Forecasting TB Incidence in Guangxi, China

David Apamo

2024-11-13

The data for this study was obtained online from an article by Yanling Zheng, Liping Zhang, Lei Wang and Ramziya Rifhat 2020, titled *"Statistical methods for predicting tuberculosis incidence based on data from Guangxi, China"*. Dataset link: https://ndownloader.figstatic.com/files/22392300. The article was published by BMC Infectious Diseases. In their study, (Zheng et al., 2020) outlines that Tuberculosis (TB) remains a serious public health problem with substantial financial burden in China. The incidence of TB in Guangxi province is much higher than that in the national level, therefore necessitating an urgent construction of a forecasting model that can accurately predict the incidence of TB, and help in the prevention and control of TB.

```
# Load packages
suppressMessages(
  {
    library(tidyverse)
    library(readxl)
    library(tseries)
    library(forecast)
    library(ggfortify)
    library(prophet)
  }
)

# Import data
TB_data <- read_excel("TB time series data.xlsx")

# View the first five observations
head(TB_data, n = 5)
```

```
## # A tibble: 5 × 2
##   Time                `TB incidence (per 100,000 populations)`
##   <dttm>                                                 <dbl>
## 1 2012-01-31 00:00:00                                     13.0
## 2 2012-02-29 00:00:00                                     17.1
## 3 2012-03-31 00:00:00                                     20.2
## 4 2012-04-30 00:00:00                                     18.2
## 5 2012-05-31 00:00:00                                     18.3
```

```
# View the structure of the dataset
glimpse(TB_data)
```

```
## Rows: 90
## Columns: 2
## $ Time                                      <dttm> 2012-01-31, 2012-02-29,
2012…
## $ `TB incidence (per 100,000 populations)` <dbl> 13.01, 17.09, 20.25,
18.24, 1…
```

The data has 90 observations of 2 variables. The first variable Time represent the date when each observation was recorded, while the second variable represents the TB incidence in Guangxi, China per 100,000 populations.

Data Cleaning

```
# Check for missing values
map_dbl(TB_data, ~sum(is.na(.)))

##                                    Time TB incidence (per 100,000
populations)
##                                       0
0
```

There are no missing values in the data.

```
# Check for duplicated observations
sum(duplicated(TB_data))

## [1] 0
```

There are no duplicated observations in the data. The data is clean and I can now generate various summary statistics.

```
# Convert the variable Time into a date format
TB_data$Time <- as.Date(TB_data$Time)

# Generate summary statistics
summary(TB_data)

##       Time             TB incidence (per 100,000 populations)
##  Min.   :2012-01-31   Min.   : 9.24
##  1st Qu.:2013-12-07   1st Qu.:12.14
##  Median :2015-10-15   Median :13.40
##  Mean   :2015-10-15   Mean   :13.72
##  3rd Qu.:2017-08-23   3rd Qu.:15.03
##  Max.   :2019-06-30   Max.   :20.25
```

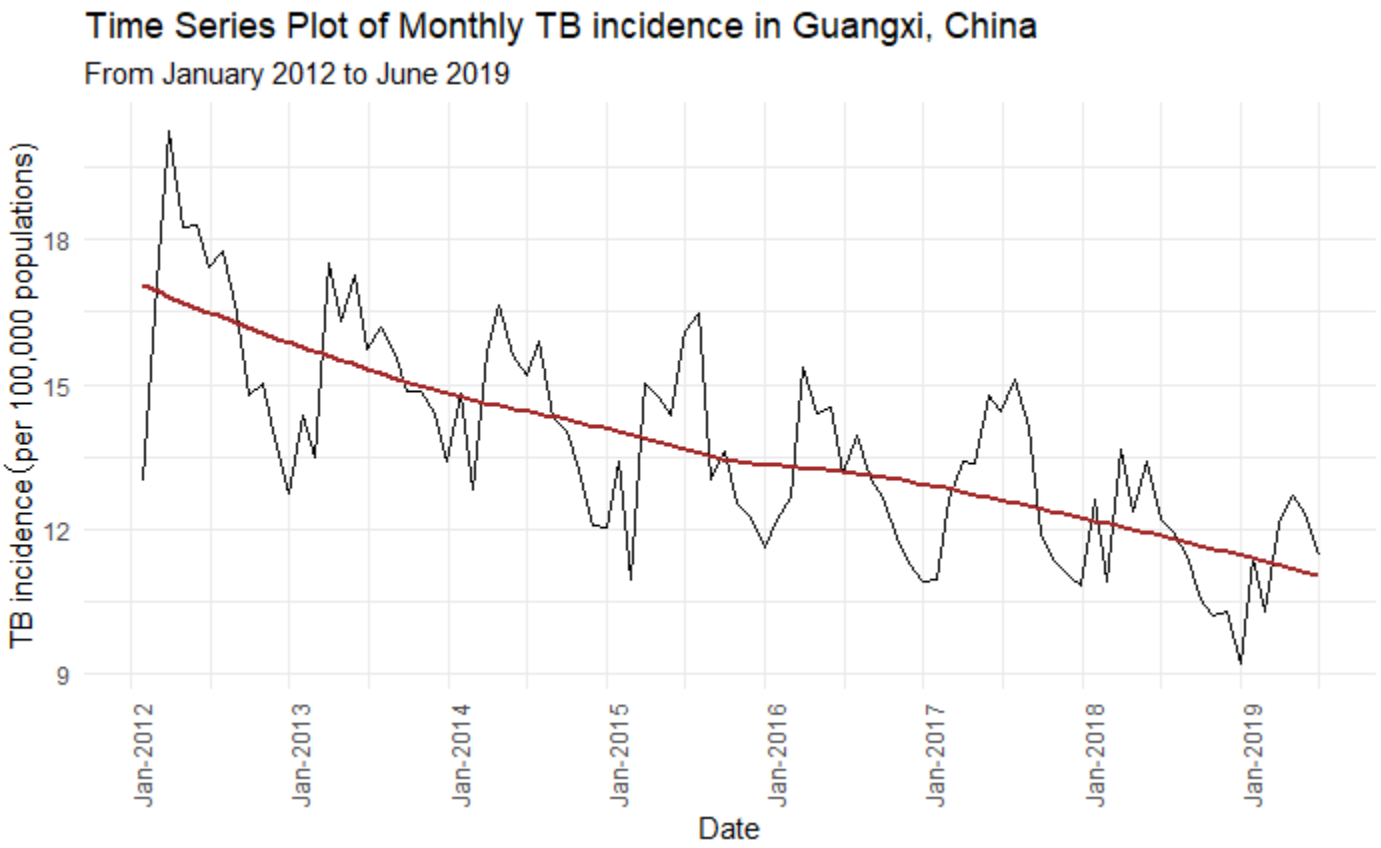
The first observation was recorded in January 2012 and the last observation was recorded in June 2019. The lowest and highest TB incidences were 9.24 and 20.25 per 100,000 populations respectively.

