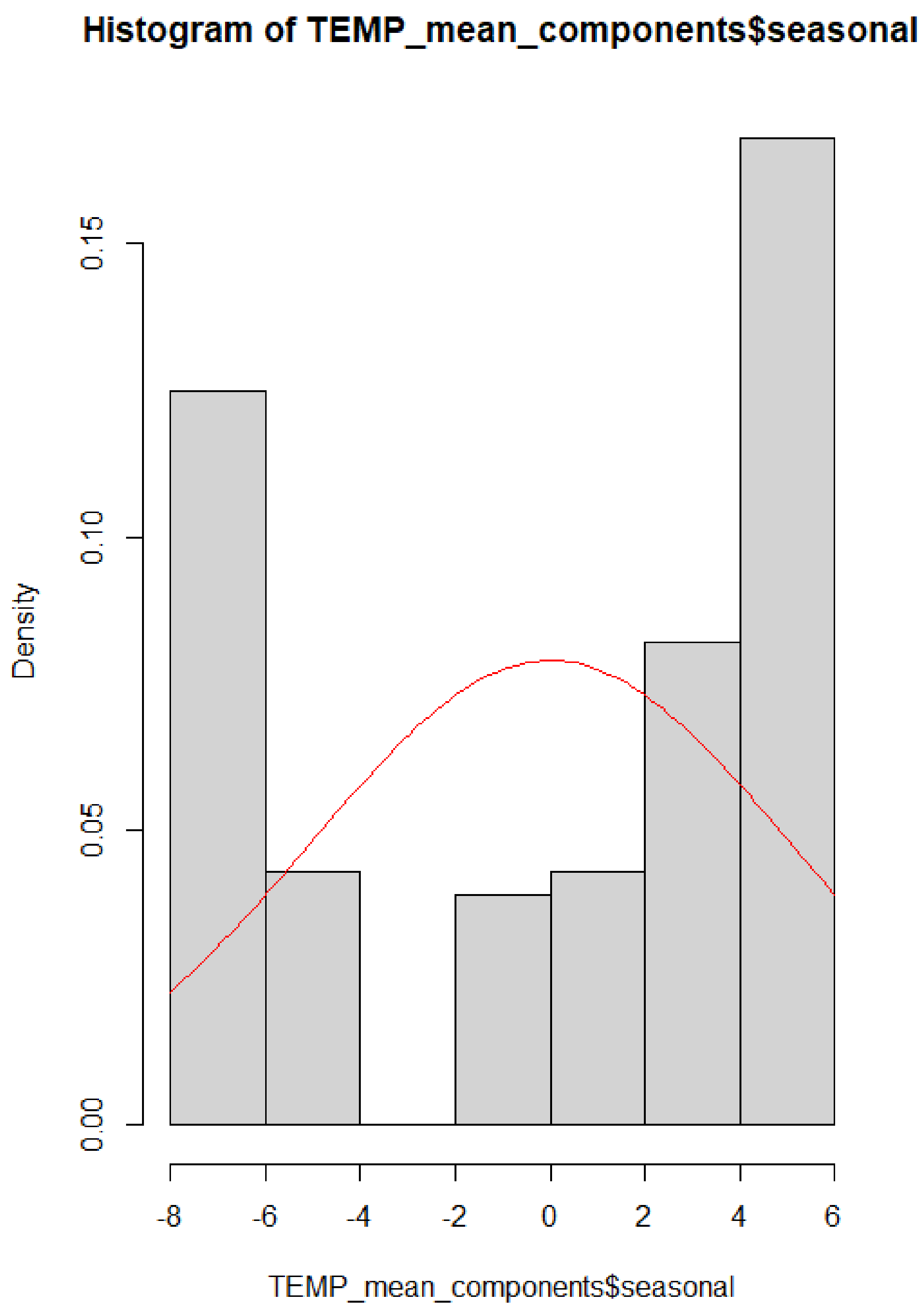
2.2 Decompose the time series into trend, seasonality, and error parts. Check whether the error part follows a white noise distribution.

