STAT6571 Project 2

Time series analysis of Temperature data in Toronto since 1900 year

201992624 Inwook Back

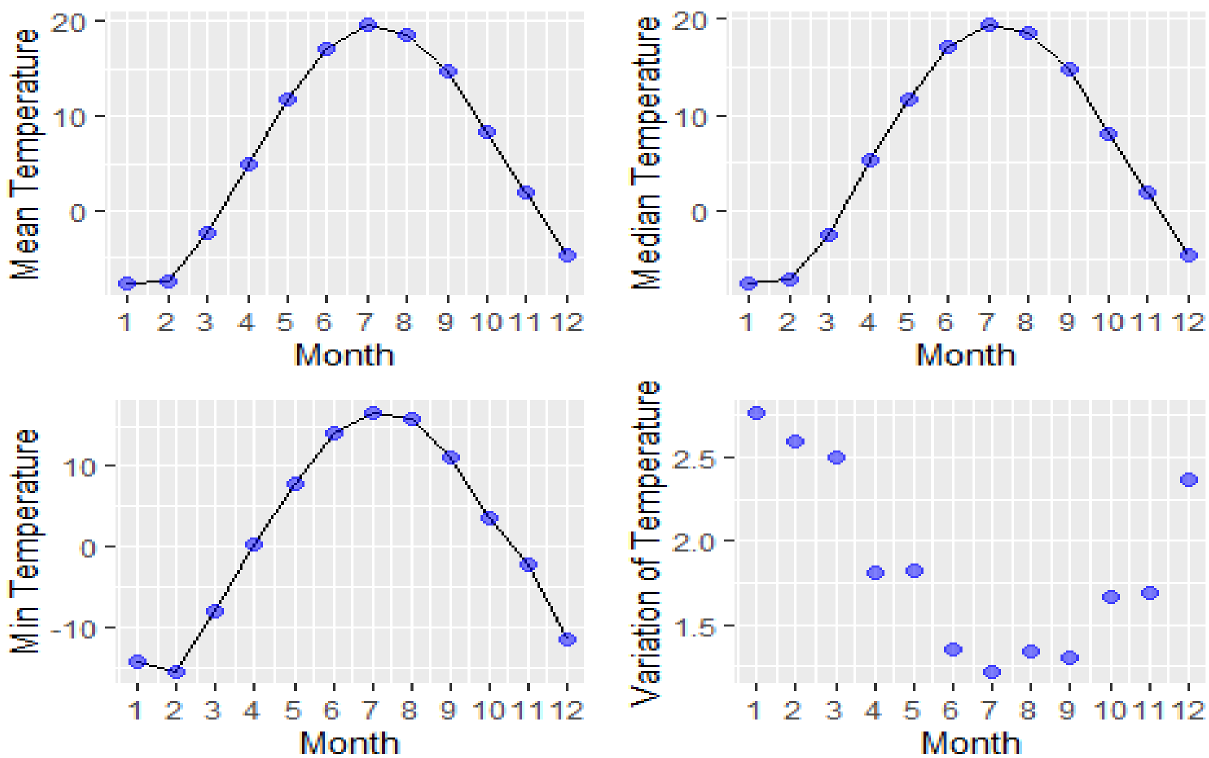
Tables

```r
library('ggplot2') # visualization
library('ggthemes') # visualization
library('scales') # visualization
library('grid') # visualisation
library('gridExtra') # visualisation
library('corrplot') # visualisation
library('ggrepel') # visualisation
library('RColorBrewer') # visualisation
library('ggridges') # visualisation

library('data.table') # data manipulation
library('dplyr') # data manipulation
library('tibble') # data wrangling
library('tidyr') # data wrangling
library('stringr') # string

library('lubridate') # date and time
library('forecast') # time series analysis

setwd("C:\\Users\\17096\\Desktop\\Data Science\\Nutrogena")


# Data Loading


sub <- fread("sub.csv") %>%

  mutate(dt = ymd(dt),
         wday = wday(dt, label = T),
         year = year(dt),
         month = month(dt)) %>%

  filter(year >= 1900) %>%

    as.tibble()



# “tp” is the dataset I will use from now



tp <- sub %>%
  filter(city == "Toronto") %>%
  mutate(diff = target - lag(target))
```

II. EDA by plot

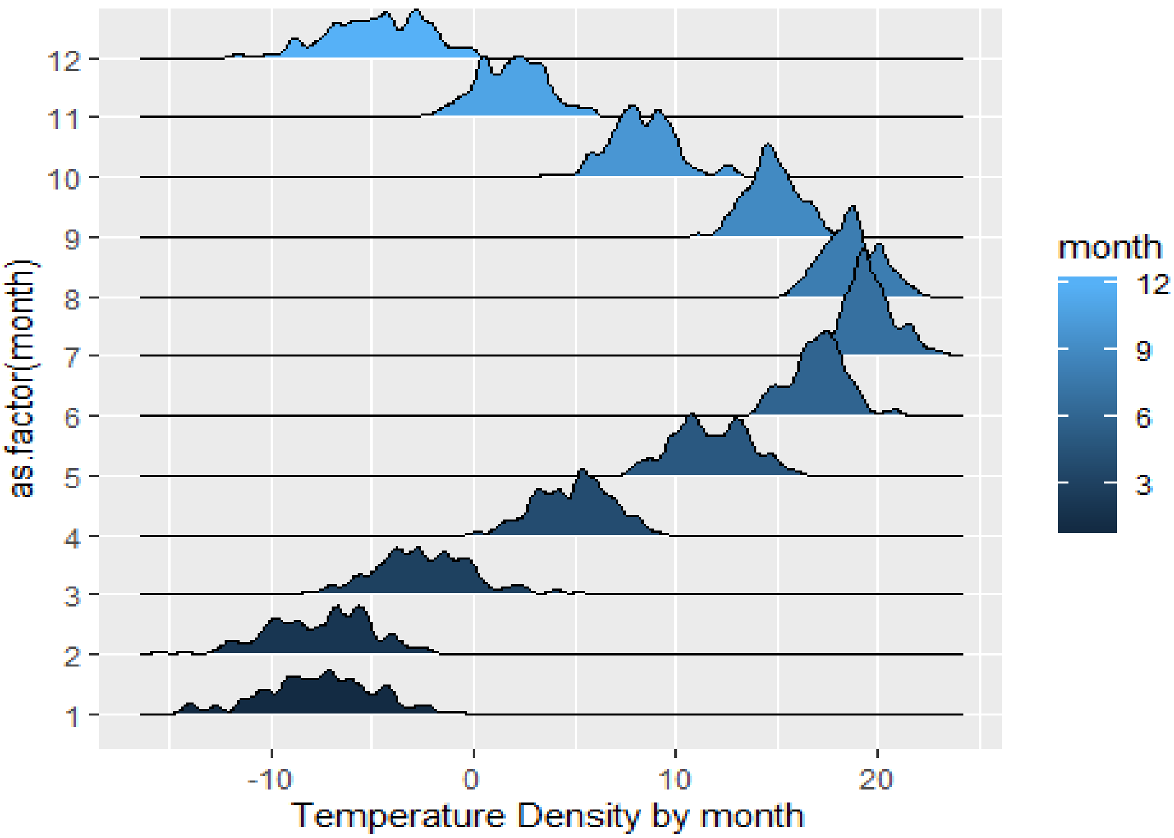
```
p1 <- tp %>%
  group_by(month) %>%
  summarise(mean = mean(target, na.rm = T),
            median = median(target, na.rm = T),
            max = max(target, na.rm = T),
            min = min(target, na.rm = T),
            sd = sd(target, na.rm = T)) %>%
  ggplot(aes(as.integer(month), mean)) +
  geom_line() +
  geom_point(size = 2, alpha = 0.5, color = 'blue') +
  scale_x_continuous(breaks = 1:12) +
  scale_colour_hue() +
  labs(x = "Month", y = "Mean Temperature")

p2 <- tp %>%
  group_by(month) %>%
  summarise(mean = mean(target, na.rm = T),
            median = median(target, na.rm = T),
            max = max(target, na.rm = T),
            min = min(target, na.rm = T),
            sd = sd(target, na.rm = T)) %>%
  ggplot(aes(as.integer(month), median)) +
  geom_line() +
  geom_point(size = 2, color = 'blue', alpha = 0.5) +
  scale_x_continuous(breaks = 1:12) +
  scale_colour_hue() +
  labs(x = "Month", y = "Median Temperature")

p3 <- tp %>%
  group_by(month) %>%
  summarise(mean = mean(target, na.rm = T),
            median = median(target, na.rm = T),
            max = max(target, na.rm = T),
            min = min(target, na.rm = T),
            sd = sd(target, na.rm = T)) %>%
  ggplot(aes(as.integer(month), min)) +
  geom_line() +
  geom_point(size = 2, alpha = 0.5, color = 'blue') +
  scale_x_continuous(breaks = 1:12) +
  scale_colour_hue() +
  labs(x = "Month", y = "Min Temperature")

p4 <- tp %>%
  group_by(month) %>%
  summarise(mean = mean(target, na.rm = T),
            median = median(target, na.rm = T),
```

```
            max = max(target, na.rm = T),
            min = min(target, na.rm = T),
            sd = sd(target, na.rm = T)
  ) %>%
  ggplot(aes(as.integer(month), sd)) +
  geom_point(size = 2, alpha = 0.5, color = 'blue') +
  scale_x_continuous(breaks = 1:12) +
  labs(x = "Month", y = "Variation of Temperature") +
  theme(legend.position = 'none')

grid.arrange(p1,p2,p3,p4, layout_matrix = matrix(1:4, nrow = 2, byrow = T))
```




```
# It is obvious that temperature has a cycle along with months.
# The plots above clearly shows this.



## Density plot by year


plt_density <- function(df, target){


  p <- df %>%
```