There are no inconsistencies in the data.

Plotting the Time Series

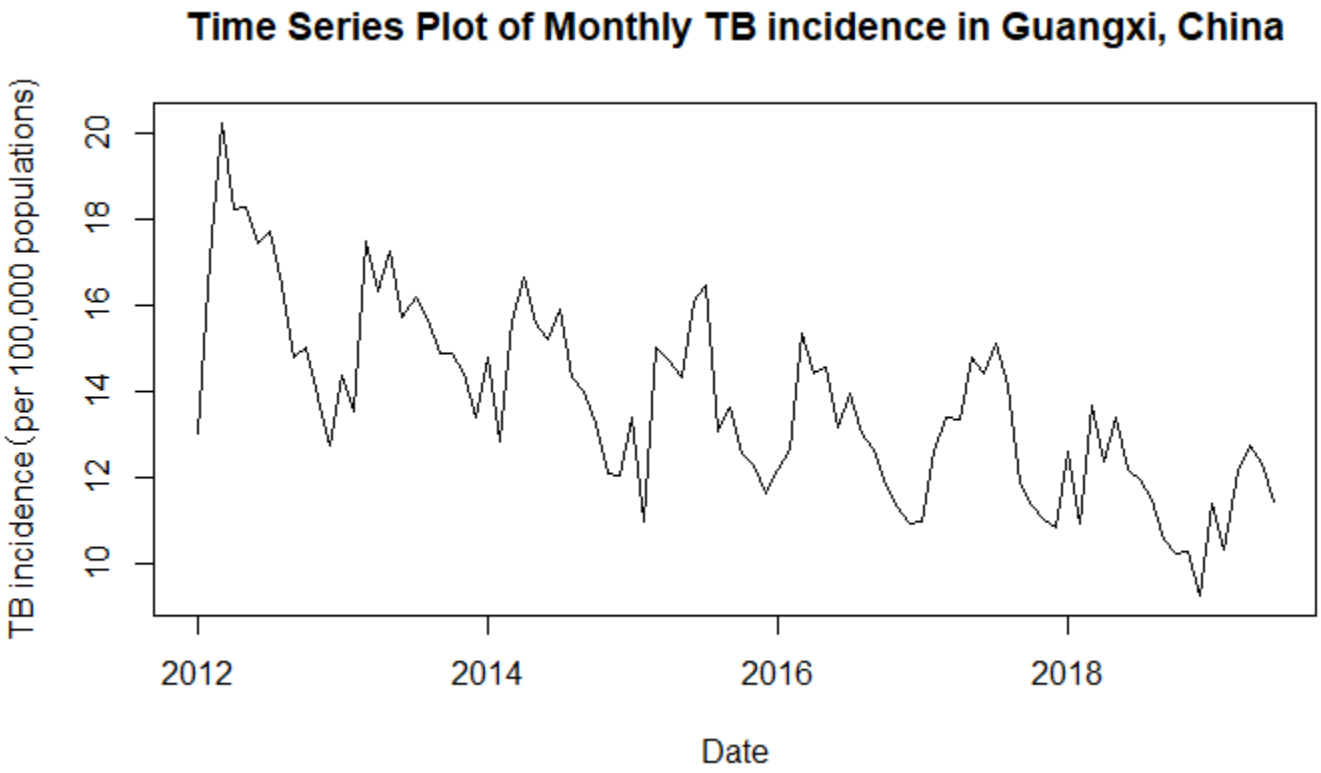
```
## Plot using ggplot2

# define grid and aesthetic mappings
ggplot(TB_data, aes(x = Time, y = `TB incidence (per 100,000 populations)`))
+
  # add a line plot layer
  geom_line() +
  # add a smoothing layer to depict the trend, suppress the SE
  geom_smooth(color = "brown", se = F) +
  # use 1 year interval on the X-axis
  scale_x_date(date_breaks = "1 year", date_labels = "%b-%Y") +
  # add a title and a subtitle
  labs(title = "Time Series Plot of Monthly TB incidence in Guangxi, China",
       subtitle = "From January 2012 to June 2019", x = "Date") +
  # add minimal theme
  theme_minimal() +
  # rotate X-axis labels to 90 degrees
  theme(axis.text.x = element_text(angle = 90, hjust = 1))
```