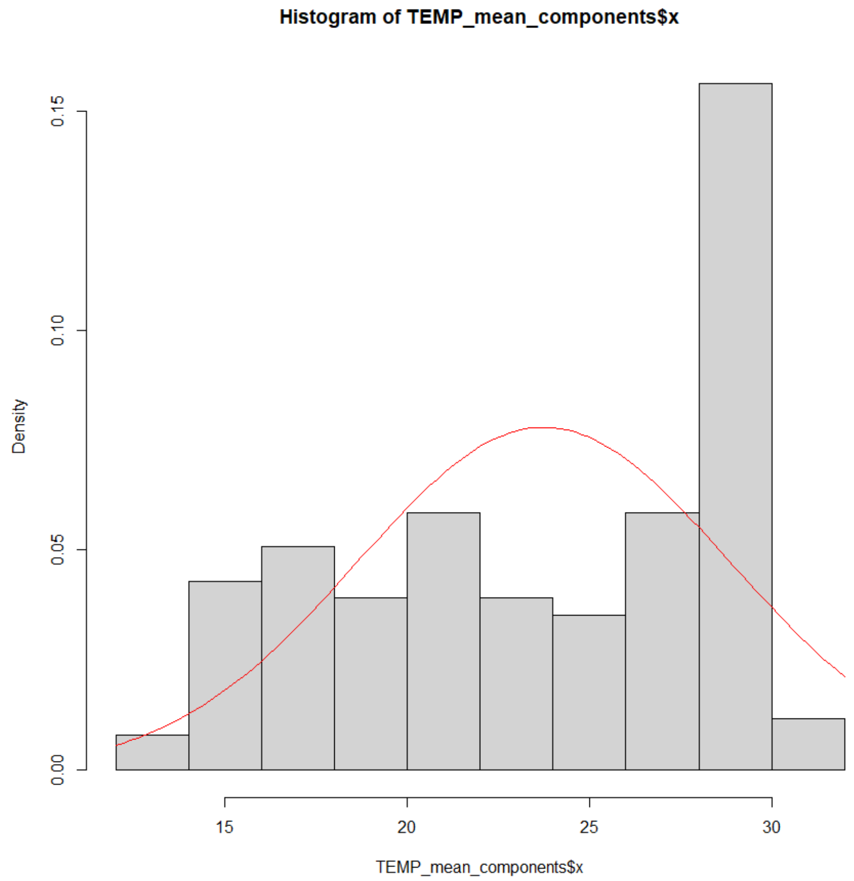
```
#2.2 Decompose the time series into trend, seasonality, and error parts. Check whether the error part follows a white noise distribution.
TEMP_mean_components <- decompose(TEMP_mean)
plot(TEMP_mean_components)
# Plot hist
hist(TEMP_mean_components$random, prob=TRUE)
# Add pdf
curve(dnorm(x, mean=mean(TEMP_mean_components$random,na.rm=T),
            sd=sd(TEMP_mean_components$random,na.rm=T)),
      add=TRUE, col="red")

# Plot hist
hist(TEMP_mean_components$trend, prob=TRUE)
# Add pdf
curve(dnorm(x, mean=mean(TEMP_mean_components$trend,na.rm=T),
            sd=sd(TEMP_mean_components$trend,na.rm=T)),
      add=TRUE, col="red")

# Plot hist
hist(TEMP_mean_components$seasonal, prob=TRUE)
# Add pdf
curve(dnorm(x, mean=mean(TEMP_mean_components$seasonal,na.rm=T),
            sd=sd(TEMP_mean_components$seasonal,na.rm=T)),
      add=TRUE, col="red")

# Plot hist
hist(TEMP_mean_components$x, prob=TRUE)
# Add pdf
curve(dnorm(x, mean=mean(TEMP_mean_components$x,na.rm=T),
            sd=sd(TEMP_mean_components$x,na.rm=T)),
      add=TRUE, col="red")
```