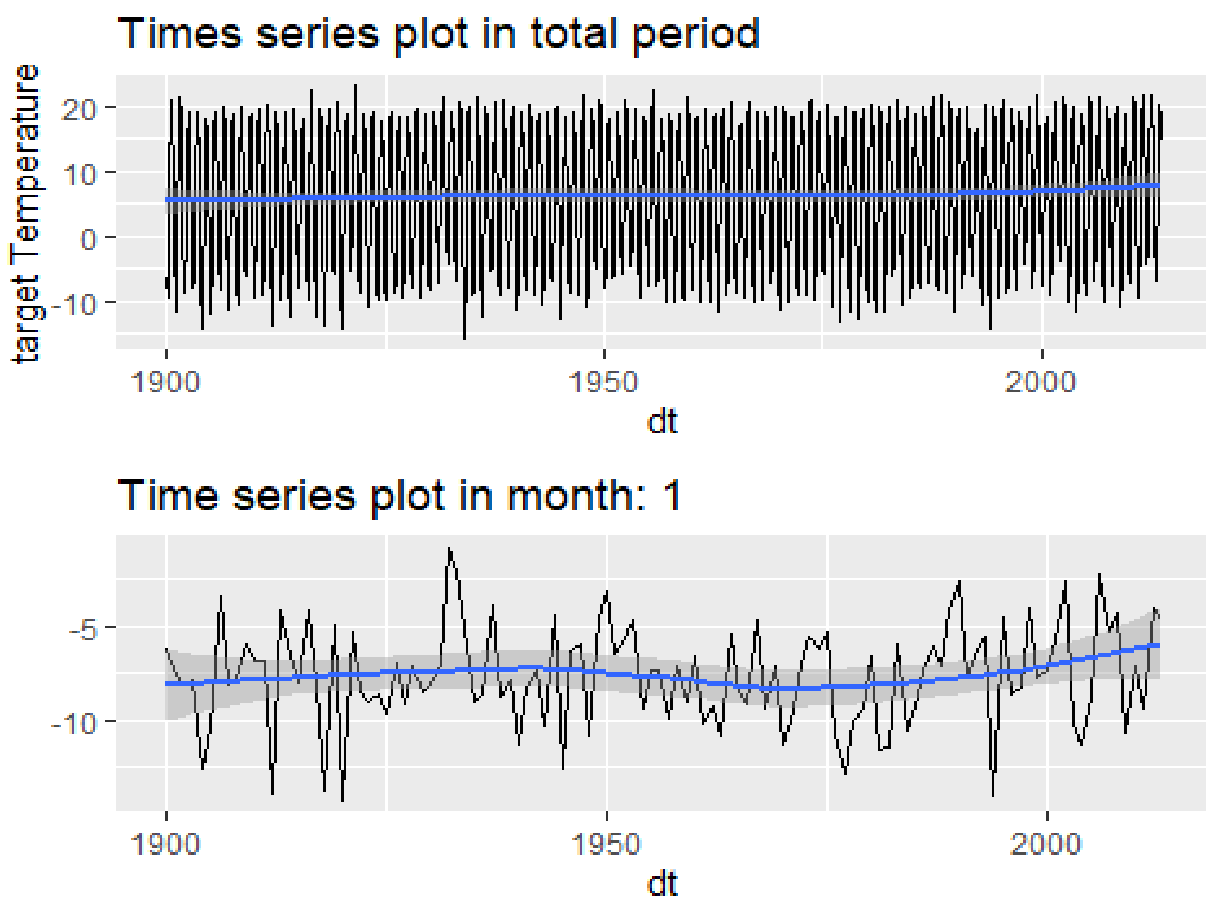
```
    ggplot(aes(target, as.factor(month), fill = month)) +
    geom_density_ridges(bandwidth = 0.3) +
    labs(x = 'Temperature Density by month')

  return(p)
}

plt_density(tp, target)
```




```
# From the density plots, temperature distributions are very different by mon
th. And in winter season, the dispersion is much more variable than the summe
r season. So, prediction for winter season's temperature should be harder tha
n the summer season.
```

```r
## Time Series plot

plt_ts <- function(df, m, series1, series2, option){

  p1 <- df %>%
    ggplot(aes(dt, !!sym(series1))) +
    geom_line() +
    geom_smooth(method = 'loess') +
    labs(y = str_c(series1, ' Temperature')) +
    ggtitle('Times series plot in total period')

  p2 <- df %>%
    filter(month == m) %>%
    ggplot(aes(dt, !!sym(series1))) +
    geom_line() +
    geom_smooth(method = 'loess') +
    labs(y = '') +
    ggtitle(str_c('Time series plot in month: ', m))

  p3 <- df %>%
    ggplot(aes(dt, !!sym(series2))) +
    geom_line() +
    geom_smooth(method = 'loess') +
    labs(y = str_c(series2, ' Temperature')) +
    ggtitle('Times series plot in total period')

  p4 <- df %>%
    filter(month == m) %>%
    ggplot(aes(dt, !!sym(series2))) +
    geom_line() +
    geom_smooth(method = 'loess') +
    labs(y = '') +
    ggtitle(str_c('Time series plot in month: ', m))
```

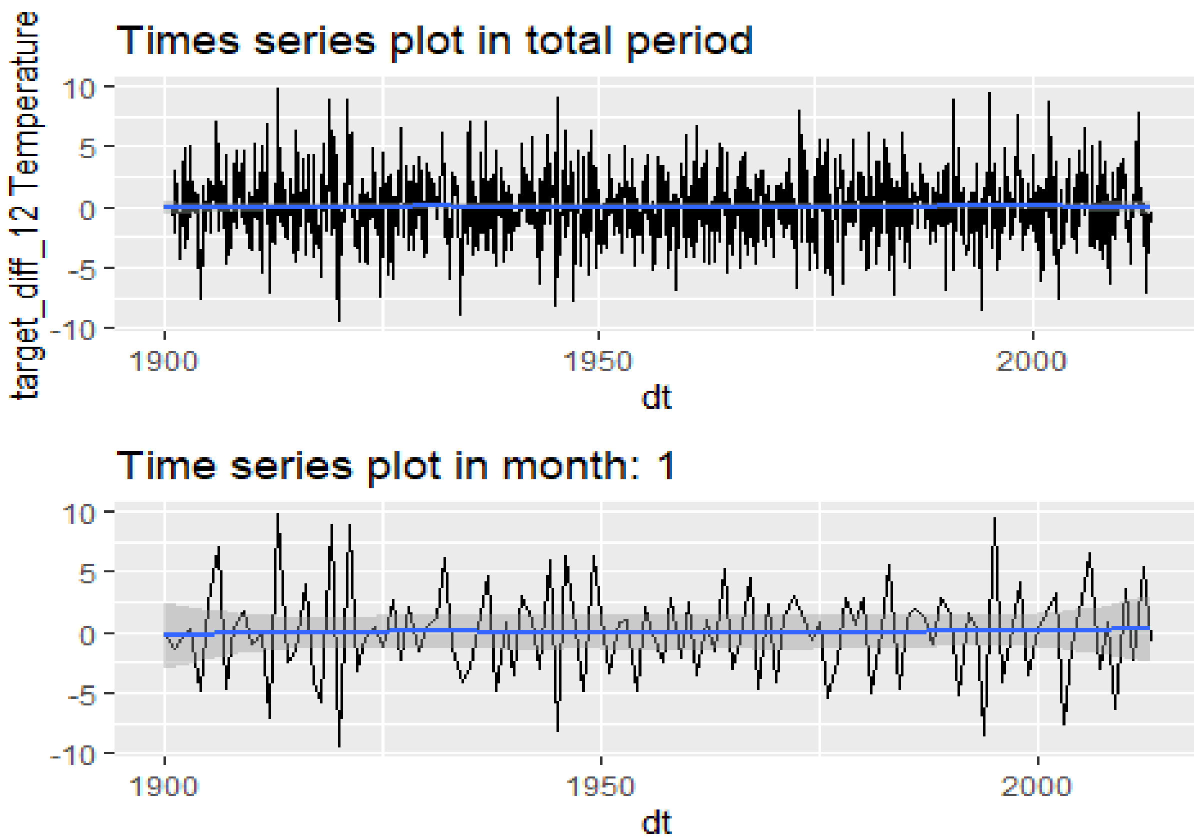
```
  if(option == F){
    grid.arrange(p1, p2, layout_matrix = matrix(c(1,2), nrow = 2, byrow = F))
    }else{
    grid.arrange(p1, p2, p3, p4, layout_matrix = matrix(c(1,2,3,4), nrow = 4,
 byrow = F))
  }
}

plt_ts(tp, 1, 'target', 'target', F)
```




```
# From the time plots, there looks seasonality without trend. It should be 12
 months cycle, but I can check this from periodogram later.
```