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



- There is a decreasing trend in TB incidence.
- There is also seasonality in the time series.
- The mean and variance of the series are not constant over time, hence the series is not stationary.
- Since the variance doesn't seem to increase with time, the time series can be explained by an additive model.

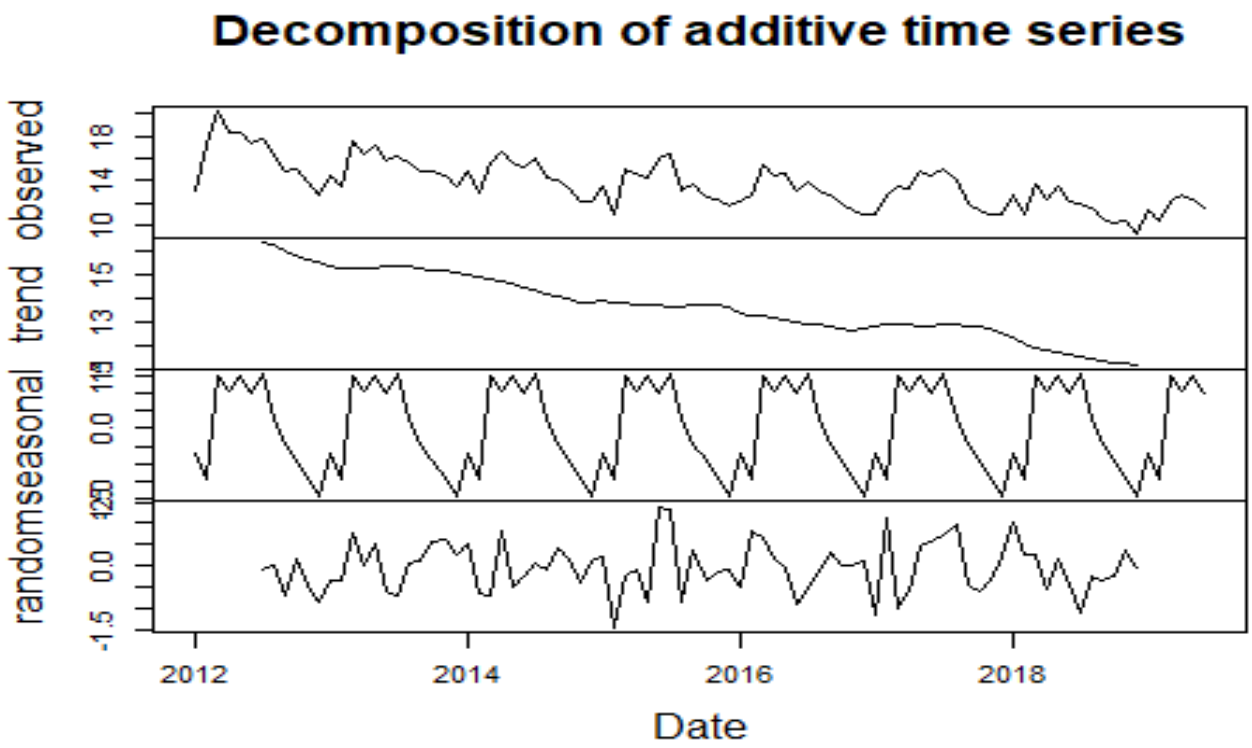
```
# Convert the data into a time series object (the data is a monthly series
starting from January 2012)
TB_Incidence.ts <- ts(TB_data$`TB incidence (per 100,000 populations)`,
                      frequency = 12, start = c(2012,1))

# Plot the time series using the plot.ts() function from stats package
plot.ts(TB_Incidence.ts, main ="Time Series Plot of Monthly TB incidence in
Guangxi, China", xlab = "Date",
        ylab = "TB incidence (per 100,000 populations)")
```



Decomposing the Time Series

Since the series shows a downward trend with seasonality, I will decompose it into its constituent elements (i.e. trend, seasonality and random component) to be able to identify the various patterns in the series (GeeksforGeeks, 2023).

```r
# Decompose the series to separate it into its constituent elements
TB_Incidence_decomposed <- decompose(TB_Incidence.ts, type = "additive")

# Plot the decomposed series
plot(TB_Incidence_decomposed, xlab = "Date")
```