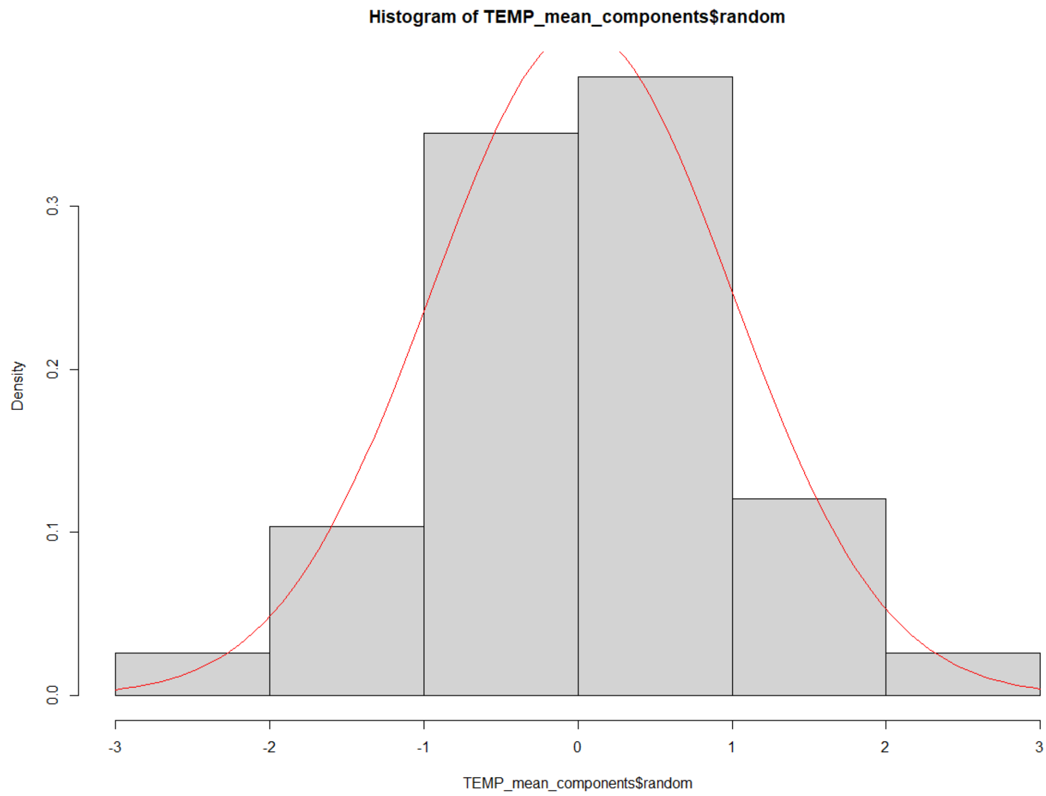




Histogram of TEMP_mean_components$trend
Density
0.0
0.2
0.4
0.6
0.8
1.0
23.0
23.5
24.0
24.5
TEMP_mean_components$trend


Histogram of TEMP_mean_components$seasonal
Density
0.00
0.05
0.10
0.15
-8
-6
-4
-2
0
2
4
6
TEMP_mean_components$seasonal


Histogram of TEMP_mean_components$x
Density
0.00
0.05
0.10
0.15
15
20
25
30
TEMP_mean_components$x


Histogram of TEMP_mean_components$random
Density
0.0
0.1
0.2
0.3
-3
-2
-1
0
1
2
3
TEMP_mean_components$random

Discussion: As you can see the distribution of the random is a Gaussian white noise, which is a particularly useful white noise series.

2.3 Fit an ARIMA(p,d,q) model to the time series. Describe the fitting process in details in your report.

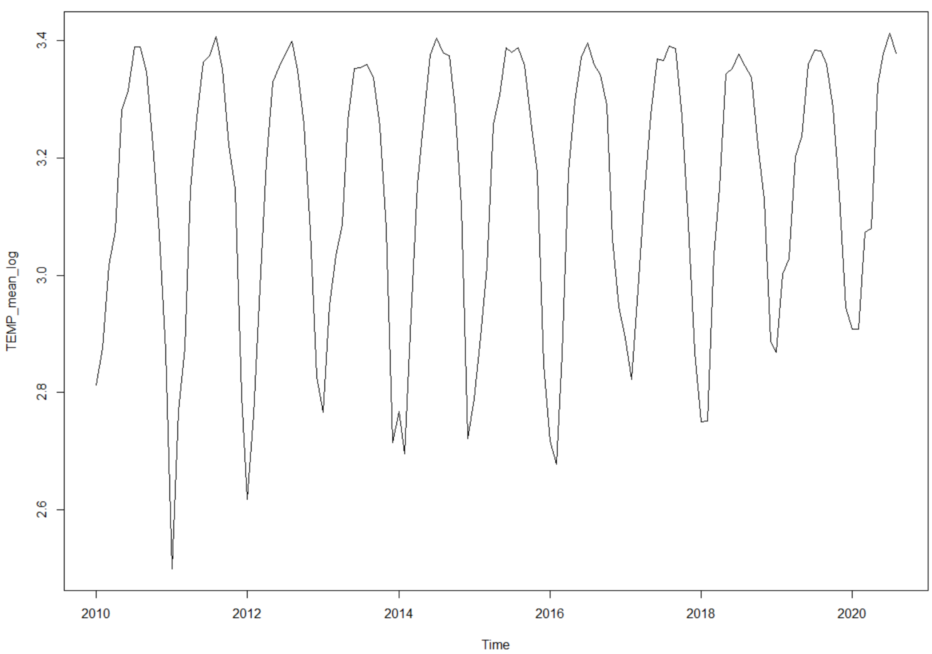
Step 1: Take log to the TEMP_mean time series.