III. Modeling

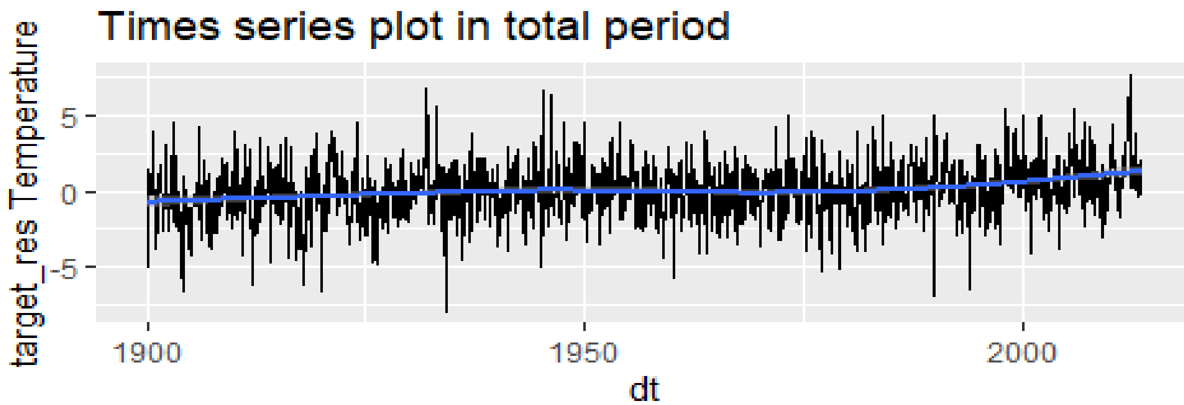
```
## First, I will use 12th order differencing, or I can try using "Month" as a
 covariate to address the seasonality.

tp$target_diff_12 <- c(rep(0, 12), diff(tp$target, 12))

par(mfrow = c(2,2))
plt_ts(tp, 1, 'target_diff_12', 'target_diff_12', option = F)
```

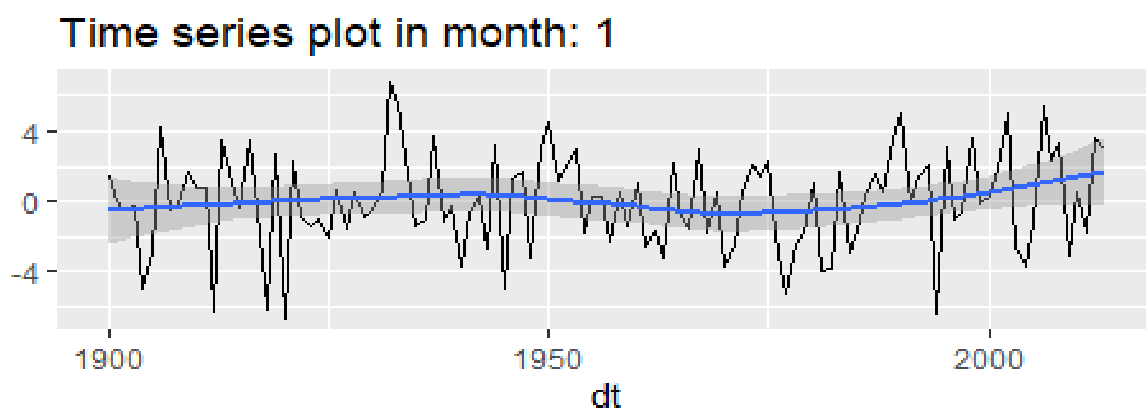



```
# From the plot, 12th order differencing seems to remove the seasonality. And
 I will try "Month" covariate and investigate its residuals as target series,
 too.

fit_linear <- lm(target ~ as.factor(month), data = tp)
summary(fit_linear)

##
## Call:
```

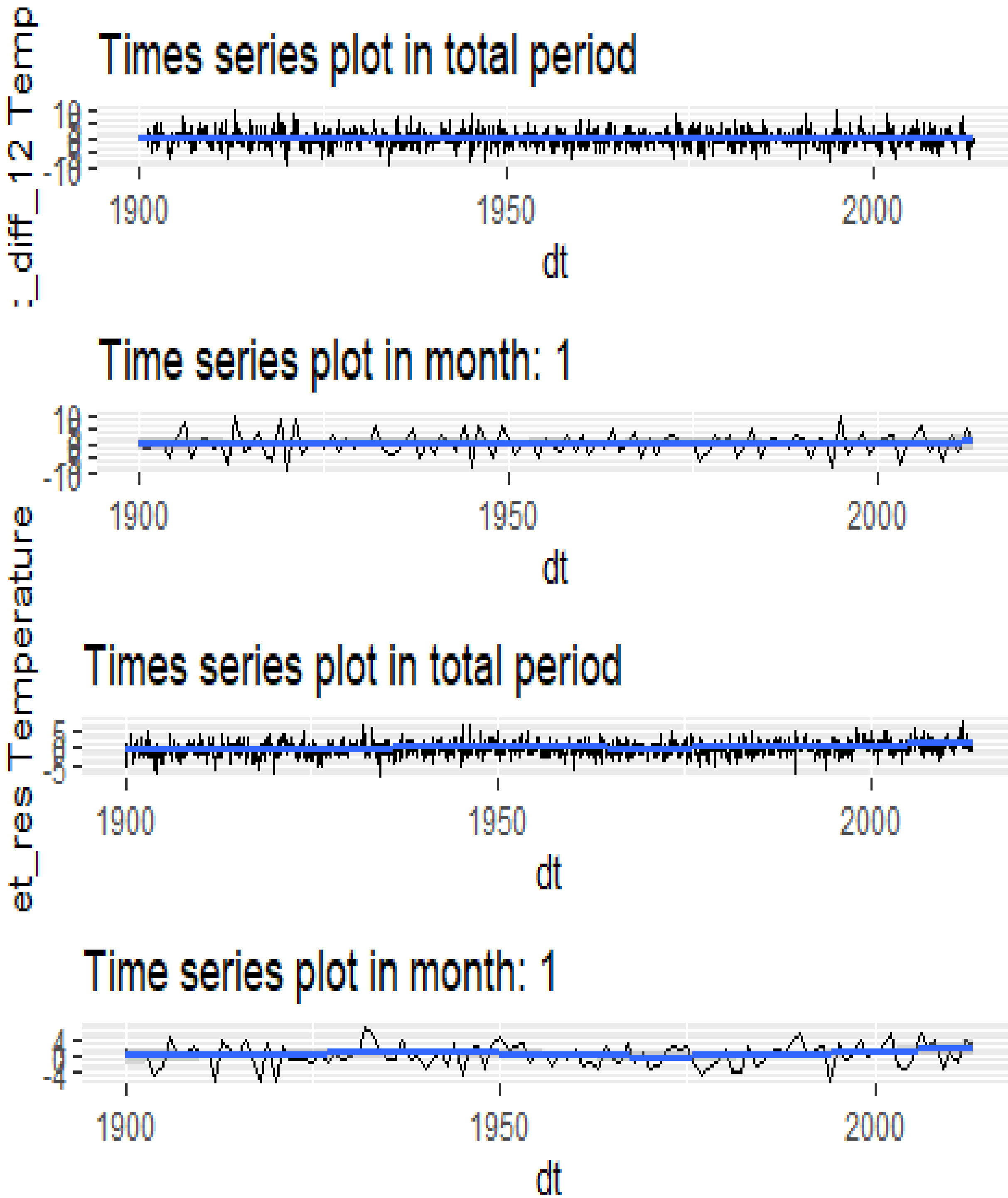
```
## lm(formula = target ~ as.factor(month), data = tp)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.9144 -1.2269 -0.0099  1.2391  7.6320
##
## Coefficients:
##                     Estimate Std. Error t value Pr(>|t|)
## (Intercept)         -7.60966    0.18191 -41.832   <2e-16 ***
## as.factor(month)2    0.02207    0.25726   0.086    0.932
## as.factor(month)3    5.21563    0.25726  20.274   <2e-16 ***
## as.factor(month)4   12.58577    0.25726  48.923   <2e-16 ***
## as.factor(month)5   19.36116    0.25726  75.260   <2e-16 ***
## as.factor(month)6   24.64132    0.25726  95.785   <2e-16 ***
## as.factor(month)7   27.22138    0.25726 105.814   <2e-16 ***
## as.factor(month)8   26.28872    0.25726 102.188   <2e-16 ***
## as.factor(month)9   22.38557    0.25726  87.016   <2e-16 ***
## as.factor(month)10  15.97238    0.25783  61.950   <2e-16 ***
## as.factor(month)11   9.49928    0.25783  36.844   <2e-16 ***
## as.factor(month)12   2.85899    0.25783  11.089   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.942 on 1353 degrees of freedom
## Multiple R-squared:  0.9627, Adjusted R-squared:  0.9624
## F-statistic:  3174 on 11 and 1353 DF,  p-value: < 2.2e-16
```

```r
# As I saw from the plots, month variable is very significant and its R-squared is 96%, very high.
# And I will add its residuals as "target_res", and investigate this time plot.

tp$target_res <- fit_linear$residuals
plt_ts(tp, 1, 'target_res','target_res', F)
```



