Anomaly Detection of Melbourne Pedestrian Count

Student

Yiying Zhu

S3806350@student.rmit.edu.au

Supervisor

Sevvandi Kandanaarachchi

sevvandi.kandanaarachchi@rmit.edu.au

Nov 2020

Abstract

Pedestrian mobility patterns indicate the vitality of a city and provides valuable insights for urban planning. In this paper, we utilise the pedestrian counting data of the City of Melbourne and model the normal pedestrian patterns for multiple locations. Since the time series show multiple seasonality patterns, dynamic harmonic regression and TBATS methods are used to forecast one-week data in advance. Forecasting errors are then used to detect anomalies for special events and public holidays. The results found using forecasting techniques are consistent with those produced by other anomaly detection method. The experiment shows the forecasting models proposed in the study are capable of detecting anomalous events across multiple locations.

Keywords

forecasting, anomaly detection, sensor networks

1 Introduction

With the rapid growth of human population in Melbourne, it is critical for city council to provide sustainable infrastructure and public facilities to engage citizens in moving across the urban area. Understanding the patterns of pedestrian movement is valuable for urban resource planning and allocation to cope with growing pedestrian demands in the long-term. The high pedestrian activities in a walking-friendly environment often indicates the vibrancy and economic prosperity of the city. Moreover, a significant number of pedestrians is influenced by various factors (e.g. weather, public holidays, special events etc.). The locations where exist prominent high human density may require update of resources and transportation services. Situations where experience a sudden increase or decrease of pedestrian numbers require careful investigation. If the mobility change is caused by special events or public holidays (e.g. Australian Open, Moomba Festival), their impacts need to be considered when planning the public services during that period.

In this study, the pedestrian counting system data provided by City of Melbourne (City of Melbourne, 2020) is used. By analysing data from 2018 to 2019, we aim to model the normal mobility flows across multiple sensor locations using forecasting techniques, which can help city planners to predict and target city services according to demand. We also aim to detect anomalous events using the prediction model, which can be used to predict the impact of these events.

This raises several research challenges:

(1) The time series data shows seasonality for both daily and weekly patterns, therefore simple forecasting models (e.g. ARIMA) might not be able to forecast accurately.
(2) The performance evaluation of the models on only one test set might be biased by randomness in the data, hence more sophisticated evaluation techniques are required.
(3) Data collected in different sensor locations respond to special events differently. For example, Flinders Street Station shows the mobility pattern of people going to work and universities clearly, so it is sensitive to the pedestrian movements during public holidays. But it might not be able to capture such changes for sporting events held in the Etihad Stadium.

To tackle the challenges, this study presents the use of dynamic harmonic regression (Hyndman & Athanasopoulos, 2018) and TBATS (De Livera et al., 2010) methods to model the time series data with multiple seasonality. Rolling window forecasting (Hyndman & Athanasopoulos, 2018), a sophisticated model evaluation method, is then used to select the model with best performance. In addition, this study selects two sensor locations – Flinders Street Station and Princes Bridge to accommodate the needs to detect anomalies for different types of special events. The results found by using the forecasting techniques are validated with other anomaly detection technique.

The rest of the report is organised as follows. Section 2 provides the related work in analysing Melbourne pedestrian movements. Section 3 provides an extensive analysis of the pedestrian counts data across multiple locations. In Section 4, the methodology of anomaly detection is presented. Section 5 describes the forecasting and model evaluation techniques used in our experiment. The results and validation of anomaly detection are presented in Section 6. Finally, we conclude our findings and future work suggestions in Section 7.

2 Literature Review

Several studies have analysed the Melbourne pedestrian dataset from different perspectives. Doan et al. (2015) modelled the normal behaviors of mobility patterns and detected the anomalous events from pedestrian counting data. The data were clustered into load profiles characterising major activities throughout the day. The research used an Ensemble Switching Model, which is a dynamic anomaly detection technique to deal with systems which switch between different states over long-time spams. They compared the results with those produced by HyCARCE, a static clustering model. The paper showed the Ensemble Switching Model produced more accurate results in terms of detecting special events. The limitation of the research is it did not provide a general solution to predict the normal and special pedestrian patterns in the Melbourne CBD.

Another study conducted by X. Wang et al. (2017) focused on forecasting the normal pedestrian patterns for the future. Unpredictable events such as traffic accidents and natural disasters and public holidays were filtered out in this research. This paper proposed an ARIMA model to model both weekday and weekend pedestrian counts, which was capable of predicting pedestrian counts up to 16 days in advance. The model outperformed other state-of-arts models for short-term prediction. However, this research only proposed the prediction model for sensor locations in the Southern Cross area in Melbourne during normal weekdays and weekends. It did not model different patterns for special events such as major football events and public holidays.

The studies above did not further investigate the mobility patterns for different sensors. They did not analyse the association between sensors, which indicates how people walk around the city throughout the day. In this study, we analyse the data for multiple sensors to find how mobility patterns change throughout the day across multiple locations. In addition, we aim to combine the techniques of the two studies to perform anomaly detection on the time series for different sensors by using the forecasting techniques, which allows us to know the exact amount of underestimation or overestimation produced by the models.

3 Data Description

3.1 Melbourne Pedestrian Counts Dataset

The pedestrian counts used in this research are sourced from pedestrian counting system provided by City of Melbourne (City of Melbourne, 2020). The open dataset provides hourly pedestrian counts since 2009 from non-vision based sensors located across Melbourne Central Business District (CBD). There are more than 69 sensors installed within the range of Melbourne city.

A sensor is installed under an awning or on a street pole to form a counting zone on the footpath below. It records all multi-directional pedestrian movements through the zone throughout the day. The data is stored in the onsite data logger and transferred to the central server every 10 to 15 minutes via wireless transmission system, which is then uploaded to the counting system every hour. The sensor locations are selected based on three criteria – retail and event activity, regular pedestrian use and the egress and entry flow to these areas. These sensors only record human movements, not images, so no individual information is collected.

In this paper, an R package rwalkr (Wang, 2020) is leveraged to provide APIs to the Melbourne pedestrian counts data in tidy data form. The *Socrata* API powers the melb_walk_fast() function, where counts are uploaded on a monthly basis. If a selection of sensors is of interest, melb_walk_fast() function provides the option for sensor choices.

The tidy dataset is structured with the following information:

- Sensor: sensor location
- Date_Time: date and time when the pedestrian counts are recorded
- Date: date associated with Date_Time
- Time: time of the day
- Count: hourly pedestrian counts

3.2 Analysis of Pedestrian Counts Data

To analyse the dynamics and detect anomaly in pedestrian patterns, two-year data from 2018 to 2019 have been collected. Before we begin training the data, we take a closer look at sensors around train station, retail and activity event area. The sensor locations are at Bourke Street Mall (North), Flinders Street Station Underpass, Princes Bridge, Southern Cross Station and Southbank.

Figure 1 and Figure 2 show the weekly and daily mobility patterns across 5 sensors in the year of 2018. Each line represents the observed data of a particular week. In figure 1, different colour represents different seasonality trends of the weekly data. In figure 2, the red line represents daily pedestrian counts in weekdays, and the green line indicates daily mobility during weekends.

We observe the pedestrian mobility for Flinders and Southern Cross stations is similar. The pedestrian counts in weekdays (Monday – Friday) are much higher than the number in weekends. During weekdays, the first peaks are around 7:00 to 9:00 reaching to over 4000 people and the second peaks are around 17:00 to 18:00 reaching to approximately 5000 to 6000 people at both stations. This is possibly caused by people arriving and leaving the office or university for and from work or study in the morning and afternoon. There is a slightly increase in pedestrian mobility from 12:00 to 13:00 in the midday. This is due to people may start to seek for food around the station during the lunchtime.