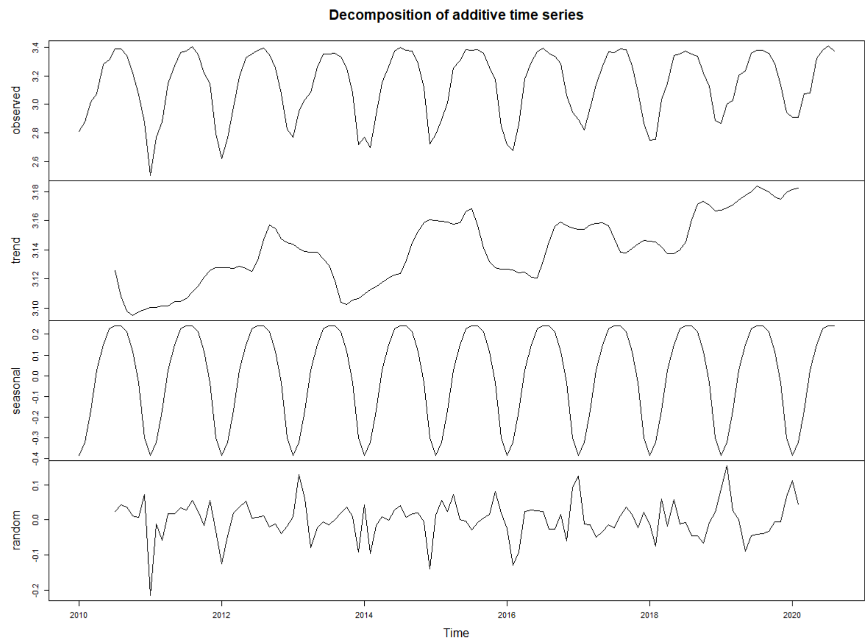
Step 2: Take the difference.

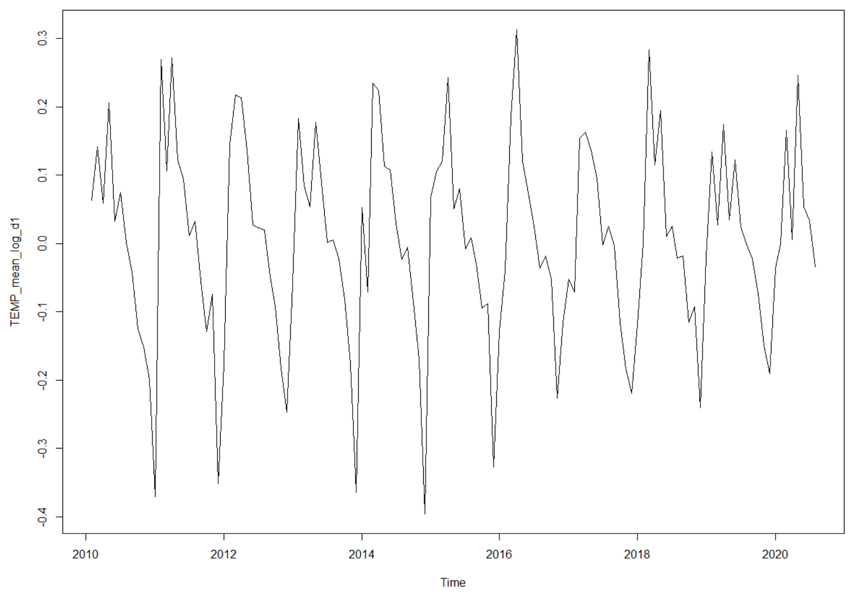
Step 3: Check acf and pacf.