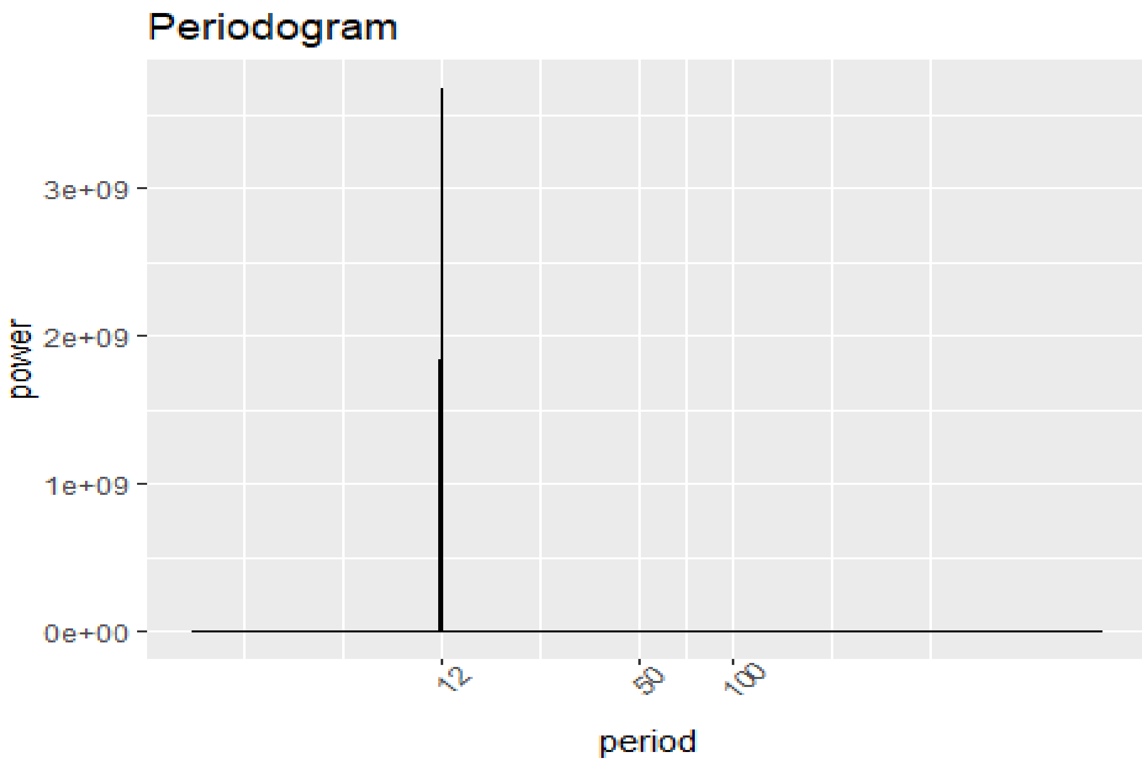



```
plt_ts(tp, 1, 'target_diff_12', 'target_res', T)
```

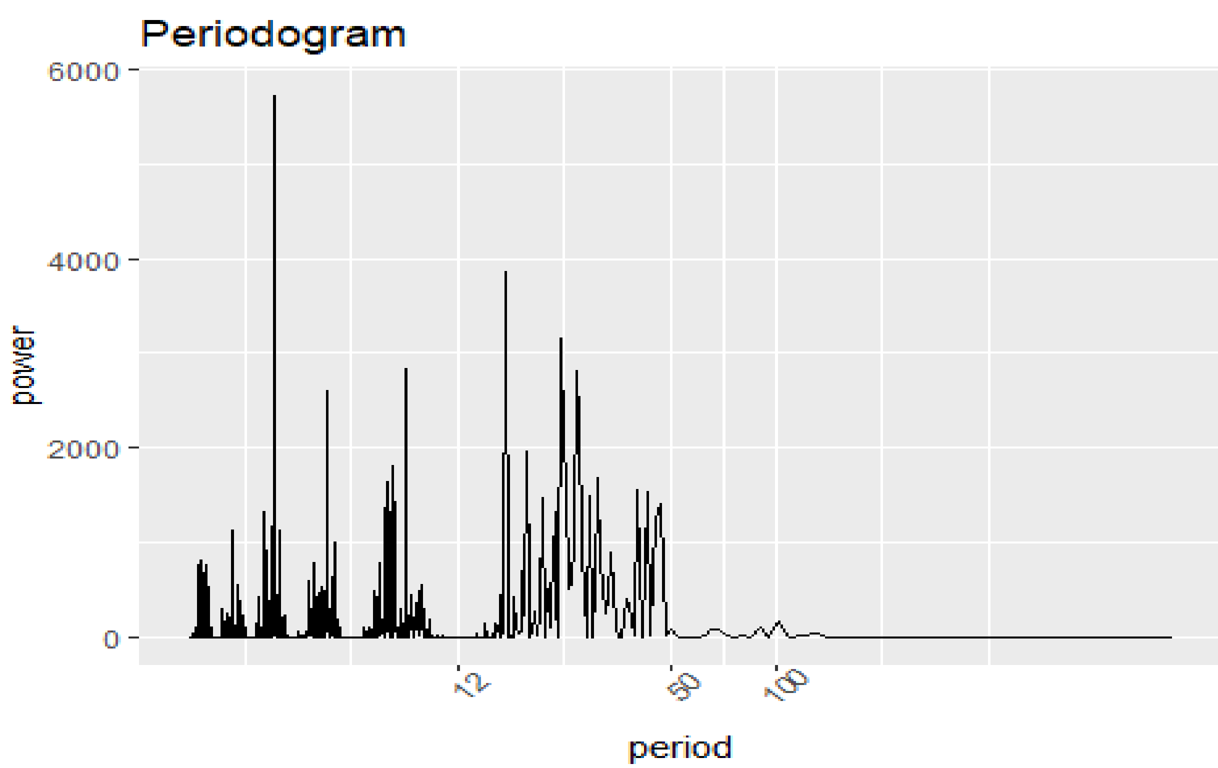



```
# The 4 time plots compare 12th order differenced series vs month effect removed residuals. They all seem to treat seasonality, but I can't find out the better one from the plots. So, I will use periodogram first to identify periodic cycles, and fit both of these new target series for modeling.
```

```r
## Periodogram

pdg <- function(target){

  pdg <- tp %>%
    select(!!sym(target)) %>%
    ts() %>%
    spectrum(plot = FALSE)

  1/pdg$freq[which.max(pdg$spec^2)]

  p1 <- tibble(period = 1/pdg$freq, power = pdg$spec^2) %>%
    ggplot(aes(period, power)) +
    geom_line(color = "black") +
    scale_x_log10(breaks = c(0, 12, 50, 100)) +
    ggtitle('Periodogram') +
    theme(legend.position = "none", axis.text.x  = element_text(angle=45))

  print(str_c("The most noticeable frequency is: ", 1/pdg$freq[which.max(pdg
$spec^2)]))
  plot(p1)
}


pdg("target")

## [1] "The most noticeable frequency is: 12"
```

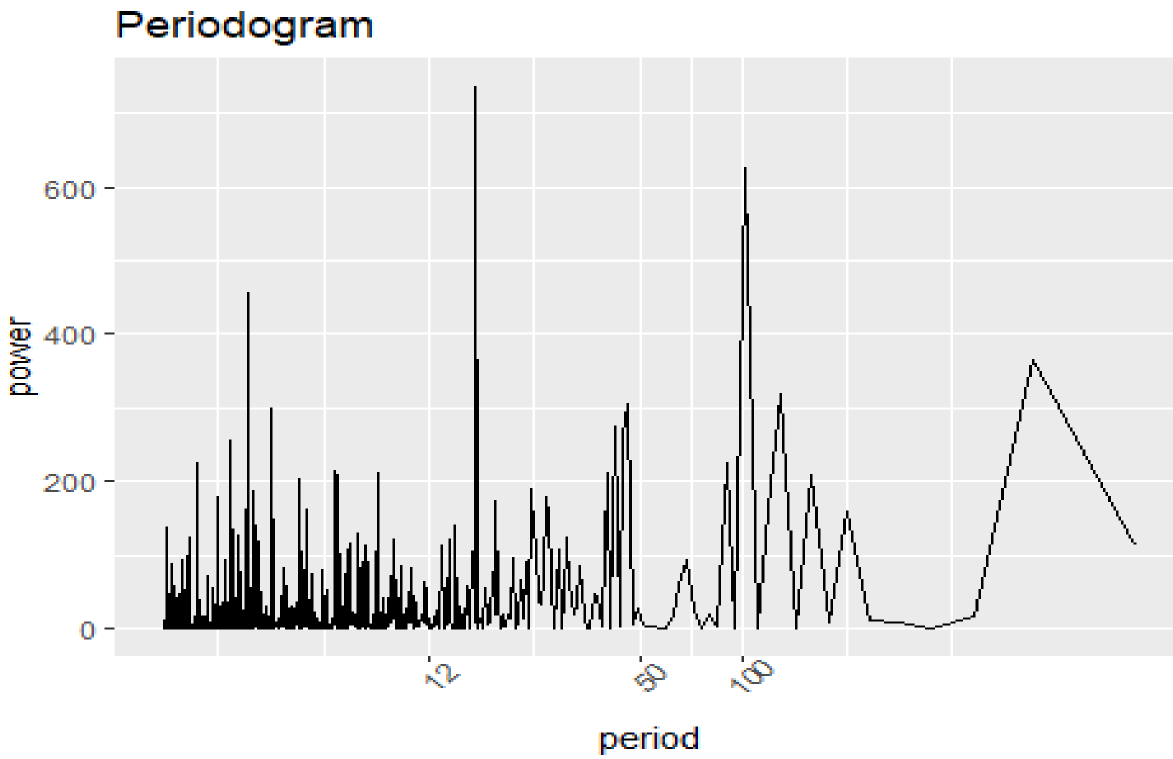



```
pdg("target_diff_12")
## [1] "The most noticeable frequency is: 3.53808353808354"
```