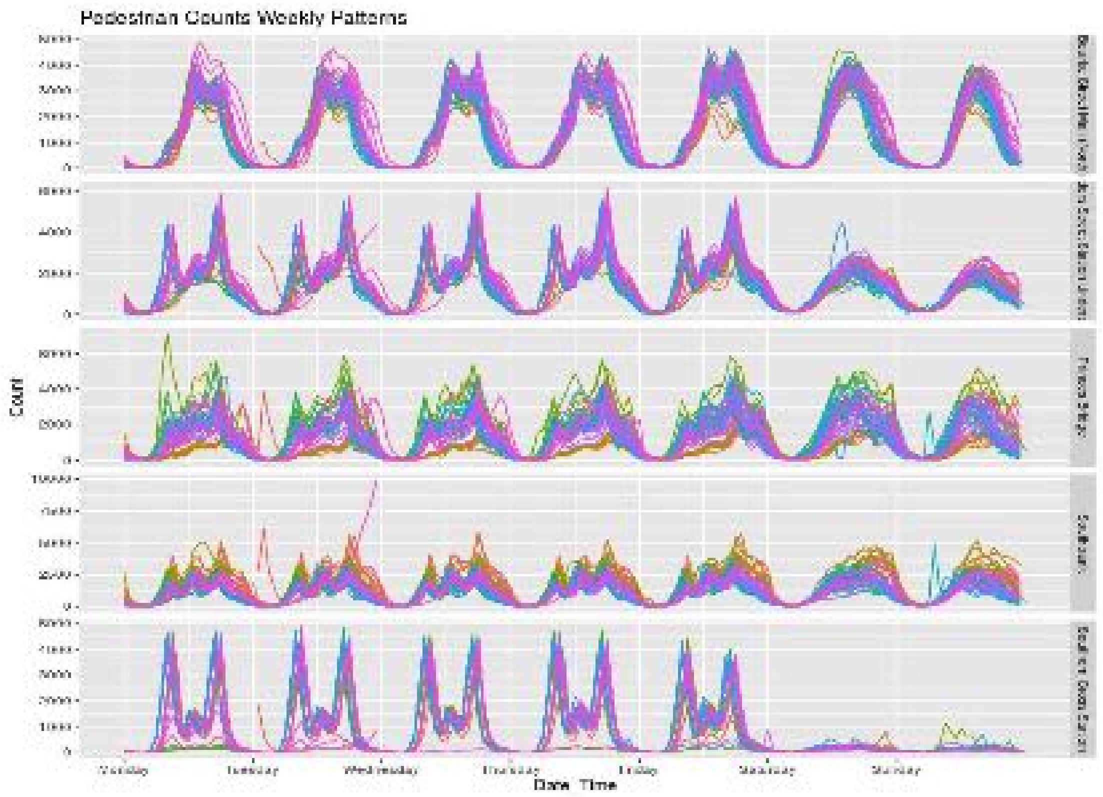


Figure 1 Weekly Patterns of Pedestrian Mobility

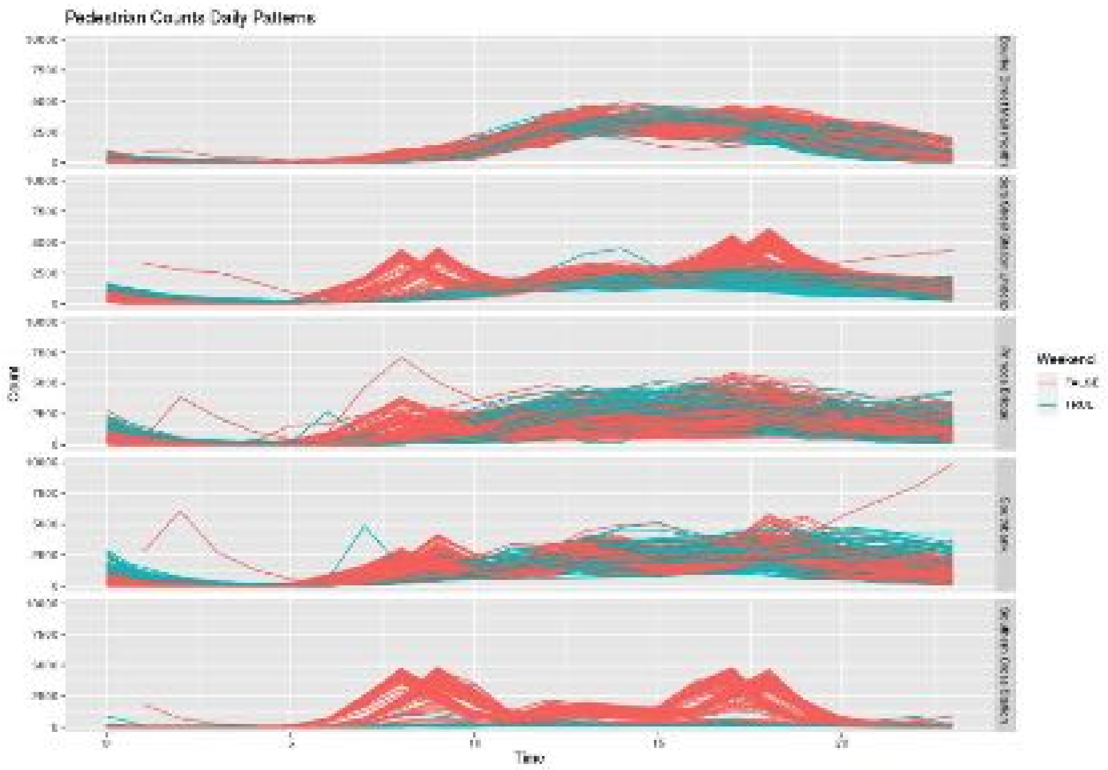


Figure 2 Daily Patterns of Pedestrian Mobility

During weekends, the mobility at Flinders Station shows maximum counts in the afternoon from 12:00 to 20:00. People tend to have weekend activities during afternoon and evening. However, nearly few people travel to Southern Cross Station on Saturdays and Sundays. This can be explained that Southern Cross Station is surrounded office area and people travel from suburbs to the station for work during weekdays.

From the observation, the plots for sensors at Bourke Street, Princes Bridge and Southbank show there is no large decrease in mobility during weekends. This is possibility because these areas are close to shopping malls and outdoor activity areas. People tend to walk around these locations during lunch breaks, after work in weekdays and in weekends' afternoons. While there is a normal mobility trend for each sensor repeating weekly, the data is relatively noisy with sudden increases and decreases compared to normal pedestrian counts. The noises may be caused by events such as public holidays and special events. Unpredictable factors such as weather and traffic accidents may also result in the anomalies.

Since the mobility trends of Flinders Station and Princes Bridge are representatives of distinct travelling trends for working and recreational activities, they may respond to special events differently. The data of Flinders Station are more sensitive to the decrease in public holidays, while the data of Princes Bridge show the anomalies during special events such as Moomba Festival more clearly. In this study, we select the data of the two sensors from 2018 to 2019 for forecasting model training and anomaly detection so that they can be used to detect different types of events.

4 Methodology

To detect anomalies in the time series using forecasting techniques and validate our findings, this study approaches the problem in three steps.

Step 1: Train forecasting models

Since the time series plots show weekly seasonality patterns, 12 weeks of weekly data are used as training data. We select the weekly data from May 2018 to August 2018, which are used to estimate parameters of forecasting models. We focus on 2 adjacent sensors – Flinders Street and Princes Bridge respectively. 5 weeks of data post training data period are used as testing data to evaluate the accuracy of trained models. To avoid the performance of forecasting models being biased by noises in the data, the weekly data containing public holidays such as Queen's Birthday and anomalies in both training and testing data are filtered out.

The purpose of this step is using forecasting techniques to construct candidate models. The model selection process uses sliding window rolling forecast approach. The criteria to select model is based on evaluation metrics that measures the predictive performance (refer to section 5).

Step 2: Use the trained model to forecast weekly data containing special events

In this step, we aim to forecast the pedestrian counts for the week containing public holidays or special events. Previous four-week data to the week including special events are required as the input for the forecasted models constructed in the previous step. The output is the forecasted pedestrian numbers in the week having special events. Forecast errors are the differences between observed values and its predicted values. Based on this definition, if the forecast error at a particular time is relatively large compared to others, this may suggest anomalies in the time series. In this study, we select two events – Christmas holidays and Moomba Festival to validate this detection of anomaly approach using forecasting techniques.

Step 3: Compare the results with other anomaly detection techniques

In this step, we aim to use other anomaly detection method on the data which include Christmas holidays and Moomba Festival. Subsequently, the results are compared with the findings in step 2 to test if the anomaly detection using forecasting techniques is consistent with other method.

In this paper, we use the anomalize package in R (Dancho & Vaughan, 2020) to perform anomaly detection in the time series. This package performs anomaly detection on remainders from the time series by removing both seasonal components and trend components. Seasonal components are cyclic pattern occurring a daily and weekly cycle for hourly Melbourne pedestrian counts data. Trend components are the longer-term growth over many observations. Seasonal decomposition is used in anomalize to deconstruct the time series and produce residuals.

5 Forecasting Models

As discussed in section 3, the hourly pedestrian data shows two seasonality patterns. There is a daily seasonal pattern with frequency 24 (there are 24 hours per day), and a weekly seasonal pattern with frequency 24 x 7 = 168. To model the time series with multiple seasonality, we use two techniques - dynamic harmonic regression and TBATS models. The candidate models are evaluated based on the average performance metrics using sliding window approach.

5.1 Dynamic Harmonic Regression Model

Dynamic harmonic regression model is a regression model combining Fourier terms with ARMA errors (Hyndman & Athanasopoulos, 2018). The model allows time series with any length seasonality, while other seasonality models such as ARIMA and ETS approaches are more suitable for short frequencies such as 4 periods for quarterly data and 12 periods for monthly data. The dynamic harmonic regression also allows time series with multiple seasonality. The pedestrian data used in this paper has both daily and weekly seasonality over 12 weeks, so the dynamic harmonic regression model is more appropriate to model the complexities in our dataset. With multiple seasonalities, Fourier terms of both frequencies need to be added, which are of the form

$$\sin(\frac{2\pi kt}{24}), \qquad \cos(\frac{2\pi kt}{24}), \qquad \sin(\frac{2\pi kt}{168}) \ and \ \cos(\frac{2\pi kt}{168})$$

for K = 1,2, ... K is the number of Fourier sin and cos pairs, which controls the smoothness of the seasonality patterns. The smaller the value of the K, the smoother the seasonal patterns are. The dynamic harmonic regression model is fitted with an ARMA error structure. We vary K from K = 1 to K = 12, and choose the one to minimise the AICc value. A log transformation is used to ensure the forecasts and prediction intervals be positive. However, the disadvantage of the dynamic harmonic model is that the seasonality is assumed to be fixed over time. This means the seasonal patterns are not allowed to change over time. But as observed in the plots of time series (Figure 1 and Figure 2), seasonality does not change significantly and remains relatively constant in the three-month period, so this is not a big disadvantage to our data.