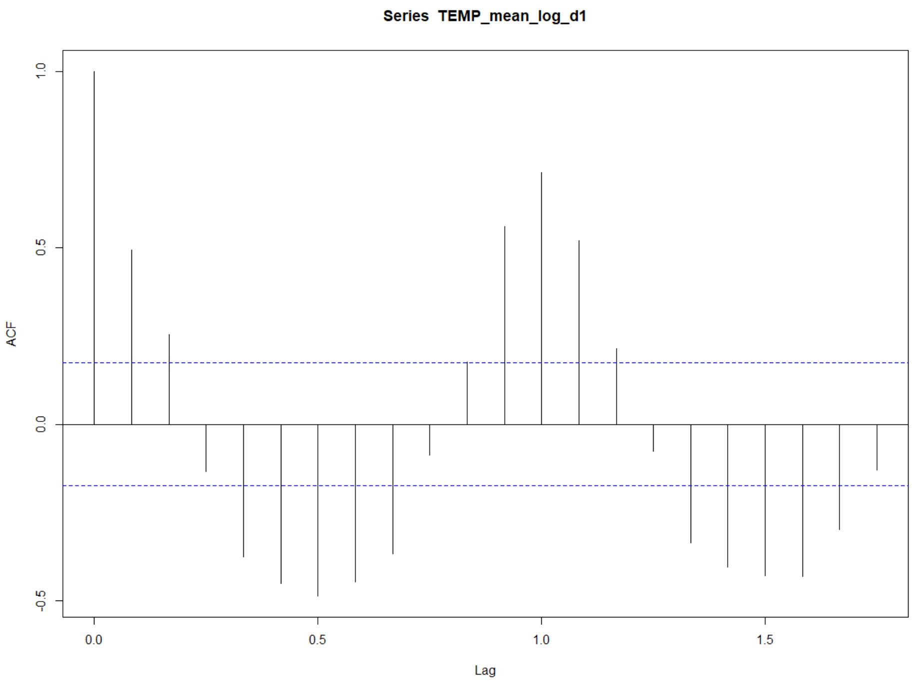
Step 4: Auto ARIMA fitting.

```
#2.3 Fit an ARIMA(p,d,q) model to the time series. Describe the fitting process in details in your report.
TEMP_mean_log <- log(TEMP_mean)
plot(TEMP_mean_log, type="l")
TEMP_mean_log_components <- decompose(TEMP_mean_log)
plot(TEMP_mean_log_components)

TEMP_mean_log_d1 <- diff(TEMP_mean_log)
plot(TEMP_mean_log_d1)

# Check acf and pacf
acf(TEMP_mean_log_d1)
pacf(TEMP_mean_log_d1)

model <- auto.arima(TEMP_mean_log)
model
```

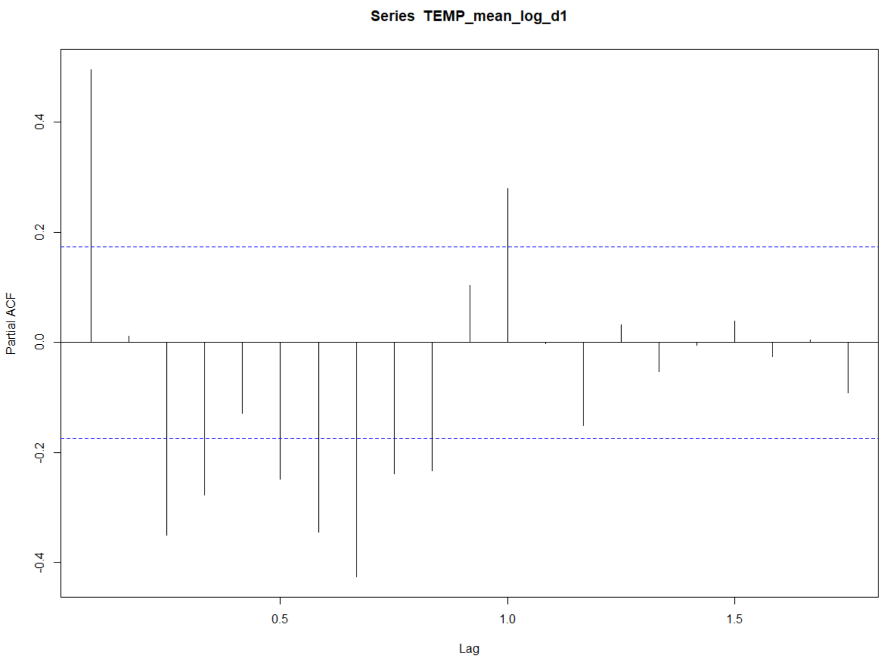



Decomposition of additive time series
observed
trend
seasonal
random
Time


TEMP_mean_log_d1
Time




```
> model <- auto.arima(TEMP_mean_log)
> model
Series: TEMP_mean_log
ARIMA(0,0,2)(0,1,1)[12] with drift

Coefficients:
         ma1     ma2     sma1  drift
      0.2495  0.1952  -0.8030  4e-04
s.e.  0.0897  0.1034   0.1153  2e-04

sigma^2 estimated as 0.003713:  log likelihood=155.8
AIC=-301.61   AICc=-301.06   BIC=-287.84
```