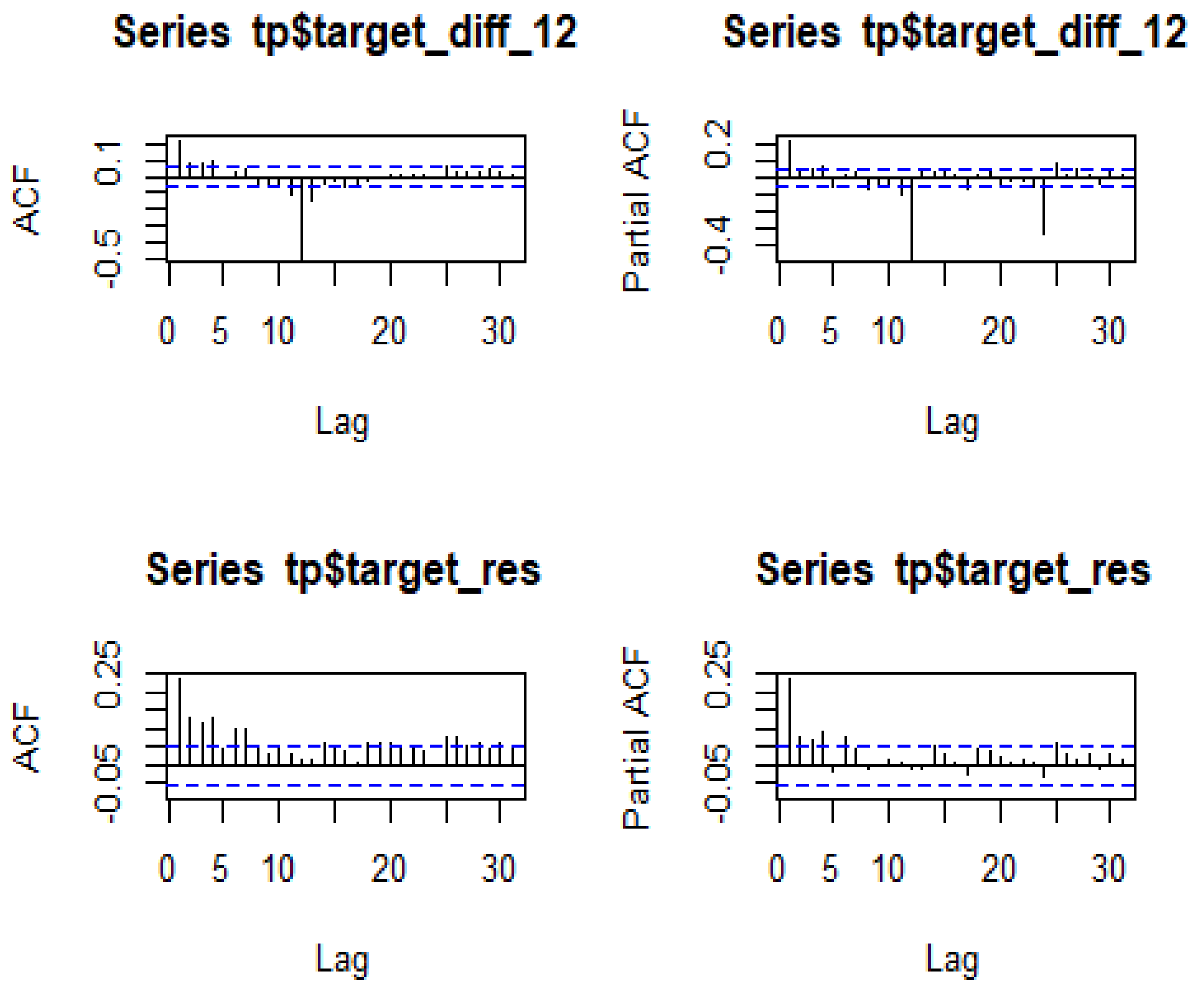
```r
pdg("target_res")
```

```
## [1] "The most noticeable frequency is: 16.551724137931"
```




```r
# From the three periodogram plots, obviously the cycle of the orginal series
 is 12 months. And the other two new targets show, there is no noticeable per
iodicity because the y-axis, power value reduced very much, and there is no d
ominant one among all periods. It means I can use 12 order differenced series
 and month effect removed residuals as new targets.
# But the y-axis of 12th order differenced series is much higher than the oth
er one. It means seasonality could remain for differenced series. So, the next
step is to investigate ACF/PACF plot to find out dependency structures.



## ACF/PACF


par(mfrow = c(2, 2))
Acf(tp$target_diff_12, type = 'correlation')
Pacf(tp$target_diff_12)
```

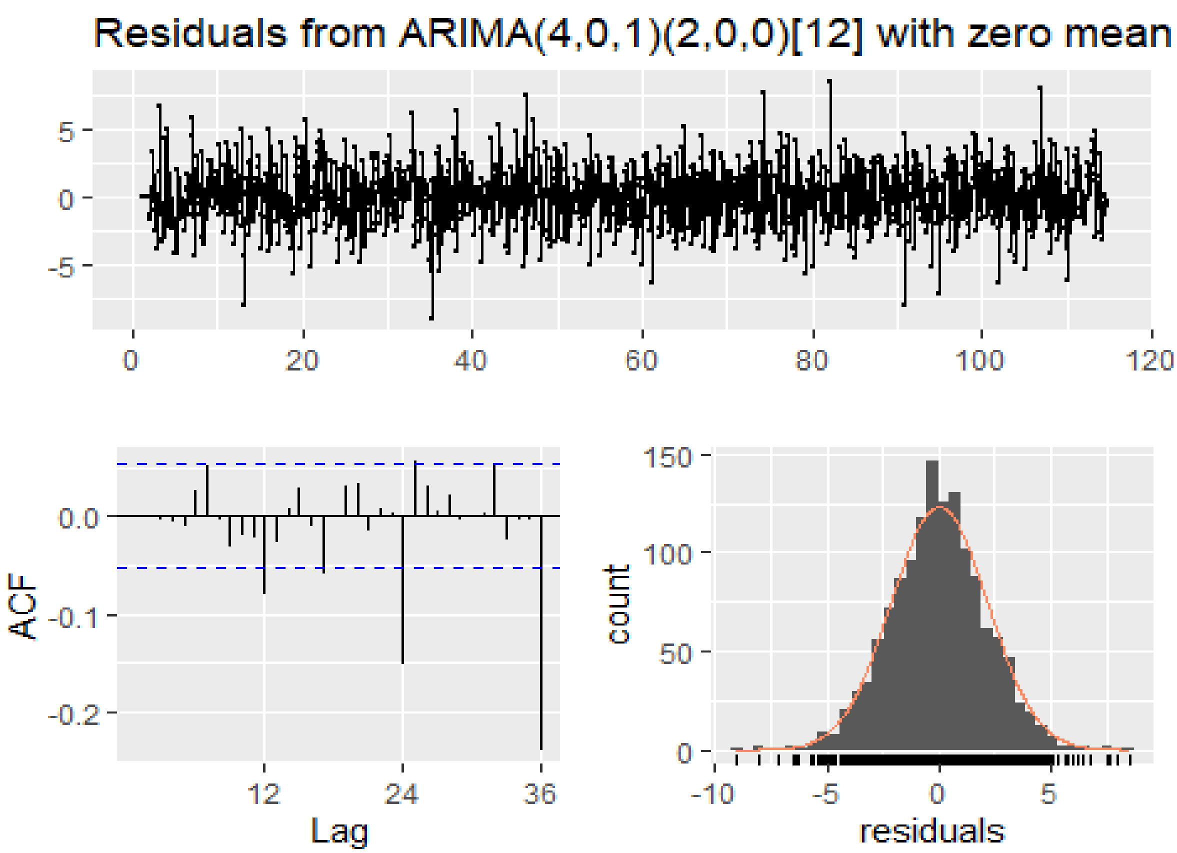
```r
Acf(tp$target_res, type = 'correlation')
Pacf(tp$target_res)
```




```r
# Two plots in the first row show ACF/PACF of 12th order differenced series a
nd, two plots in the second row show those of month effect removed residuals
series.
# It is interesting that they show very different structures. Differenced one
still shows 12-month periodic dependencies because the peaks are repeated eve
ry 12 lag. In contrast, residual series does not show such a cycle, that is,
seasonality seems to be effectively removed. But, ACF and PACF both are not c
ollapsed to zero. So, I can assume ARMA for this series. I will compare two m
odels below..
```