5.2 TBATS Model

TBATS (De Livera et al., 2010) an acronym for the characteristics of the model:

- T represents trigonometric seasonality.
- B indicates Box-Cox transformation.
- A indicates ARIMA errors.
- T represents the global and local trend.
- S stands for seasonal components.

TBATS model combines Fourier terms with an exponential smoothing state space model and a Box-Cox transformation in an automated process. Compared to dynamic harmonic regression model, a TBATS model allows the seasonality to change slowly over time. However, it can be relatively slow to estimate the TBATS model, especially for a long time series. This is not a significant issue in this study since only 12 weeks of hourly data are extracted from the database as the training data.

5.3 Model Evaluation

A model which fits the training data well will not necessarily predict well, so the size of residuals cannot reliably indicate the genuine forecast errors. When evaluating forecast accuracy and comparing between candidate models, the common practice is to partition the data into training and test data. We use 12 weeks of training data to fit the dynamic harmonic regression and TBATS models, and one week of observed data post the date of training data is used as test set to evaluate the accuracy of the models. Since the test data is not used to predict one week of data in advance, it should be reliable to indicate how well a model will forecast on new data.

The performance metrics used in this study is RMSE (Root Mean Squared Error) and MAPE (Mean Absolute Percentage Error). RMSE (Hyndman & Koehler, 2006) is the scale-dependent measure based on squared errors, which is expressed as

$$\text{RMSE} = \sqrt{mean({e_t}^2)}$$

MAPE (Hyndman & Koehler, 2006) is the percentage error, which is unit-free, and so it is widely-used to compare forecast performances between datasets. It is defined as

$$\text{MAPE} = \text{mean}(\left|\frac{100e_t}{y_t}\right|)$$

A model which performs prediction well on one test data may not predict accurately on another set of test data. It is more reliable to use more test sets to evaluate the performance of the models, so a more sophisticated version of holding out sample is used in this study, which is also known as rolling window forecast (Hyndman & Athanasopoulos, 2018). There are a series of test sets, each consisting of observations of one-week data. In this study, we use the sliding window approach the procedure of which is illustrated in Figure 3. A fixed window size of 12 weeks of hourly data is shifted through the data to fit the model and forecast one week in advance. Once one run is completed, we move the window by stepping size of one week and start the next run. The window is rolled five times on 17 weeks of historical data. The forecast accuracy is calculated by averaging over the test sets. The model which minimises the average of RMSE and MAPE is chosen as the best model for prediction.

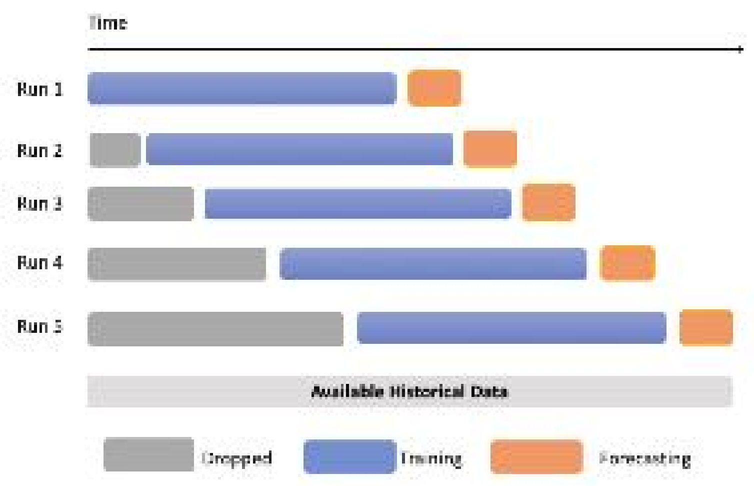


Figure 3 Evaluation on Rolling Window Forecasting

5.4 Results

5.4.1 Model for Flinders Street

For the historical data of Flinders Street, we estimate various dynamic harmonic regression models. We firstly estimate models by using auto.arima () function in R to automatically generate parameters of the models from K =1 to K = 12. To further assess this model, we overfit another 4 dynamic harmonic models. Table 1 shows the regression model with ARIMA errors (2,1,6) and 12 pairs of seasonality patterns is the one which minimises the AICc value.

Then we estimate the parameters of TBATS model, and compare its average performance metrics among rolling 5 test datasets with the values of the estimated dynamic harmonic regression model. Table 2 shows the three models has similar average MAPE values, but TBATS model generates the lowest average RMSE value. This suggests TABTS model outperforms dynamic harmonic regression models for the data of Flinders Street .

5.4.2 Model for Princes Bridge

For the historical data of Princes Bridge, we estimate various dynamic harmonic regression models as well. Table 3 shows the regression model with ARIMA errors (4,1,3) and 12 pairs of seasonality patterns is the one which minimises the AICc value.

Then we estimate the parameters of TBATS model, and compare its average performance metrics with the those of the estimated dynamic harmonic regression model. Table 4 shows the estimated TBATS has lower average RMSE values than dynamic harmonic regression model although the dynamic harmonic regression models generate the relatively lower average MAPE value. This suggests the TABTS model is selected to forecast one week of data in advance.

K	ARIMA errors	AICC
K=1	ARIMA(3,0,3)	1951.44
K=2	ARIMA(3,0,4)	1657.27
K=3	ARIMA(3,0,2)	1517.55
K=4	ARIMA(4,0,3)	1371.8
K=5	ARIMA(2,0,5)	1099.3
K=6	ARIMA(5,0,2)	905.99
K=7	ARIMA(5,0,1)	901.74
K=8	ARIMA(2,1,4)	541.92
K=9	ARIMA(2,1,5)	484.11
K=10	ARIMA(2,1,5)	459.24
K=11	ARIMA(2,1,2)	403.29
K=12	ARIMA(2,1,5)	333.99
K=12	ARIMA(2,1,4)	340.43
K=12	ARIMA(2,1,6)	288.70
K=12	ARIMA(3,1,5)	343.64
K=12	ARIMA(1,1,5)	684.33

Table 1 AICc values with respect to dynamic harmonic regression models for Flinders Street

	RMSE	MAPE
TBATS model (0.485, {4,2}, -, {<24,11>, <168,6>})	461.2994	37.2501
Dynamic harmonic regression model errors with ARIMA (2,1,6), K = 12	531.7028	36.7445
errors with ARIMA (2,1,5), K = 12	534.0123	36.9425

Table 2 Performance metrics of estimated models for Flinders Street

K	ARIMA errors	AICC
K=1	ARIMA(5,0,1)	1435.27
K=2	ARIMA(3,0,0)	1285.88
K=3	ARIMA(2,0,1)	1134.92
K=4	ARIMA(4,0,5)	713.92
K=5	ARIMA(5,0,1)	602.35
K=6	ARIMA(2,1,2)	1028.15
K=7	ARIMA(2,1,2)	1029.32
K=8	ARIMA(4,1,3)	373.54
K=9	ARIMA(4,1,2)	285.37
K=10	ARIMA(5,1,2)	254.02
K=11	ARIMA(5,1,3)	221.22
K=12	ARIMA(4,1,2)	187.35
K=12	ARIMA(4,1,3)	80.43
K=12	ARIMA(4,1,1)	230.85
K=12	ARIMA(5,1,2)	182.15
K=12	ARIMA(3,1,2)	192.43

Table 3 AICc values with dynamic harmonic regression models of Princes Bridge

	RMSE	MAPE
TBATS model (0.002, {2,2}, -, {<24,6>, <168,6>})	394.3057	35.5494
Dynamic harmonic regression model errors with ARIMA (4,1,2), K = 12	377.8467	28.9708
errors with ARIMA (4,1,), K = 12	486.0918	27.2336

Table 4 Performance metrics of estimated models for Princes Bridge

6 Anomaly Detection

In this section, we aim to detect anomalous events in the time series by using the estimated models to generate the forecast errors. Because forecast errors are the differences between observed historical data and the predicted values, anomalous events are suggested when the plot of forecast errors shows large successive differences. We aim to use this approach to detect the anomaly patterns for two special events in this study, one the Christmas holidays, the other the Moomba Festival. The findings will be validated with the results from anomalize package (Dancho & Vaughan, 2020) in R, which is another anomaly detection method applied to remainders by removing seasonal patterns and trends in the time series.