The downward trend is now clearly visible. The seasonal and random components (variations) of the series are also clearly visible. I'll extract the seasonal component from the decomposed series to be able to know the exact periods of the seasonality.

```r
# Extract the seasonal component from the decomposed series
TB_Incidence_decomposed$seasonal
```

```
##             Jan        Feb        Mar        Apr        May        Jun
## 2012 -0.7103406 -1.4405489  1.4700761  1.0677844  1.4759094  1.0066038
## 2013 -0.7103406 -1.4405489  1.4700761  1.0677844  1.4759094  1.0066038
## 2014 -0.7103406 -1.4405489  1.4700761  1.0677844  1.4759094  1.0066038
## 2015 -0.7103406 -1.4405489  1.4700761  1.0677844  1.4759094  1.0066038
## 2016 -0.7103406 -1.4405489  1.4700761  1.0677844  1.4759094  1.0066038
## 2017 -0.7103406 -1.4405489  1.4700761  1.0677844  1.4759094  1.0066038
## 2018 -0.7103406 -1.4405489  1.4700761  1.0677844  1.4759094  1.0066038
## 2019 -0.7103406 -1.4405489  1.4700761  1.0677844  1.4759094  1.0066038
##             Jul        Aug        Sep        Oct        Nov        Dec
## 2012  1.5139253  0.2482110 -0.4844081 -0.8731581 -1.3604795 -1.9135747
## 2013  1.5139253  0.2482110 -0.4844081 -0.8731581 -1.3604795 -1.9135747
## 2014  1.5139253  0.2482110 -0.4844081 -0.8731581 -1.3604795 -1.9135747
```

```
## 2015  1.5139253  0.2482110 -0.4844081 -0.8731581 -1.3604795 -1.9135747
## 2016  1.5139253  0.2482110 -0.4844081 -0.8731581 -1.3604795 -1.9135747
## 2017  1.5139253  0.2482110 -0.4844081 -0.8731581 -1.3604795 -1.9135747
## 2018  1.5139253  0.2482110 -0.4844081 -0.8731581 -1.3604795 -1.9135747
## 2019
```

The largest seasonal component is in July (1.5139) and the lowest seasonal component is in December (-1.9136), implying that there is a peak in TB incidence every July and a trough every December each year.

Checking for Stationarity

Stationarity is one of the major assumptions of ARIMA model, and it is therefore important to check if the stationarity assumption holds, before fitting an ARIMA model (Coghlan, 2018). If the series is not stationary, I will perform differencing to make it stationary. I'll use the Augmented Dickey-Fuller (ADF) test and KPSS test to check for stationarity.

```
# Check for stationarity using ADF test

# Ho: The time series has a unit root i.e. it is not stationary
# Ha: The time series doesn't have a unit root i.e. it is stationary
adf.test(TB_Incidence.ts, k = 20)

##
##  Augmented Dickey-Fuller Test
##
## data:  TB_Incidence.ts
## Dickey-Fuller = -2.5914, Lag order = 20, p-value = 0.3327
## alternative hypothesis: stationary
```

The p-value (0.3327) is much greater than 0.05, hence I fail to reject the null hypothesis at 5% level of significance and conclude that the time series is not stationary.

```
# Check for stationarity using KPSS test
# Ho: The time series is stationary
# Ha: The time series is not stationary
kpss.test(TB_Incidence.ts)

## Warning in kpss.test(TB_Incidence.ts): p-value smaller than printed p-
value

##
##  KPSS Test for Level Stationarity
##
## data:  TB_Incidence.ts
## KPSS Level = 1.5344, Truncation lag parameter = 3, p-value = 0.01
```

The p-value (0.01) for the KPSS test is less than 0.05, providing enough evidence of non-stationarity in the series.

Forecasting

I will forecast for future TB incidence using ARIMA model and Prophet. Before I forecast, I'll first partition the data into training and validation sets using 85/15% split. The training set will have data for the first 77 months and the validation set will have data for the last 13 months. I'll use the training set for model training, and use the validation set for model evaluation.

```
## Data partitioning

# Subset the training set
training_data <- TB_data[1:77, ]
# Convert the training set into a time series object
train <- ts(training_data$`TB incidence (per 100,000 populations)`,
            frequency = 12, start = c(2012,1))

# Subset validation set
test_data <- TB_data[78:90, ]
# Convert the validation set into a time series object
test <- ts(test_data$`TB incidence (per 100,000 populations)`,
           frequency = 12, start = c(2018,6))
```