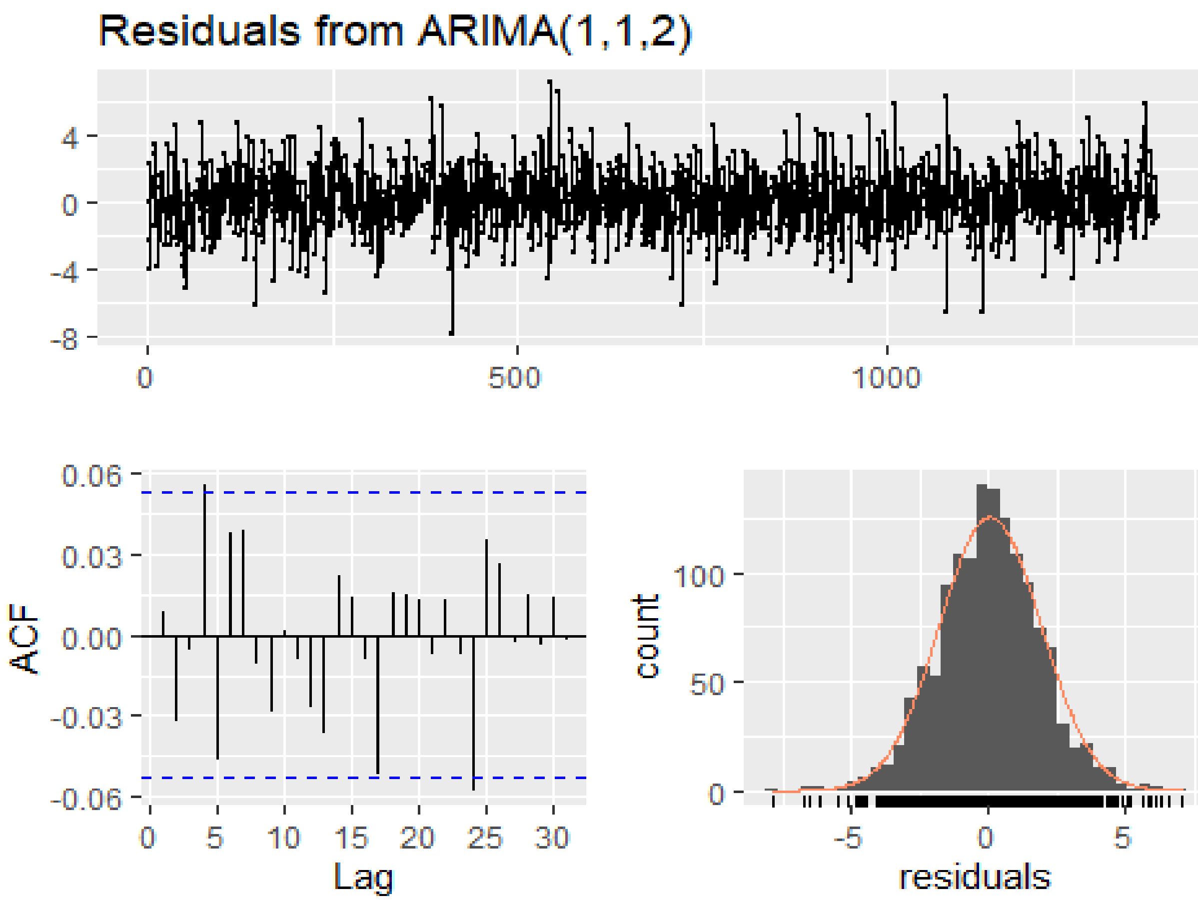
```r
# The first model : SARIMA using 12-order differenced series
# The second model : ARIMA using “Month effect” removed residuals series with
 "auto.arima" function.



fit_diff_12 <- auto.arima(ts(tp$target_diff_12, frequency = 12))
summary(fit_diff_12)

## Series: ts(tp$target_diff_12, frequency = 12)
## ARIMA(4,0,1)(2,0,0)[12] with zero mean
##
## Coefficients:
##           ar1     ar2     ar3     ar4     ma1     sar1     sar2
##       -0.3099  0.1184  0.0579  0.0897  0.5017  -0.6488  -0.3443
## s.e.   0.2237  0.0515  0.0291  0.0272  0.2236   0.0254   0.0254
##
## sigma^2 estimated as 4.651:  log likelihood=-2985.53
## AIC=5987.07   AICc=5987.17   BIC=6028.82
##
## Training set error measures:
##                      ME     RMSE      MAE      MPE     MAPE      MASE
## Training set 0.01436639 2.151074 1.669635 1.421299 296.7061 0.4570954
##                      ACF1
## Training set 0.0009317143

fit_res <- auto.arima(ts(tp$target_res))
summary(fit_res)

## Series: ts(tp$target_res)
## ARIMA(1,1,2)
##
## Coefficients:
##          ar1      ma1     ma2
##       0.5701  -1.3831  0.3893
## s.e.  0.1242   0.1398  0.1380
##
## sigma^2 estimated as 3.438:  log likelihood=-2777.76
## AIC=5563.53   AICc=5563.55   BIC=5584.4
##
## Training set error measures:
##                      ME     RMSE      MAE      MPE     MAPE      MASE
## Training set 0.06688742 1.851466 1.439678 101.0043 189.1487 0.7797775
##                     ACF1
## Training set 0.008944246
```

```r
# From the models' summaries, differenced target series was fitted ARIMA(4,0,1)(2,0,0) with frequency term = 12, and its BIC = 6028.82. And the residual target series was fitted ARIMA(1,1,2), and its BIC = 5584.4 which is lower than the first model. The second model using residuals is much simpler than the differenced series.
# So, the best model would be ARIMA(1,1,2) using residual target series.
# In addtion, this is temperature data, so there would be no outlier, which was checked from "tsclean" function.
# The next step is a diagnosis of residuals.




## Residual plots of each model
checkresiduals(fit_diff_12)
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(4,0,1)(2,0,0)[12] with zero mean
## Q* = 58.883, df = 17, p-value = 1.605e-06
##
## Model df: 7.   Total lags used: 24
```

```
checkresiduals(fit_res)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,2)
## Q* = 14.186, df = 7, p-value = 0.04797
##
## Model df: 3.   Total lags used: 10
```

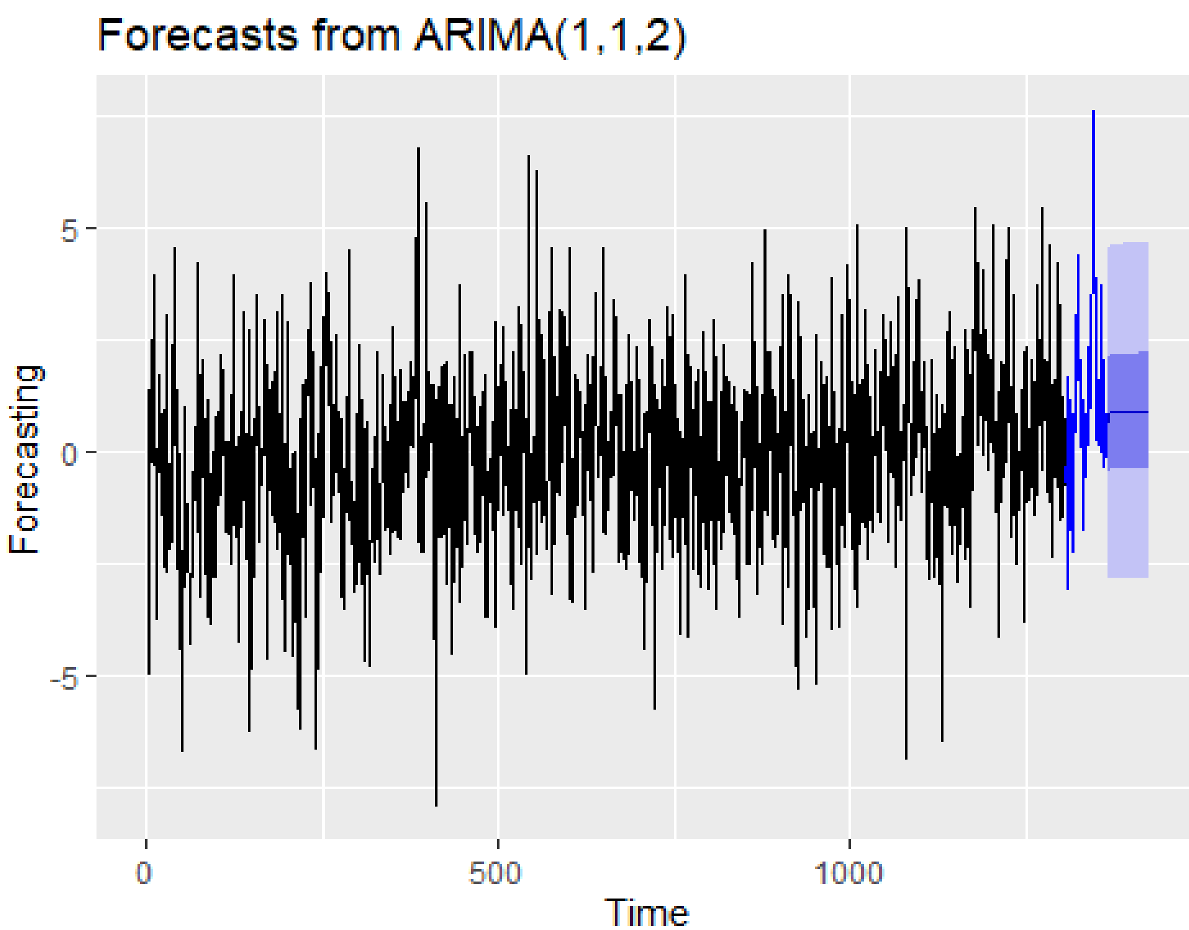
```
# From the residuals plots, the first model ARIMA(4,0,1)(2,0,0) does not show
 good results. This still has seasonality every 12 month cycle, which shows p
eaks at lag = 12, 24, ...
```

```r
# In contrast, ARIMA(1,1,2) model on residuals show that almost every lag is within the zero boundary. So, obviously ARIMA(1,1,2) should be chosen.
# And from the coefficient of AR(1)term = 0.57, the unit root is outside 1, which means stationary.
# But, one thing is that from Ljung-Box test, this model show "Lack of fitting" under the significance level = 0.05, that is, this residuals do not seems to be a white noise series. Maybe other more complicated methods need to be applied to solve this problem. But I will stop here because the diagnostics plots show okay results.
```