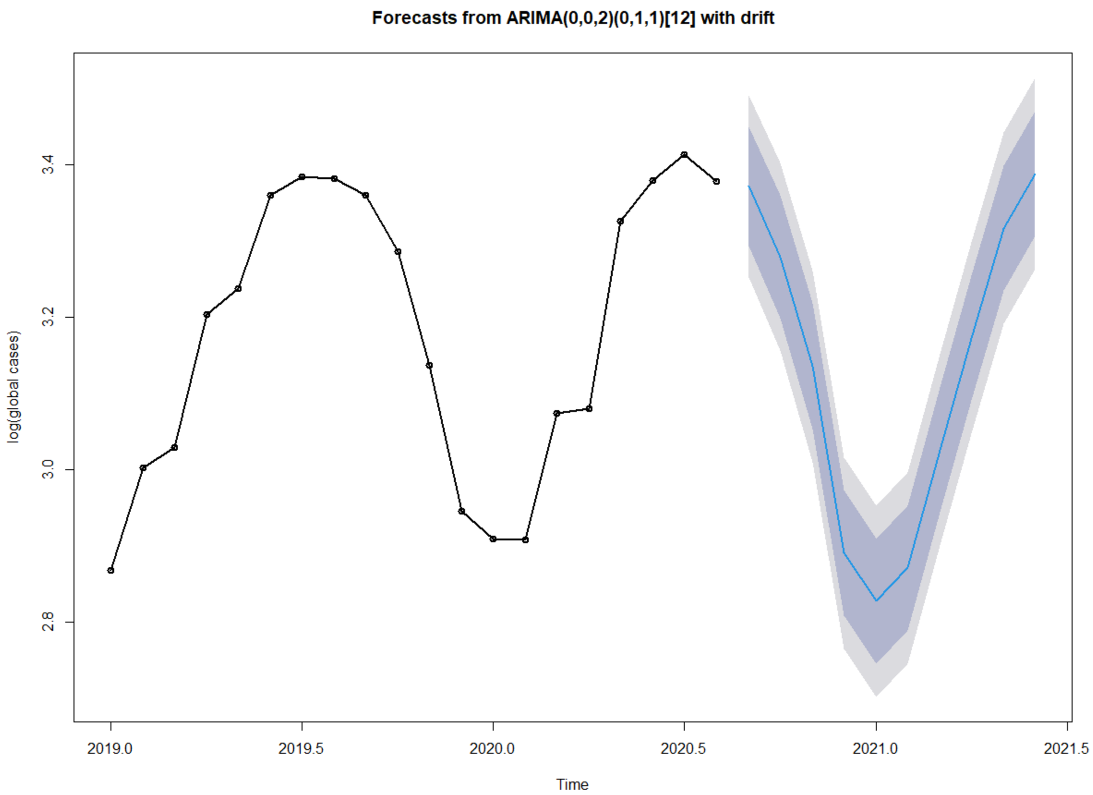


Discussion: The best ARIMA models are (0, 0, 2) and (0, 1, 1).