1.Forecast Using ARIMA model

I'll use the auto.arima() function to identify the candidate ARIMA model. Since the data has seasonality, I'll use the argument *"seasonal = TRUE"* inside the auto.arima() function. The function automatically identifies the optimal ARIMA model based on metrics like AIC, BIC, RMSE, MAE, MAPE etc (Coghlan, 2018).

```
# Build a SARIMA model using the auto.arima() function
sarima_model <- auto.arima(train, seasonal = TRUE)
# Have a model summary
summary(sarima_model)

## Series: train
## ARIMA(0,0,1)(0,1,1)[12] with drift
##
## Coefficients:
##          ma1     sma1    drift
##       0.2394  -0.8637  -0.0570
## s.e.  0.1548   0.4503   0.0056
##
## sigma^2 = 0.8451:  log likelihood = -92.47
## AIC=192.94   AICc=193.61   BIC=201.64
##
## Training set error measures:
##                       ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -0.05221832 0.824908 0.598949 -0.4344504 4.392384 0.5622716
```

```
##                      ACF1
## Training set 0.001020802
```

The candidate model is ARIMA(0,0,1)(0,1,1)[12]. Non-seasonal parameters are; p=0, d=0, q=1 (no autoregressive terms (*p*), 0 non-seasonal differencing (*d*), and a moving average model of order 1 (*q*)). Seasonal parameters are P=0, D=1, Q=1, s=12. (First Seasonal differencing, a moving average model of order 1 and 12 periods/months). Even though not very closer to zero, the training RMSE and MAE values are low, indicating a good fit.

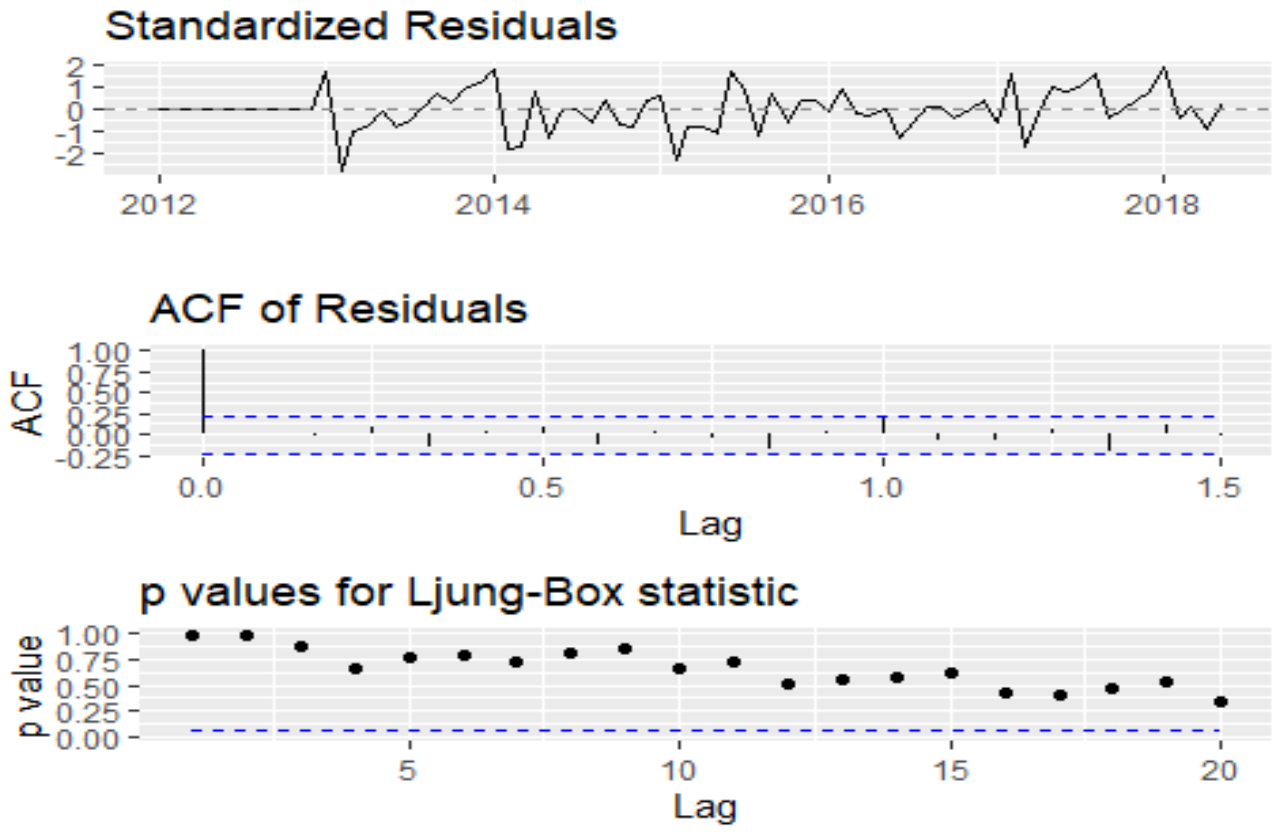
The Seasonal ARIMA fitted model is:

- Yt − Yt−12 = et + θ1et−1 + Θ1et−12 + Θ1θ1et−13 + E, where Yt is the observed value at time t, θ1 is the coefficient for the non-seasonal MA(1) term, Θ1 is the coefficient for the seasonal MA(1) term and E is some error.