6.1 Christmas

Since people travel to Flinders Street Station for working during weekdays and recreative activities such as go shopping during weekends, the data of Flinders Street is sensitive to mobility changes in public holidays. Accordingly, the model for Flinders Street is used to detect anomalies in the time series during Christmas breaks. Historical data from 12 November 2018 to 30 December 2018 is selected to perform anomaly detection. As illustrated in Figure 4, we use a fixed rolling window of four weeks to refit the estimated TBATS model, and forecast one week in advance. The window is then moved by one week ahead for another run. Unexpected changes in the data which are used to refit the model are filtered out.

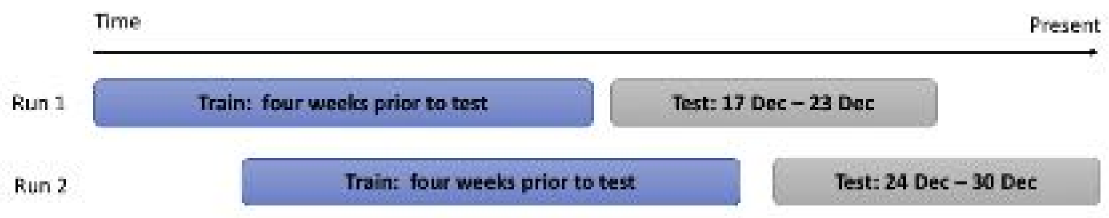
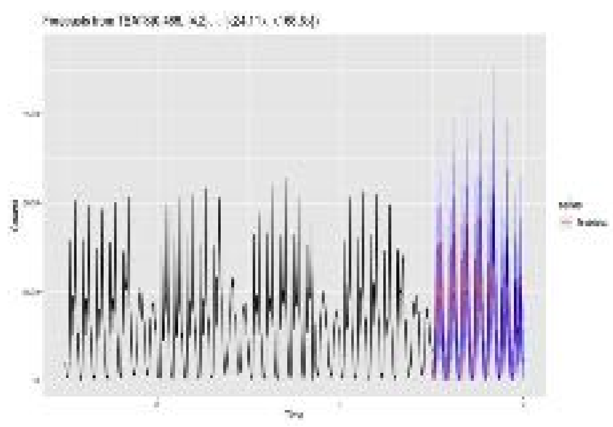
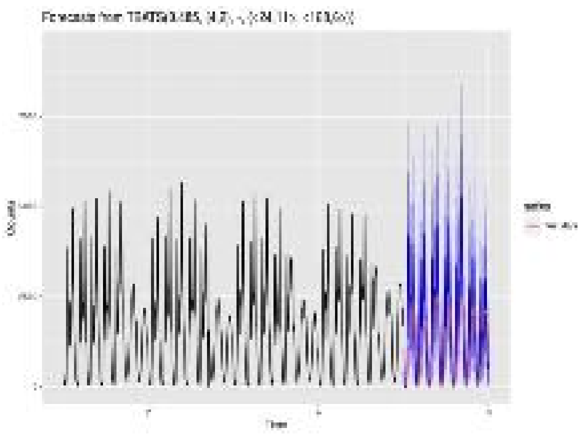


Figure 4 Sliding window forecast for Christmas holidays

Figure 5 shows the forecasted data and observed data from 17 December 2018 to 30 December 2018. We observe the forecasted values capture the mobility patterns well in Figure 5 (a), but fail to predict the relatively low pedestrian counts in Figure 5 (b). This is clearer when plotting the forecast errors for the test data sets. Figure 6 shows the forecast errors are between 0 to 1000 people from 17 December to 22 December, which is relatively low. The forecast errors significantly increase to approximately 2000 to 3000 people from 23 December to 30 December. The successive large differences between observed historical data and predicted data suggest the changing mobility pattern during the Christmas break. This is consistent with the anomalies detected by using anomalize package in R. We observe continuous red dots marked in the decomposition of time series in Figure 7, which indicate anomalous events in the last week of December. The decrease in pedestrian counts is possibly because people tend to go travelling or stay home during the long break.



(a)



(b)

Figure 5 (a) Forecasted data and observed data for Christmas holidays from 17 Dec to 23 Dec
(b) Forecasted data and observed data for Christmas holidays from 24 Dec to 30 Dec