2.4 Predict monthly-averaged temperatures in 2020 Sep. and Oct. with the ARIMA

model from 2.3. The predictions will be evaluated against actual observations in those two months.

```
#2.4 Predict monthly-averaged temperatures in 2020 Sep. and Oct. with the ARIMA model from 2.3. The predictions will be eval
# Number of days to predict
days_forecast  <- 10
# Number of include in the plot
days_in_plot   <- 20
# Make predictions using the forecast() function
forecast_10days <- forecast(model, days_forecast)
# Plot
plot(forecast(model, days_forecast), include=days_in_plot,
     xlab="Time", ylab="log(global cases)",type="o",lwd=2)

# Sep 20
day_forward <- (2020-2010)*12+9-((2020-2010)*12+8) #(year-start_year)*12+month-((end_year-start_year)*12+end_month)
exp(forecast_10days$mean[day_forward])
exp(forecast_10days$lower[day_forward,1])
exp(forecast_10days$upper[day_forward,1])

# Oct 20
day_forward <- (2020-2010)*12+10-((2020-2010)*12+8)
exp(forecast_10days$mean[day_forward])
exp(forecast_10days$lower[day_forward,1])
exp(forecast_10days$upper[day_forward,1])
```




```
> # Sep 20
> day_forward <- (2020-2010)*12+9-((2020-2010)*12+8) #(year-start_year)*12+month-((end_year-start_year)*12+end_month)
> exp(forecast_10days$mean[day_forward])
[1] 29.11733
> exp(forecast_10days$lower[day_forward,1])
     80%
26.92535
> exp(forecast_10days$upper[day_forward,1])
     80%
31.48775
>
> # Oct 20
> day_forward <- (2020-2010)*12+10-((2020-2010)*12+8)
> exp(forecast_10days$mean[day_forward])
[1] 26.54346
> exp(forecast_10days$lower[day_forward,1])
     80%
24.48642
> exp(forecast_10days$upper[day_forward,1])
    80%
28.7733
```

Discussion: The true value for 2020 Sep is 29.45206. The estimated mean in 2020 Sep is 29.11733.

Further Discussion: I have a question about the data preprocessing. Above, I have taken log calculation and the difference to make the data smooth and stationary based on the lab 04. But, is the preprocessing necessary? According to 2.1 and 2.2, the original data seem to be stationary without preprocessing. So, I do the ARIMA(p,d,q) model and estimation with the original data again, and compare the result.

```
#青助教解释一下，数据是否需要进行前处理的原因，我都做了结果，但是感觉结果差别不大，是不需要做吗？？
model1 <- auto.arima(TEMP_mean)
model1

# Number of days to predict
days_forecast  <- 10
# Number of include in the plot
days_in_plot   <- 20
# Make predictions using the forecast() function
forecast_10days <- forecast(model1, days_forecast)
# Plot
plot(forecast(model1, days_forecast), include=days_in_plot,
     xlab="Time", ylab="global cases",type="o",lwd=2)

# Sep 20
day_forward <- (2020-2010)*12+9-((2020-2010)*12+8) #(year-start_year)*12+month-((end_year-start_year)*12+end_month)
forecast_10days$mean[day_forward]
forecast_10days$lower[day_forward,1]
forecast_10days$upper[day_forward,1]

# Oct 20
day_forward <- (2020-2010)*12+10-((2020-2010)*12+8)
forecast_10days$mean[day_forward]
forecast_10days$lower[day_forward,1]
forecast_10days$upper[day_forward,1]
```

```
> model1
Series: TEMP_mean
ARIMA(0,0,2)(1,1,1)[12] with drift

Coefficients:
         ma1     ma2     sar1     sma1   drift
      0.2090  0.1880  -0.0308  -0.8664  0.0085
s.e.  0.0902  0.1153   0.1684   0.2391  0.0037

sigma^2 estimated as 1.236:  log likelihood=-182.72
AIC=377.44   AICc=378.21   BIC=393.96
> # Number of days to predict
> days_forecast  <- 10
> # Number of include in the plot
> days_in_plot   <- 20
> # Make predictions using the forecast() function
> forecast_10days <- forecast(model1, days_forecast)
> # Plot
> plot(forecast(model1, days_forecast), include=days_in_plot,
+      xlab="Time", ylab="global cases",type="o",lwd=2)
>
> # Sep 20
> day_forward <- (2020-2010)*12+9-((2020-2010)*12+8) #(year-start_year)*12+month-((end_year-start_year)*12+end_month)
> forecast_10days$mean[day_forward]
[1] 29.04313
> forecast_10days$lower[day_forward,1]
     80%
27.60776
> forecast_10days$upper[day_forward,1]
     80%
30.4785
>
> # Oct 20
> day_forward <- (2020-2010)*12+10-((2020-2010)*12+8)
> forecast_10days$mean[day_forward]
[1] 26.40155
> forecast_10days$lower[day_forward,1]
     80%
24.93515
> forecast_10days$upper[day_forward,1]
     80%
27.86794
```

The best ARIMA models are (0, 0, 2) and (1, 1, 1). The true value for 2020 Sep is 29.45206. The estimated mean here without preprocessing in 2020 Sep is 29.04313.

The estimated mean here with preprocessing in 2020 Sep is 29.11733. The results are similar. Why?