IV. Forecasting

```r
tp <- tp %>%
  rownames_to_column() %>%
  mutate(rowname = as.integer(rowname))

draw_forecast <- function(fit, pred_len){

  pred_len <- pred_len
  pred_range <- c(nrow(tp)-pred_len+1, nrow(tp))
  pre <- tp %>% head(nrow(tp)-pred_len)
  post <- tp %>% tail(pred_len)

  fc <- fit %>% forecast(h = pred_len, level = c(50,95))

  p <-  autoplot(fc) +
    geom_line(aes(rowname, target_res), data = post, color = "blue") +
    labs(x = "Time", y = "Forecasting")

  return(p)
}
```

```r
draw_forecast(fit_res, 60)
```




```r
forecast(fit_res, 60)
```

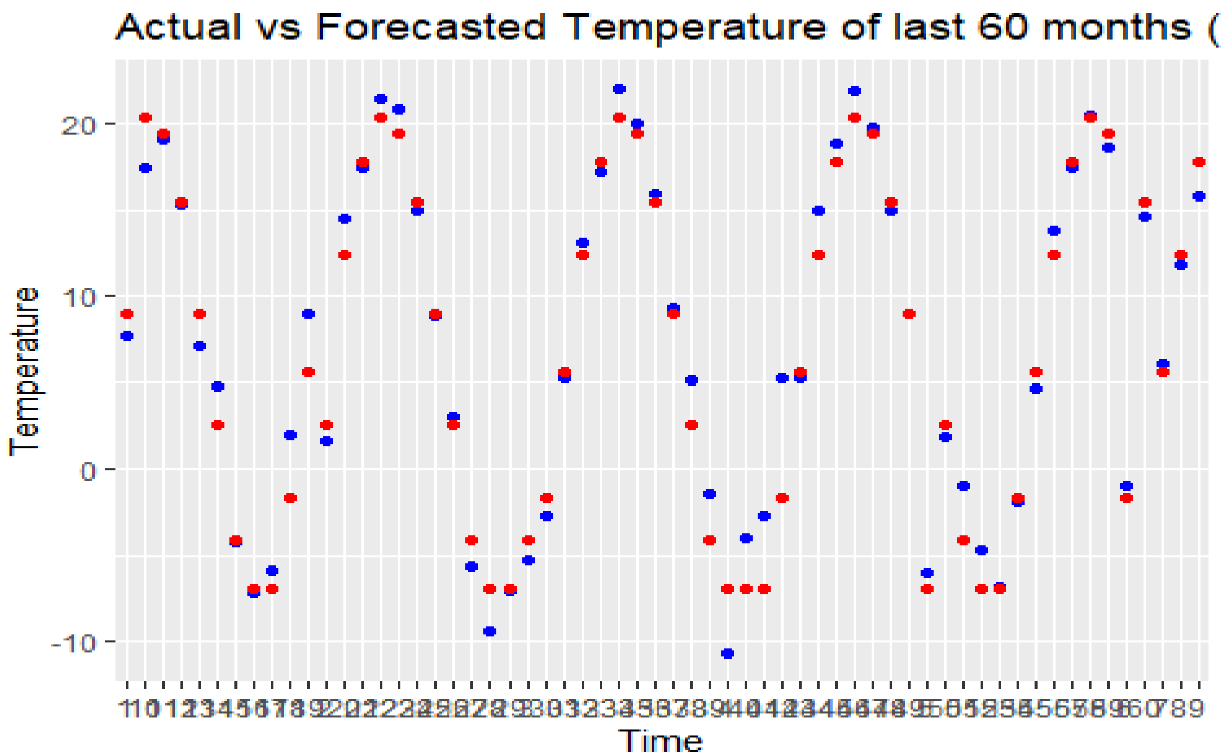
```
##      Point Forecast      Lo 80    Hi 80     Lo 95    Hi 95
## 1366      0.6391081 -1.737125 3.015341 -2.995027 4.273243
## 1367      0.7559709 -1.661483 3.173424 -2.941205 4.453147
## 1368      0.8225976 -1.609682 3.254878 -2.897254 4.542449
## 1369      0.8605834 -1.577464 3.298631 -2.868089 4.589256
## 1370      0.8822402 -1.558300 3.322780 -2.850244 4.614724
## 1371      0.8945874 -1.547186 3.336361 -2.839782 4.628957
## 1372      0.9016268 -1.540858 3.344112 -2.833831 4.637085
## 1373      0.9056402 -1.537323 3.348603 -2.830549 4.641830
## 1374      0.9079284 -1.535400 3.351257 -2.828820 4.644677
## 1375      0.9092329 -1.534404 3.352870 -2.827987 4.646453
## 1376      0.9099767 -1.533938 3.353891 -2.827668 4.647622
## 1377      0.9104007 -1.533775 3.354577 -2.827644 4.648445
## 1378      0.9106425 -1.533785 3.355070 -2.827787 4.649072
## 1379      0.9107803 -1.533894 3.355455 -2.828027 4.649587
## 1380      0.9108589 -1.534059 3.355777 -2.828321 4.650038
```

```
## 1381       0.9109037 -1.534256 3.356064 -2.828646 4.650453
## 1382       0.9109292 -1.534472 3.356330 -2.828989 4.650847
## 1383       0.9109438 -1.534698 3.356585 -2.829342 4.651230
## 1384       0.9109521 -1.534929 3.356834 -2.829701 4.651605
## 1385       0.9109568 -1.535165 3.357078 -2.830063 4.651977
## 1386       0.9109595 -1.535402 3.357321 -2.830427 4.652346
## 1387       0.9109611 -1.535640 3.357562 -2.830792 4.652714
## 1388       0.9109619 -1.535879 3.357802 -2.831158 4.653082
## 1389       0.9109624 -1.536118 3.358042 -2.831524 4.653448
## 1390       0.9109627 -1.536357 3.358282 -2.831890 4.653815
## 1391       0.9109629 -1.536596 3.358522 -2.832256 4.654182
## 1392       0.9109630 -1.536836 3.358762 -2.832622 4.654548
## 1393       0.9109630 -1.537075 3.359001 -2.832988 4.654914
## 1394       0.9109631 -1.537314 3.359241 -2.833354 4.655280
## 1395       0.9109631 -1.537554 3.359480 -2.833720 4.655647
## 1396       0.9109631 -1.537793 3.359719 -2.834087 4.656013
## 1397       0.9109631 -1.538033 3.359959 -2.834453 4.656379
## 1398       0.9109631 -1.538272 3.360198 -2.834819 4.656745
## 1399       0.9109631 -1.538511 3.360437 -2.835185 4.657111
## 1400       0.9109631 -1.538751 3.360677 -2.835551 4.657477
## 1401       0.9109631 -1.538990 3.360916 -2.835917 4.657843
## 1402       0.9109631 -1.539229 3.361155 -2.836283 4.658209
## 1403       0.9109631 -1.539468 3.361395 -2.836648 4.658575
## 1404       0.9109631 -1.539708 3.361634 -2.837014 4.658940
## 1405       0.9109631 -1.539947 3.361873 -2.837380 4.659306
## 1406       0.9109631 -1.540186 3.362112 -2.837746 4.659672
## 1407       0.9109631 -1.540425 3.362351 -2.838112 4.660038
## 1408       0.9109631 -1.540664 3.362590 -2.838477 4.660403
## 1409       0.9109631 -1.540903 3.362829 -2.838843 4.660769
## 1410       0.9109631 -1.541142 3.363069 -2.839209 4.661135
## 1411       0.9109631 -1.541381 3.363308 -2.839574 4.661500
## 1412       0.9109631 -1.541620 3.363547 -2.839940 4.661866
## 1413       0.9109631 -1.541859 3.363786 -2.840305 4.662231
## 1414       0.9109631 -1.542098 3.364025 -2.840671 4.662597
## 1415       0.9109631 -1.542337 3.364264 -2.841036 4.662962
## 1416       0.9109631 -1.542576 3.364502 -2.841402 4.663328
## 1417       0.9109631 -1.542815 3.364741 -2.841767 4.663693
## 1418       0.9109631 -1.543054 3.364980 -2.842132 4.664058
## 1419       0.9109631 -1.543293 3.365219 -2.842498 4.664424
## 1420       0.9109631 -1.543532 3.365458 -2.842863 4.664789
## 1421       0.9109631 -1.543771 3.365697 -2.843228 4.665154
## 1422       0.9109631 -1.544009 3.365936 -2.843593 4.665519
## 1423       0.9109631 -1.544248 3.366174 -2.843958 4.665885
## 1424       0.9109631 -1.544487 3.366413 -2.844324 4.666250
## 1425       0.9109631 -1.544726 3.366652 -2.844689 4.666615
```

```
# From the forecasting plot, it does not seem to capture the variation well.
# Except some initial forecasts, the values are all very similar, this would
be because this models contains 1 terms of a AR term, and 2 terms of MA terms
```

```
 with small coefficients and small variance of MA noises. So, this model is n
ot likely to show large variations. Other more complicated models should be u
sed to forecast.


# To get temperature data from "Month effect" removed residuals, some c
alculation is needed below.

get_temperature <- function(fit_linear, ini_month, residuals){

  coef_linear_fitting <- rep(c(0, fit_linear$coefficients[2:12]), 100)

  forecasts <- c()
  for(j in 1:length(residuals)){
  forecasts[j] <- -7.609658 + coef_linear_fitting[j+ini_month-1] + residuals
[j]}
  return(round(forecasts, 1))
}

# This is a transformed result from residuals.
get_temperature(fit_linear, 10, forecast(fit_res, 60)$mean)

##  [1]  9.0  2.6 -3.9 -6.7 -6.7 -1.5  5.9 12.7 17.9 20.5 19.6 15.7  9.3  2.8
## [15] -3.8 -6.7 -6.7 -1.5  5.9 12.7 17.9 20.5 19.6 15.7  9.3  2.8 -3.8 -6.7
## [29] -6.7 -1.5  5.9 12.7 17.9 20.5 19.6 15.7  9.3  2.8 -3.8 -6.7 -6.7 -1.5
## [43]  5.9 12.7 17.9 20.5 19.6 15.7  9.3  2.8 -3.8 -6.7 -6.7 -1.5  5.9 12.7
## [57] 17.9 20.5 19.6 15.7


# Now, I will calculate RMSE between actual temperature and forecasts
of last 60 months.

tp %>% slice(1306:1365) %>% tail()

## # A tibble: 6 x 14
##   rowname dt         target AverageTemperat~ city  country lat   long
##     <int> <date>      <dbl>            <dbl> <chr> <chr>   <chr> <chr>
## 1    1360 2013-04-01   4.66            0.318 Toro~ Canada  44.2~ 80.5~
```