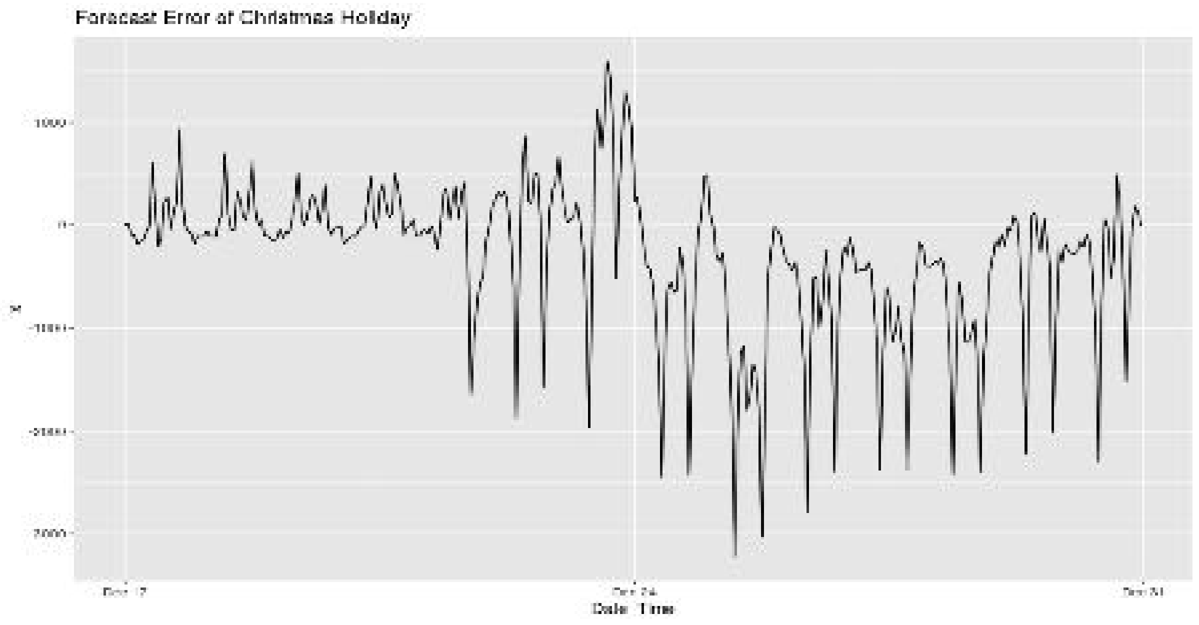


Figure 6 Forecast errors for Christmas holidays

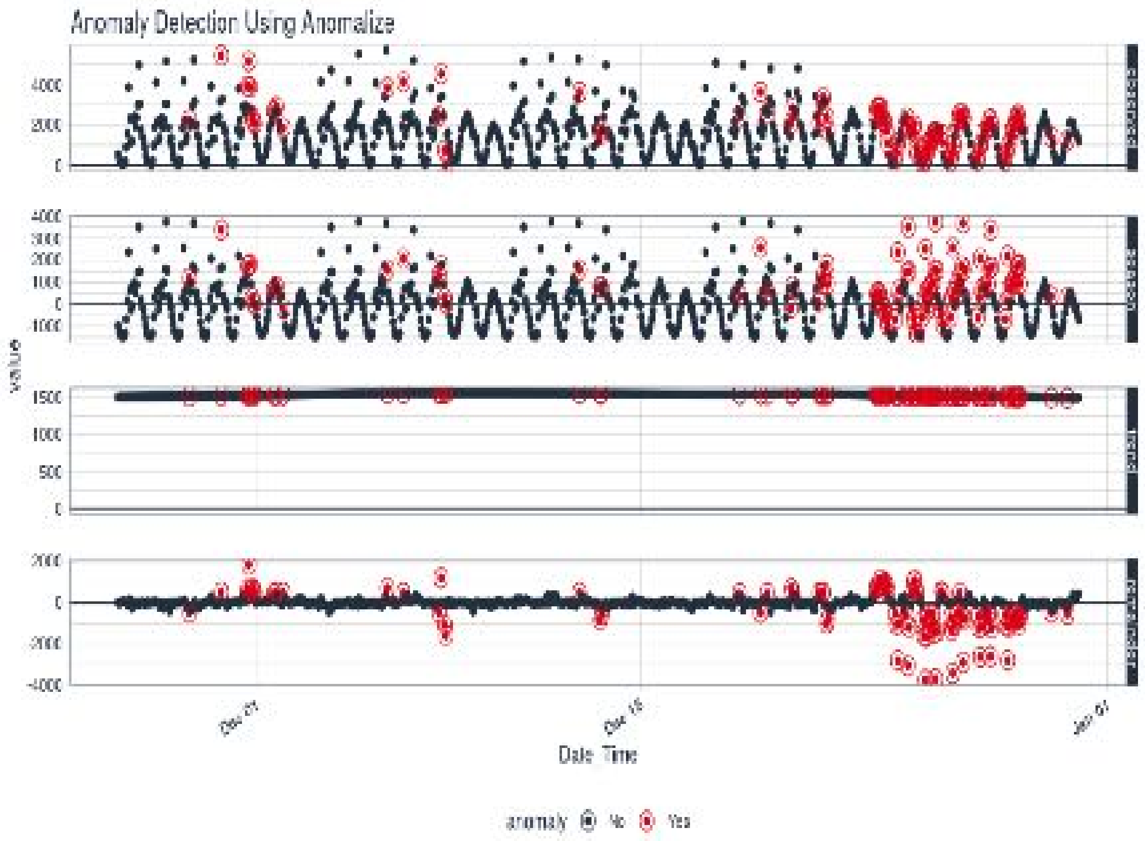


Figure 7 Anomalies detected by anomalize for Christmas holidays

6.2 Moomba Festival

Princes Bridge is located close to Flinders Street Station representing the pedestrian movement between Melbourne CBD and South Melbourne. People walk there for recreational activities such as going to events held at Southbank and National Art Gallery, so the data of Princes Bridge is sensitive to different mobility patterns during special events. Based on this fact, the model for Princes Bridge is used to detect anomalies during Moomba Festival (8 March 2019 to 11 March 2019). Historical data from 5 February 2019 to 11 March 2019 is extracted by filtering out anomalous patterns in this period. As illustrated in Figure 7, four weeks of data before the week of Moomba Festival are used to refit the estimated dynamic harmonic regression model, and forecast one week in advance. The observed data is compared with the predicted value, and the forecast error is plotted.

Figure 9 and Figure 10 shows the forecast errors largely increases from approximately 500 people to nearly 3000 to 4000 pedestrian counts during the four days of Moomba Festival. From Figure 11, we observe the anomalies found using anomalize package are successive from 8 March to 11 March, which is consistent with the findings from the forecasted model. The reason is more people walk to the CBD and Southbank area for Moomba Festival, the model of normal daily and weekly patterns cannot capture the sudden increases of the mobility.