Model Diagnostics

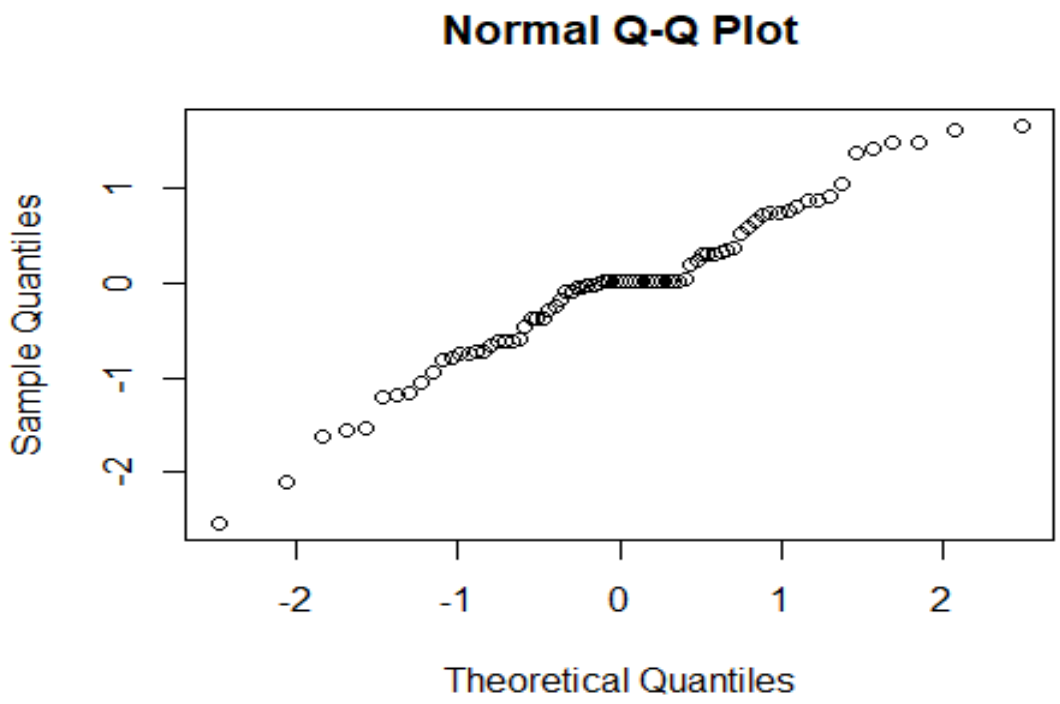
Before I use the SARIMA model to forecast for future TB incidence, I'll first carry out model diagnostics to check if the ARIMA model assumptions hold i.e. stationarity, normality of the residuals and no autocorrelations between the error terms(residuals).

```
# Carry out model diagnostics
ggtsdiag(sarima_model, gof.lag = 20)
```



- The ACFs of the first 20 lags do not exceed the significance bounds, and the p-values from Ljung-Box statistic are also greater than 0.05, providing enough evidence of zero autocorrelations between the error terms.
- The standardized residuals seem to have a constant variance, implying that the first seasonal differencing made the time series stationary.
- Some of the standardized residuals are greater than 3, implying the presence of outliers in the series.

```
# Check for normality of the residuals
qqnorm(sarima_model$residuals)
```



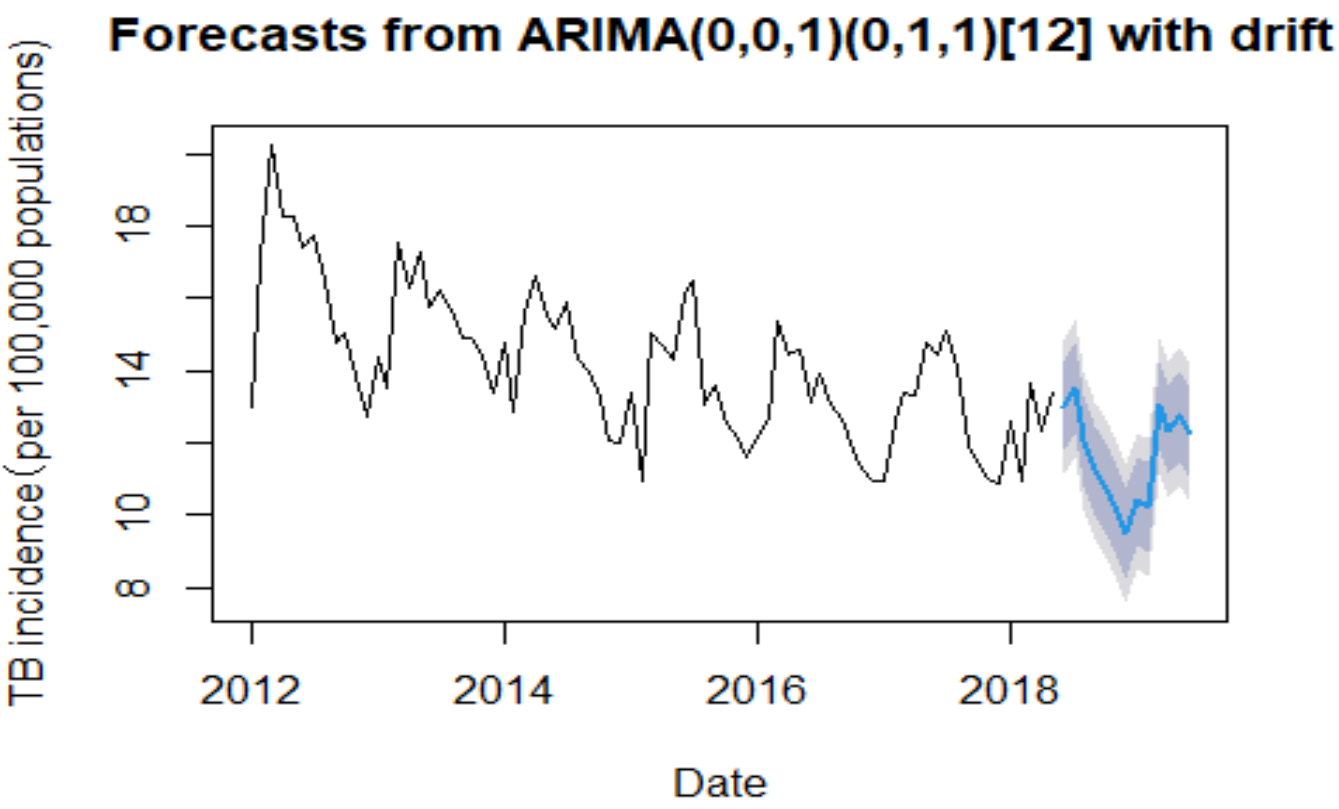
The qq-plot doesn't resemble a straight line, implying that the residuals do not follow a normal distribution. This assumption is thus violated.