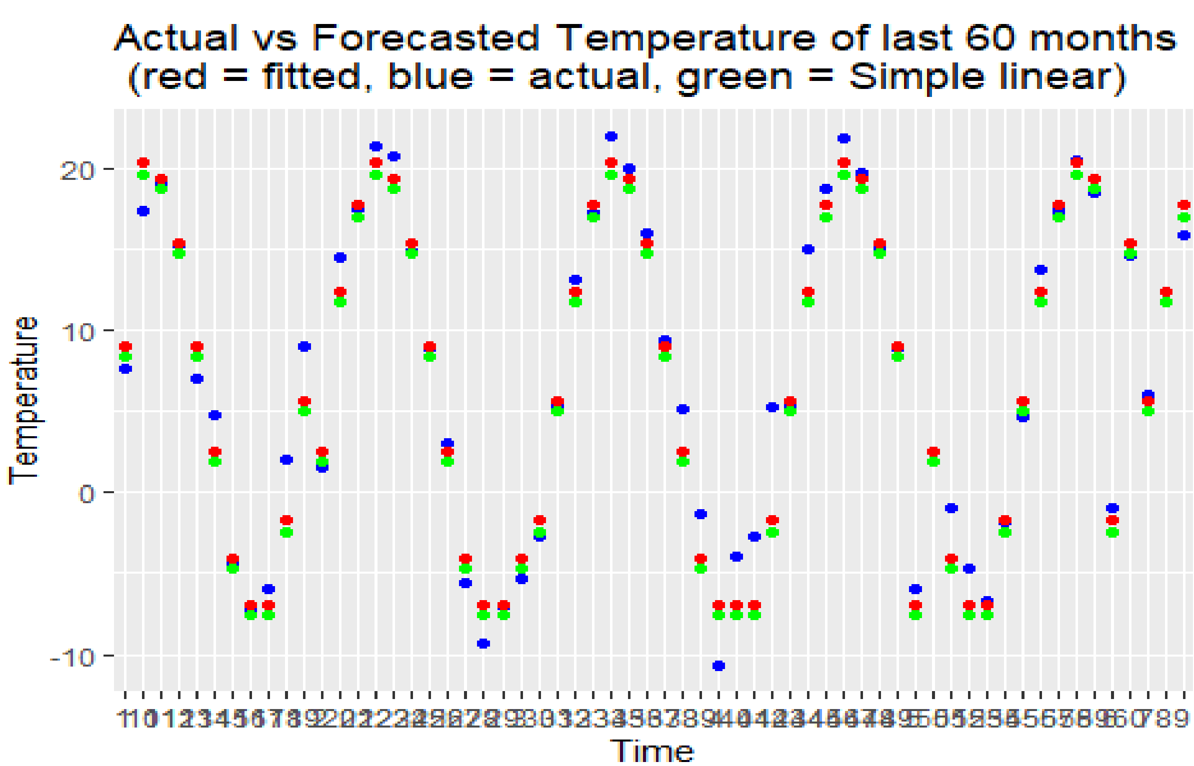
```
## 2    1361 2013-05-01  13.8             0.278 Toro~ Canada  44.2~ 80.5~
## 3    1362 2013-06-01  17.4             0.226 Toro~ Canada  44.2~ 80.5~
## 4    1363 2013-07-01  20.5             0.290 Toro~ Canada  44.2~ 80.5~
## 5    1364 2013-08-01  18.5             0.342 Toro~ Canada  44.2~ 80.5~
## 6    1365 2013-09-01  14.6             1.27  Toro~ Canada  44.2~ 80.5~
## # ... with 6 more variables: wday <ord>, year <dbl>, month <dbl>,
## #   diff <dbl>, target_diff_12 <dbl>, target_res <dbl>

train <- tp %>%
  filter(dt > ymd("1900-01-01") & dt < ymd("2008-10-01"))

test <- tp %>%
  filter(dt > ymd("2008-09-01") & dt < ymd("2013-10-01"))

fit_val <- arima(train$target_res, order = c(1, 1, 2))


rmse <- function(actual, fitted){

  sqrt(sum(actual - fitted)^2)

}

## Plot actual residual vs fitted residual in last 60 months

fitted <- get_temperature(fit_linear, 10, forecast(fit_val, 60)$mean)

tibble(actual = test$target, fitted = fitted) %>%
  rownames_to_column() %>%
  ggplot() +
  geom_point(aes(rowname, actual), color = "blue") +
  geom_point(aes(rowname, fitted), color = "red") +
  labs(x = 'Time', y = 'Temperature') +
  ggtitle('Actual vs Forecasted Temperature of last 60 months (red = fitted,
blue = actual)')
```




```r
rmse(test$target, fitted)
## [1] 26.433
# RMSE is 26.43, and from the plot, the values are close to actual temperatur
es. So, this forecasting looks good. One limit is that I didn't split the tes
t set when modeling, so this test is not a perfect validation dataset.
# So, this forecasting could show better result than what it really is.

# And from the linear regression summary using "Month" variable, the R-square
d was 96%. So, this good forecasting result is mainly from linear regression
rather than times series model, because in the previous forecasting plot, the
 ARIMA model does not capture variation very well.
```

V. Some questions

```r
## Does the time series model improve the precision of forecasting?

# I can check comparing RMSE of two models.
# First model : Simple linear regression using Month variable
# Second model : Apply ARIMA on the residuals of the first model.

rmse(test$target, fit_linear$fitted.values[1306:1365])

## [1] 66.65108

fitted2 <- fit_linear$fitted.values[1306:1365]

tibble(actual = test$target, fitted = fitted) %>%
  rownames_to_column() %>%
  ggplot() +
  geom_point(aes(rowname, actual), color = "blue") +
  geom_point(aes(rowname, fitted), color = "red") +
  geom_point(aes(rowname, fitted2), color = "green") +
  labs(x = 'Time', y = 'Temperature') +
  ggtitle('Actual vs Forecasted Temperature of last 60 months \n (red = fitte
d, blue = actual, green = Simple linear)')
```



```
# RMSE without time series fitting is 66.65. This is much higher than times s
eries model RMSE: 26.43
# So, for this data, time series model performs well.


## Is forecasting for winter season that has larger variance harder th
an summer season?

# From the density plots of each month,
# Winter season : 12, 1, 2, 3 months
# Summer season : 6, 7, 8, 9 months

w <- c(3,4,5,6)
winter <- c(w, w+12, w+24, w+36, w+48)
s <- c(9,10,11,12)
summer <- c(s, s+12, s+24, s+36, s+48)

rmse(test$target[winter], fitted[winter])

## [1] 18.146

rmse(test$target[summer], fitted[summer])

## [1] 0.584


# The result is so interesting, RMSE of winter season: 18.146 is much higher
than the summer season: 0.584, so more caution is needed to predict more vari
able period in the series.


VI. Summary


# Because this is temperature data, it has seasonality whose the cycle is 12
```

```
months.
# So, this should be removed to make the stationary series. I used linear reg
ression and 12th order differencing to remove the seasonality. First, month e
ffect removed residual series does not remain seasonality, but differencing d
oes not work well. These are checked from ACF/PACF plots and periodogram. So,
 the final model is ARIMA(1,1,2) using residuals series
# The orders are chosen from AIC using "auto.arima" function.
# Regarding forcasting, ARIMA model does not catch the variation well, it jus
t converges to a number as the period goes far from the last data point. So,
the more complicated model is needed to forecast. But still, ARIMA model was
meaningful because when comparing RMSE with the model without ARIMA, it shows
 much better performance.

# And some periods that has larger variation obviously showed much worse
prediction than the periods that has smaller variation.
```