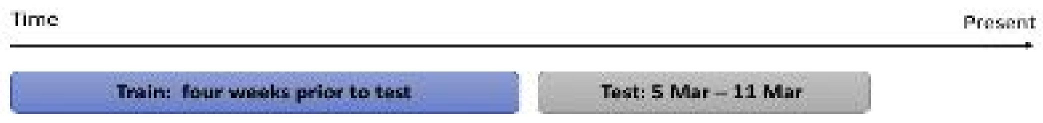


Figure 8 Forecasting training and testing data for Moomba Festival

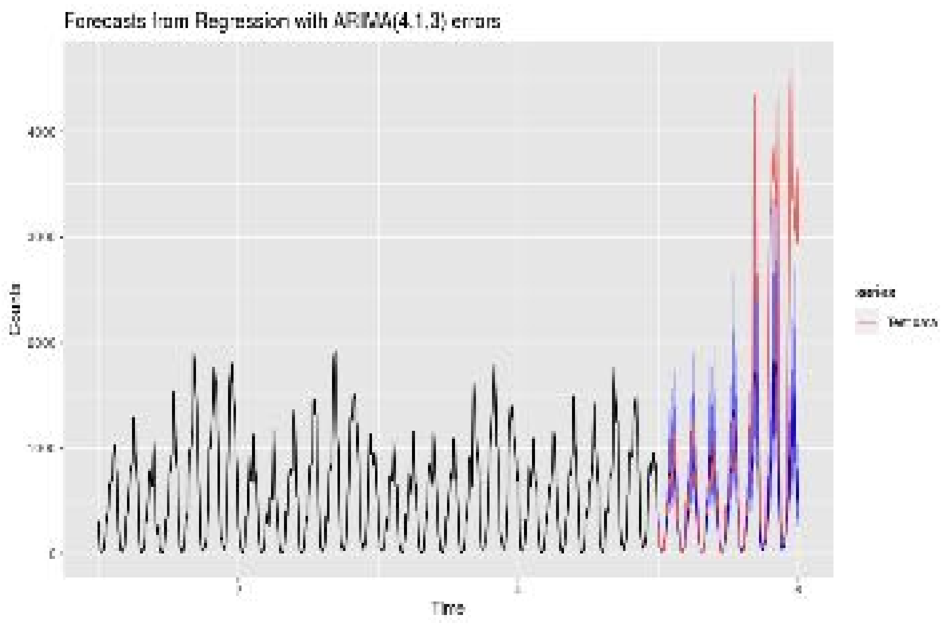


Figure 9 Forecasted data and observed data for Moomba Festival

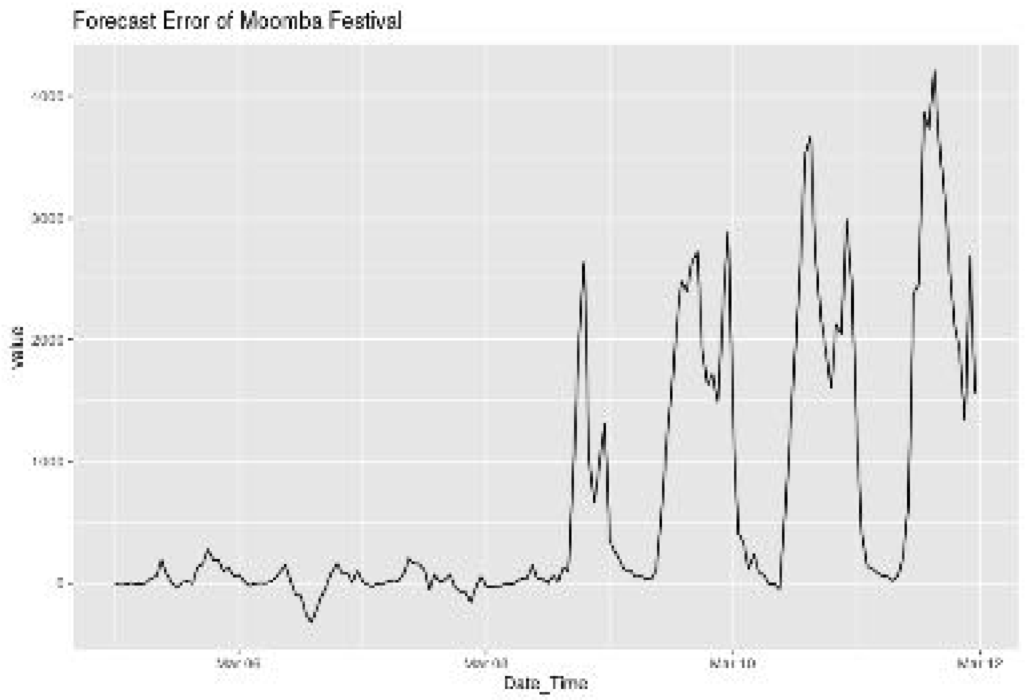


Figure 10 Forecast errors for Moomba Festival

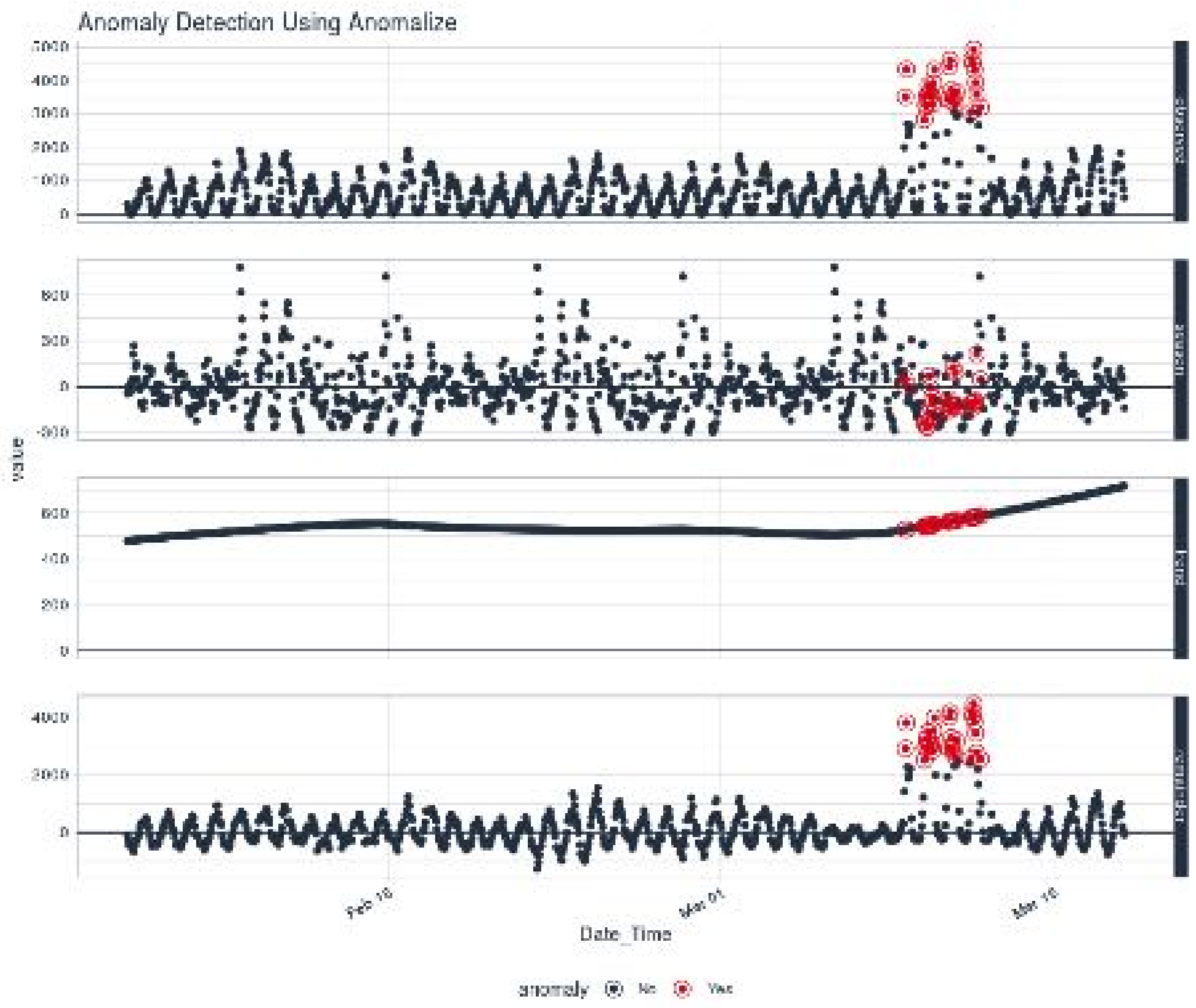


Figure 11 Anomalies detected by anomalize for Moomba Festival

7 Conclusion and Future Work

In this research, we present an anomaly detection approach which is built upon forecasting techniques. Two forecasting methods – dynamic harmonic regression and TBATS model are applied to the data for sensors located at Flinders Street Station and Princes Bridge. We choose the best model minimising the average of MAPE and RMSE values among multiple test sets using the rolling window forecasting method. The anomalous events in the time series are detected when there are sudden and successive increases of forecast errors in a period of time, which are compared with the results produced by anomalize package in R, a robust anomaly detection method by decomposing the time series. This study shows using forecasting techniques is capable of detecting anomalous events. More importantly, the forecast errors indicate the exact amounts of underestimation or overestimation for the estimated models, which can be included as dummy factors in the models to improve the accuracy of prediction for future events.

For the future work, there are several improvements that could be investigated and applied to enhance the current research. First, an integrated forecasting system which can accommodate switching between states can be further developed. The models in our current study are built upon the data from May to August, which is in winter. A whole year of data can be extracted to model the pedestrian movements across different seasons which will automatically switch between states over such a long-time span.

Secondly, ensemble techniques can be applied to both forecasting and anomaly detection methods. This approach combining different prediction and anomaly detection methods (e.g. clustering techniques, machine learning models) would possibly improve the accuracy of predicting future data and detecting unexpected changes in the time series.

Furthermore, the pedestrian movement between different locations throughout the day can be explored. The spatial-temporal analysis of the mobility across the city would be important to improve the current anomaly detection approach, which would allow City of Melbourne to better understand how people moves around the city in normal days and special events so that they can target plan and allocate services according to the prediction in the future. In addition, clustering techniques for multivariate series can be used to group sensors with similar mobility patterns throughout the day. This can help to understand for what purposes people travel to locations across the city.

8 Acknowledgement

I thank Sevvandi Kandanaarachchi for kindly providing valuable guidance and support to this minor thesis research.

References

City of Melbourne. (2020). *Pedestrian Counting System – 2009 to Present (counts per hour)*. https://data.melbourne.vic.gov.au/Transport/Pedestrian-Counting-System-2009-to-Present-counts-/b2ak-trbp?src=featured_banner.

Dancho, M., & Vaughan, D. (2020). *anomalize: Tidy Anomaly Detection*. https://cran.r-project.org/web/packages/anomalize/index.html.

De Livera, A. M., Hyndman, R. J., & Snyder, R. D. (2010). Forecasting time series with complex seasonal patterns using exponential smoothing Forecasting time series with complex seasonal patterns using exponential smoothing. *Monash University Working Paper*, *October*, to appear. http://www.buseco.monash.edu.au/depts/ebs/pubs/wpapers/%0Ahttp://www.buseco.monash.edu.au/ebs/pubs/wpapers/2009/wp9-09.pdf.

Doan, M. T., Rajasegarar, S., Salehi, M., Moshtaghi, M., & Leckie, C. (2015). Profiling pedestrian distribution and anomaly detection in a dynamic environment. *International Conference on Information and Knowledge Management, Proceedings*, *19-23-Oct*-(October), 1827–1830. https://doi.org/10.1145/2806416.2806645.

Hyndman, R. J., & Athanasopoulos, G. (2018). *Forecasting: principles and practice* (2nd ed.). OTexts, Melbourne, Australia.

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting*, *22*, 679–688. https://robjhyndman.com/publications/another-look-at-measures-of-forecast-accuracy/

Wang, E. (2020). *rwalkr: API to Melbourne Pedestrian Data*. https://cran.r-project.org/web/packages/rwalkr/index.html.

Wang, X., Liono, J., McIntosh, W., & Salim, F. D. (2017). Predicting the city foot traffic with pedestrian sensor data. *ACM International Conference Proceeding Series*, 1–10. https://doi.org/10.1145/3144457.3152355