Evaluate Performance of the SARIMA model on test data

To evaluate how my Seasonal ARIMA model would perform on new data, I'll forecast for the next 13 months (same as the months in test data) and calculate the out of sample RMSE.

```
# Forecast using the SARIMA model (forecast for the next 13 months)
sarima_forecasts <- forecast(sarima_model, h = 13)
```

```r
# Plot the full forecast
plot(sarima_forecasts, xlab = "Date",
     ylab = "TB incidence (per 100,000 populations)")
```



The out of sample forecast closely follows the pattern of the series. The forecast shows a downward trend in future TB incidence. The seasonality is still persistent, and the residents of Guangxi, China should expect high incidences of TB in March and July each year.

```r
# Collect predictions from SARIMA forecasts
sarima_predictions <- sarima_forecasts$mean

# Find test SSE and RMSE
test_SSE1 <- sum((test - sarima_predictions)^2)
test_SSE1

## [1] 7.173404

test_RMSE1 <- sqrt(test_SSE1/13)
test_RMSE1

## [1] 0.7428326
```

SARIMA model has a validation RMSE of 0.7428, implying that the predictions made by this model would be within (plus or minus) 0.7428 of true TB incidence per 100,000 population. This is a good performance.

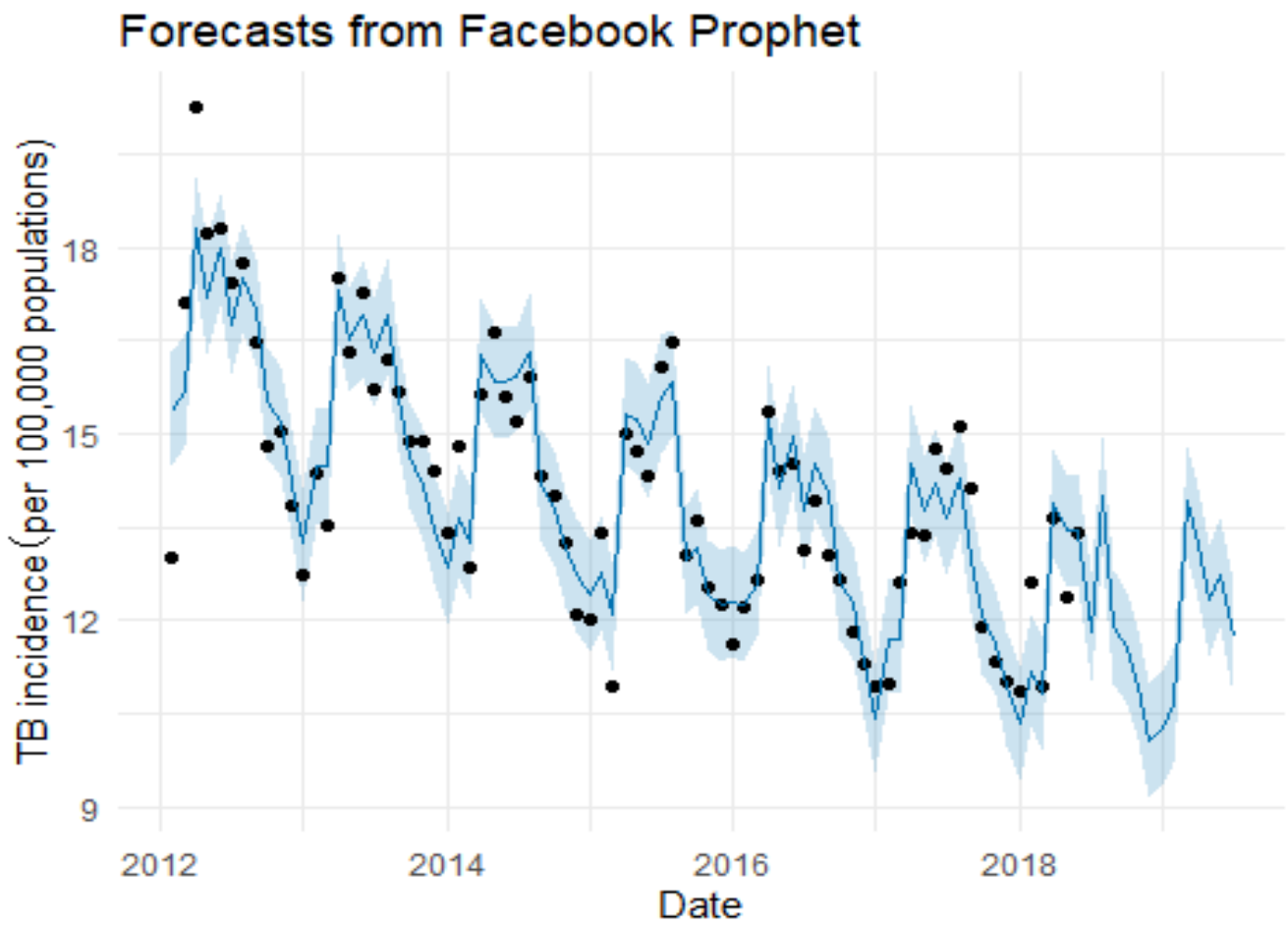
2.Forecast Using Prophet

Prophet is a robust and an open-source time series algorithm that was developed by data scientists at Facebook in 2017 to help business users with a powerful and easy-to-use tool for forecasting business results (Khare, 2023). Prophet is capable of effectively handling non-linear trends, seasonality, holidays and sudden changepoints, missing values and outliers. The algorithm works well with seasonal data that can be described by an additive model, and can be applied to various fields such as public health, meteorology, agriculture (Rao, 2024).

```
# First prepare the training data for prophet
data <- training_data |> mutate(ds = as.Date(Time),
                                y = `TB incidence (per 100,000 populations)`)
|>
  select(ds, y)

# Fit the Prophet model
m <- prophet(data)

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

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

# Make a data frame with future dates for Forecasting
future <- make_future_dataframe(m, periods = 13, freq = "month",
include_history = T)
```