Appendices (R Code)

```
graphics.off() # This closes all of R's graphics windows.
rm(list=ls())

library(rwalkr)
library(forecast)
library(feasts)
library(tsibbledata)
library(dplyr)
library(ggplot2)
library(lubridate)
library(tsibble)
library(fpp3)
library(fable)
library(sugrrants)


#===================Time Series Plot =====================#

# Provides API using Socrata to Melbourne pedestrian sensor locations.
sensor <- pull_sensor(app_token = NULL)


# Pedestrian data across 5 sensors
walk_2018 <- melb_walk_fast(year = 2018,
                sensor = c('Flinders Street Station Underpass',"Princes Bridge",
                      "Bourke Street Mall (North)","Southern Cross Station",
                      "Southbank"),
                na.rm = FALSE, app_token = NULL)

summary(is.na(walk_2018)) # 702 missing values

walk_2019 <- melb_walk_fast(year = 2019,
                sensor = c('Flinders Street Station Underpass',"Princes Bridge",
                      "Bourke Street Mall (North)","Southern Cross Station",
                      "Southbank"),
                na.rm = FALSE, app_token = NULL)


# Convert to tsibble
ts_2018 <- as_tsibble(walk_2018, key=Sensor, index = Date_Time)
ts_2019 <- as_tsibble(walk_2019, key=Sensor, index = Date_Time)


# Time Series Plots
ts_2018 %>% gg_season(Count, period="week") + theme(legend.position = "none") + ggtitle("Seasonal plot:
Pedestrian Counts Weekly Patterns")

ts_2018 %>%
  mutate(
    Day = lubridate::wday(Date, label = TRUE),
    Weekend = (Day %in% c("Sun", "Sat"))
  ) %>%
  ggplot(aes(x = Time, y = Count, group = Date)) +
  geom_line(aes(col = Weekend)) +
  ggtitle("Pedestrian Counts Daily Patterns") +
  facet_grid(Sensor ~ .)



#ts_2019 %>% gg_season(Count, period="day") + theme(legend.position = "none") + ggtitle("Seasonal plot: Pedestrian Counts Daily Patterns")
ts_2019 %>% gg_season(Count, period="week", labels = "right") + theme(legend.position = "none") + ggtitle("Pedestrian Counts Weekly Patterns")


ts_2019 %>%
  mutate(
    Day = lubridate::wday(Date, label = TRUE),
    Weekend = (Day %in% c("Sun", "Sat"))
  ) %>%
  ggplot(aes(x = Time, y = Count, group = Date)) +
  geom_line(aes(col = Weekend)) +
  ggtitle("Pedestrian Counts Daily Patterns") +
  facet_grid(Sensor ~ .)

holiday_aus(2018, state = "VIC")
holiday_aus(2019, state = "VIC")


#=========Calender view plot to view anomalies in training data=========#
# Princes-Bridege #
ts_2018 %>% filter(month(Date, label = TRUE)=="Jan" & Sensor =='Princes Bridge') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)

ts_2018 %>% filter(month(Date, label = TRUE)=="Feb" & Sensor =='Princes Bridge') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)

ts_2018 %>% filter(month(Date, label = TRUE)=="Mar" & Sensor =='Princes Bridge') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)

ts_2018 %>% filter(month(Date, label = TRUE)=="Apr" & Sensor =='Princes Bridge') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)

ts_2018 %>% filter(month(Date, label = TRUE)=="May" & Sensor =='Princes Bridge') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)

ts_2018 %>% filter(month(Date, label = TRUE)=="Jun" & Sensor =='Princes Bridge') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)

ts_2018 %>% filter(month(Date, label = TRUE)=="Jul" & Sensor =='Princes Bridge') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)

ts_2018 %>% filter(month(Date, label = TRUE)=="Aug" & Sensor =='Princes Bridge') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)

ts_2018 %>% filter(month(Date, label = TRUE)=="Sep" & Sensor =='Princes Bridge') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)

ts_2018 %>% filter(month(Date, label = TRUE)=="Oct" & Sensor =='Princes Bridge') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)
```

```
ts_2018 %>% filter(month(Date, label = TRUE)=="Nov" & Sensor =='Princes Bridge') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)


# Flinders Street Station Underpass #
ts_2018 %>% filter(month(Date, label = TRUE)=="Jan" & Sensor =='Flinders Street Station Underpass') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)

ts_2018 %>% filter(month(Date, label = TRUE)=="Feb" & Sensor =='Flinders Street Station Underpass') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)

ts_2018 %>% filter(month(Date, label = TRUE)=="Mar" & Sensor =='Flinders Street Station Underpass') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)

ts_2018 %>% filter(month(Date, label = TRUE)=="Apr" & Sensor =='Flinders Street Station Underpass') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)

ts_2018 %>% filter(month(Date, label = TRUE)=="May" & Sensor =='Flinders Street Station Underpass') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)

ts_2018 %>% filter(month(Date, label = TRUE)=="Jun" & Sensor =='Flinders Street Station Underpass') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)

ts_2018 %>% filter(month(Date, label = TRUE)=="Jul" & Sensor =='Flinders Street Station Underpass') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)

ts_2018 %>% filter(month(Date, label = TRUE)=="Aug" & Sensor =='Flinders Street Station Underpass') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)

ts_2018 %>% filter(month(Date, label = TRUE)=="Sep" & Sensor =='Flinders Street Station Underpass') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)


#================= Data preprocessing =================#
# Select data from May to August (12 weeks) by filtering out anomalies

# ---------- Flinders Station  -----------------------
train_flinders <- ts_2018 %>% filter(Sensor =='Flinders Street Station Underpass' &
                             Date_Time>=as.POSIXlt("2018-05-07 00:00", tz="Australia/Melbourne") &
                             Date_Time<=as.POSIXlt("2018-08-05 23:00", tz="Australia/Melbourne"))


drop_rows <- train_flinders %>% filter(Date_Time>=as.POSIXlt("2018-06-11 00:00", tz="Australia/Melbourne")
&
        Date_Time<=as.POSIXlt("2018-06-17 23:00", tz="Australia/Melbourne"))

train_flinders <- setdiff(train_flinders,drop_rows)

train_flinders %>%
  mutate(
    Day = lubridate::wday(Date, label = TRUE),
    Weekend = (Day %in% c("Sun", "Sat"))
  ) %>%
  ggplot(aes(x = Time, y = Count, group = Date)) +
  geom_line(aes(col = Weekend)) +
  ggtitle("Pedestrian Counts Daily Patterns")
```

```
# Testing data for cross validation
test_flinders <- ts_2018 %>% filter(Sensor =='Flinders Street Station Underpass' &
                              Date_Time>=as.POSIXlt("2018-08-27 00:00", tz="Australia/Melbourne") &
                              Date_Time<=as.POSIXlt("2018-09-30 23:00", tz="Australia/Melbourne"))

test_flinders %>%
  mutate(
    Day = lubridate::wday(Date, label = TRUE),
    Weekend = (Day %in% c("Sun", "Sat"))
    ) %>%
  ggplot(aes(x = Time, y = Count, group = Date)) +
  geom_line(aes(col = Weekend)) +
  ggtitle("Pedestrian Counts Daily Patterns")



# test_flinders <- ts_2018 %>% filter(Sensor =='Flinders Street Station Underpass' &
#                             Date_Time>=as.POSIXlt("2018-08-27 00:00", tz="Australia/Melbourne") &
#                             Date_Time<=as.POSIXlt("2018-09-02 23:00", tz="Australia/Melbourne"))

# This is the test data used for cross validation
flinders_18 <- rbind(train_flinders, test_flinders)

# ------------- Princes Bridge ------------------------
train_pb <- ts_2018 %>% filter(Sensor =='Princes Bridge' &
                              Date_Time>=as.POSIXlt("2018-05-07 00:00", tz="Australia/Melbourne") &
                              Date_Time<=as.POSIXlt("2018-08-19 23:00", tz="Australia/Melbourne"))


drop_rows <- train_pb %>% filter((Date_Time>=as.POSIXlt("2018-06-04 00:00", tz="Australia/Melbourne") &
                         Date_Time<=as.POSIXlt("2018-06-17 23:00", tz="Australia/Melbourne"))|
                         (Date_Time>=as.POSIXlt("2018-07-09 00:00", tz="Australia/Melbourne") &
                         Date_Time<=as.POSIXlt("2018-07-15 23:00", tz="Australia/Melbourne")))

train_pb <- setdiff(train_pb,drop_rows)

train_pb %>%
  mutate(
    Day = lubridate::wday(Date, label = TRUE),
    Weekend = (Day %in% c("Sun", "Sat"))
  ) %>%
  ggplot(aes(x = Time, y = Count, group = Date)) +
  geom_line(aes(col = Weekend)) +
  ggtitle("Pedestrian Counts Daily Patterns")

test_pb <- ts_2018 %>% filter(Sensor =='Princes Bridge' &
                             Date_Time>=as.POSIXlt("2018-08-20 00:00", tz="Australia/Melbourne") &
                             Date_Time<=as.POSIXlt("2018-09-23 23:00", tz="Australia/Melbourne"))
pb_18 <- rbind(train_pb, test_pb)



#=================Forecasting: Model Fitting =================#

##### Flinders Station #####
```

```
# 1. dynamic harmonic regression #

train_flinders_msts <- msts(train_flinders$Count, seasonal.periods=c(24,24*7))

#---------------------------------------------------------------------------
# What is K here? how to choose k?
plots <- list()
for (i in seq(12)) {