It is important to note that prophet's make_future_dataframe() function uses fixed number of days for each month i.e. 31 days, and does not consider the fact that some months have 30 days, while February has 28/29 days. However, I'll make sure that the length of my future dates (months) for forecasting is same as those in the test data.

```
# Use the Prophet model to make predictions
prophet_forecasts <- predict(m, future)

# Plot the forecast
p <- plot(m, prophet_forecasts,
     ylabel = "TB incidence (per 100,000 populations)",
     xlabel = "Date")
p + labs(title = "Forecasts from Facebook Prophet") + theme_minimal()
```



The out of sample forecast by prophet closely follows the pattern of the series.

Evaluate Performance of the Prophet model on test data

```
# Collect predictions
preds <- prophet_forecasts[78:90,16]

# Find test SSE and RMSE
test_SSE2 <- sum((test_data$`TB incidence (per 100,000 populations)` -
preds)^2)
test_SSE2

## [1] 22.36664

test_RMSE2 <- sqrt(test_SSE2/13)
test_RMSE2

## [1] 1.311683
```

The prophet model has a validation RMSE of 1.3117, implying that the predictions made by this model would be within (plus or minus) 1.3117 of true TB incidence per 100,000 populations. The prophet model doesn't perform quite well, it's outperformed by the Seasonal ARIMA model.

References

Coghlan, A. (2018, September 10). A little book of R for time series (Release 0.2). https://readthedocs.org/projects/a-little-book-of-r-for-time-series/downloads/pdf/latest/

GreeksforGreeks. (2020, July 22). Time series analysis using Facebook Prophet in R programming. https://www.geeksforgeeks.org/time-series-analysis-using-facebook-prophet-in-r-programming/

GeeksforGeeks. (2023, October 20). Time series decomposition techniques. GeeksforGeeks. https://www.geeksforgeeks.org/time-series-decomposition-techniques/

Khare, P. (2023, May 13). Understanding FB Prophet: A time series forecasting algorithm. ILLUMINATION. https://medium.com/illumination/understanding-fb-prophet-a-time-series-forecasting-algorithm-c998bc52ca10.

Rao, V. (2024). Prophet for enterprise time series forecasting. LinkedIn. https://www.linkedin.com/pulse/prophet-enterprise-time-series-forecasting-vasu-rao-md6fc#:~:text=Assumption%20of%20Additive%20Seasonality%3A%20Prophet,multiplicative%20seasonality%20is%20more%20appropriate.

Zheng, Y., Zhang, L., Wang, L. et al. Statistical methods for predicting tuberculosis incidence based on data from Guangxi, China. BMC Infect Dis 20, 300 (2020). https://doi.org/10.1186/s12879-020-05033-3