  fit <- auto.arima(train_flinders_msts, xreg = fourier(train_flinders_msts, K = c(i,i)), lambda = 0,
                  seasonal = FALSE)
  plots[[i]] <- autoplot(forecast(fit,
                              xreg=fourier(train_flinders_msts, K=c(i,i), h=24*7))) +
    xlab(paste("K=",i,"   AICC=",round(fit[["aicc"]],2))) +
    ylab("")
}
gridExtra::grid.arrange(
  plots[[1]],plots[[2]],plots[[3]],
  plots[[4]],plots[[5]],plots[[6]],
  plots[[7]], plots[[8]], plots[[9]],
  plots[[10]], plots[[11]], plots[[12]], nrow=6)

#---------------------------------------------------------------------------
# automatic model found by auto.arima
fit <- auto.arima(train_flinders_msts, seasonal=FALSE, lambda = 0,
                xreg=fourier(train_flinders_msts, K=c(12,12)))

fit

fcast <- fit %>%
  forecast(xreg=fourier(train_flinders_msts, K=c(12,12), h=24*7))

fcast$mean

fcast%>%
  autoplot() +
  ylab("Counts") + xlab("Date_Time")

checkresiduals(fcast)

#accuracy(fcast, msts(test_flinders[1:168, 5], seasonal.periods=c(24,24*7), start=c(13,1)))


# Overfitting
fit1 <- Arima(train_flinders_msts, xreg = fourier(train_flinders_msts, K=c(12,12)),
            lambda = 0, seasonal = c(0,0,0), order = c(2,1,4) )

fit2 <- Arima(train_flinders_msts, xreg = fourier(train_flinders_msts, K=c(12,12)),
             lambda = 0, seasonal = c(0,0,0), order = c(2,1,6) )

fit3 <- Arima(train_flinders_msts, xreg = fourier(train_flinders_msts, K=c(12,12)),
             lambda = 0, seasonal = c(0,0,0), order = c(3,1,5) )

fit4 <- Arima(train_flinders_msts, xreg = fourier(train_flinders_msts, K=c(12,12)),
             lambda = 0, seasonal = c(0,0,0), order = c(1,1,5) )
```

```
aic <- AIC(fit, fit1, fit2, fit3, fit4)
bic <- BIC(fit, fit1, fit2, fit3, fit4)
AICc <- rbind(fit["aicc"], fit1["aicc"], fit2["aicc"], fit3["aicc"], fit4["aicc"])
aic_bic <- cbind(aic,AICc,bic)
aic_bic <- aic_bic[, c(1,2,3,5)]
aic_bic   # fit 2 gives smallest aic and bic


# Can use cross validation by rolling a fixed window to compare the performance
#------------------------------------------------------------------------
# ---- fit -----------
# Test
y <- msts(flinders_18$Count, seasonal.periods=c(24,24*7))
h <- 168
fcmat <- matrix(0, nrow=5, ncol=h)
accuracy <- matrix(0, nrow=5, ncol=7)
colnames(accuracy) <- c("ME","RMSE","MAE","MPE","MAPE","ACF1","Theil's U")

for (i in 1:5)
{
  x <- window(y, start=c(i,1), end=c(i+11,168))
  xreg <- fourier(x, K=c(12,12))
  fit <- Arima(x, xreg = xreg, lambda = 0, seasonal = c(0,0,0), order = c(2, 1, 5))
  fcmat[i,] <- forecast(fit, xreg = fourier(x, K=c(12,12), h=h))$mean
  accuracy[i,] <- accuracy(fcmat[i,], window(y, start=c(i+12,1), end=c(i+12,168)))
}

accuracy
colSums (accuracy, na.rm = FALSE, dims = 1)/5


# ---- fit2 -----------
for (i in 1:5)
{
  x <- window(y, start=c(i,1), end=c(i+11,168))
  xreg <- fourier(x, K=c(12,12))
  fit <- Arima(x, xreg = xreg, lambda = 0, seasonal = c(0,0,0), order = c(2, 1, 6))
  fcmat[i,] <- forecast(fit, xreg = fourier(x, K=c(12,12), h=h))$mean
  accuracy[i,] <- accuracy(fcmat[i,], window(y, start=c(i+12,1), end=c(i+12,168)))
}

accuracy
colSums (accuracy, na.rm = FALSE, dims = 1)/5



# 2. tbats #

tbats_flinders <- tbats(train_flinders_msts)
# tbats.components(tbats)

fc_tbats <- forecast(tbats_flinders, h = 24*7)
autoplot(fc_tbats) +
  xlab('Date') + ylab('Counts')

# accuracy(fc_tbats, window(y, start=c(13,1), end=c(13,168)))
# tscv
y <- msts(flinders_18$Count, seasonal.periods=c(24,24*7))
h <- 168
fcmat <- matrix(0, nrow=5, ncol=h)
accuracy <- matrix(0, nrow=5, ncol=7)
colnames(accuracy) <- c("ME","RMSE","MAE","MPE","MAPE","ACF1","Theil's U")

# cross validation without re-estimation
for (i in 1:5)
{
  x <- window(y, start=c(i,1), end=c(i+11,168))
  fit <- tbats(x, model=tbats)
  fcmat[i,] <- forecast(fit, h=h)$mean
  accuracy[i,] <- accuracy(fcmat[i,], window(y, start=c(i+12,1), end=c(i+12,168)))
}

# cross validation with re-estimation
for (i in 1:5)
{
  x <- window(y, start=c(i,1), end=c(i+11,168))
  fit <- tbats(x)
  fcmat[i,] <- forecast(fit, h=h)$mean
  accuracy[i,] <- accuracy(fcmat[i,], window(y, start=c(i+12,1), end=c(i+12,168)))
}

accuracy
colSums (accuracy, na.rm = FALSE, dims = 1)/5


# Forecasting Errors
Date_Time = test_flinders %>% filter(Date_Time>=as.POSIXlt("2018-08-27 00:00", tz="Australia/Melbourne")
&
                              Date_Time<=as.POSIXlt("2018-09-02 23:00", tz="Australia/Melbourne"))

error_flinders <- window(y, start=c(13,1), end=c(13,168)) - fc_tbats$mean
error_flindres <- error_flinders %>% as_tibble() %>% mutate(Date_Time = Date_Time$Date_Time)

error_flindres <- error_flindres %>% as_tsibble(index = Date_Time)

error_flindres %>% ggplot() +
  geom_line(aes(x = Date_Time, y = x)) +
  ggtitle(label = "Forecast Error of Trained Model")


##### Princes Bridge #####

# 1. dynamic harmonic regression #

train_pb_msts <- msts(train_pb$Count, seasonal.periods=c(24,24*7))

#-----------------------------------------------------------------------------
# What is K here? how to choose k?
plots <- list()
for (i in seq(12)) {
  fit <- auto.arima(train_pb_msts, xreg = fourier(train_pb_msts, K = c(i,i)), lambda = 0,
                    seasonal = FALSE)
  plots[[i]] <- autoplot(forecast(fit,
                              xreg=fourier(train_pb_msts, K=c(i,i), h=24*7))) +
    xlab(paste("K=",i,"   AICC=",round(fit[["aicc"]],2))) +
    ylab("")
}
gridExtra::grid.arrange(
  plots[[1]],plots[[2]],plots[[3]],
  plots[[4]],plots[[5]],plots[[6]],
  plots[[7]], plots[[8]], plots[[9]],
  plots[[10]],plots[[11]], plots[[12]], nrow=6)

#--------------------------------------------------------------------------
# automatic model found by auto.arima
fit_pb <- auto.arima(train_pb_msts, seasonal=FALSE, lambda = 0,
                  xreg=fourier(train_flinders_msts, K=c(12,12)))

fit_pb

fcast_pb <- fit_pb %>%
  forecast(xreg=fourier(train_pb_msts, K=c(12,12), h=24*7))

fcast_pb %>%
  autoplot() +
  ylab("Counts") + xlab("Date_Time")

checkresiduals(fcast_pb)


# Overfitting
fit_pb1 <- Arima(train_pb_msts, xreg = fourier(train_pb_msts, K=c(12,12)),
            lambda = 0, seasonal = c(0,0,0), order = c(4,1,3) )

fit_pb2 <- Arima(train_pb_msts, xreg = fourier(train_pb_msts, K=c(12,12)),
              lambda = 0, seasonal = c(0,0,0), order = c(4,1,1) )

fit_pb3 <- Arima(train_pb_msts, xreg = fourier(train_pb_msts, K=c(12,12)),
              lambda = 0, seasonal = c(0,0,0), order = c(5,1,2) )

fit_pb4 <- Arima(train_pb_msts, xreg = fourier(train_pb_msts, K=c(12,12)),
            lambda = 0, seasonal = c(0,0,0), order = c(3,1,2) )


aic <- AIC(fit_pb, fit_pb1, fit_pb2, fit_pb3, fit_pb4)
bic <- BIC(fit_pb, fit_pb1, fit_pb2, fit_pb3, fit_pb4)
AICc <- rbind(fit_pb["aicc"], fit_pb1["aicc"], fit_pb2["aicc"], fit_pb3["aicc"], fit_pb4["aicc"])
aic_bic <- cbind(aic,AICc,bic)
aic_bic <- aic_bic[, c(1,2,3,5)]
aic_bic  # fit_pb1 gives the lowest aic and bic

fc_pb1 <- fit_pb1 %>%
  forecast(xreg=fourier(train_pb_msts, K=c(12,12), h =24*7))

fc_pb1%>%
  autoplot() +
  ylab("Counts") + xlab("Date_Time")

checkresiduals(fc_pb1)
```

```
# Compare accuracy
# accuracy(fcast_pb, test_pb$Count)
# accuracy(fc_pb1, test_pb$Count)

# Can use cross validation by rolling a fixed window to compare the performance
# ---- fit_pb -----------
# Test
y <- msts(pb_18$Count, seasonal.periods=c(24,24*7))
h <- 168
fcmat <- matrix(0, nrow=5, ncol=h)
accuracy <- matrix(0, nrow=5, ncol=7)
colnames(accuracy) <- c("ME","RMSE","MAE","MPE","MAPE","ACF1","Theil's U")

for (i in 1:5)
{
  x <- window(y, start=c(i,1), end=c(i+11,168))
  xreg <- fourier(x, K=c(12,12))
  fit <- Arima(x, xreg = xreg, lambda = 0, seasonal = c(0,0,0), order = c(4, 1, 2))
  fcmat[i,] <- forecast(fit, xreg = fourier(x, K=c(12,12), h=h))$mean
  accuracy[i,] <- accuracy(fcmat[i,], window(y, start=c(i+12,1), end=c(i+12,168)))
}

accuracy
colSums (accuracy, na.rm = FALSE, dims = 1)/5


# ---- fit_pb1 -----------
for (i in 1:5)
{
  x <- window(y, start=c(i,1), end=c(i+11,168))
  xreg <- fourier(x, K=c(12,12))
  fit <- Arima(x, xreg = xreg, lambda = 0, seasonal = c(0,0,0), order = c(4, 1, 3))
  fcmat[i,] <- forecast(fit, xreg = fourier(x, K=c(12,12), h=h))$mean
  accuracy[i,] <- accuracy(fcmat[i,], window(y, start=c(i+12,1), end=c(i+12,168)))
}

accuracy
colSums (accuracy, na.rm = FALSE, dims = 1)/5




# 2. tbats #

tbats <- tbats(train_pb_msts)
# tbats.components(tbats)

fc_tbats <- forecast(tbats, h = 24*7)
autoplot(fc_tbats) +
  xlab('Date') + ylab('Counts')

# accuracy(fc_tbats, test_pb$Count)

# tscv
y <- msts(pb_18$Count, seasonal.periods=c(24,24*7))
h <- 168
```

```
fcmat <- matrix(0, nrow=5, ncol=h)
accuracy <- matrix(0, nrow=5, ncol=7)
colnames(accuracy) <- c("ME","RMSE","MAE","MPE","MAPE","ACF1","Theil's U")

# cross validation without re-estimation
for (i in 1:5)
{
  x <- window(y, start=c(i,1), end=c(i+11,168))
  fit <- tbats(x, model=tbats)
  fcmat[i,] <- forecast(fit, h=h)$mean
  accuracy[i,] <- accuracy(fcmat[i,], window(y, start=c(i+12,1), end=c(i+12,168)))
}

# cross validation with re-estimation
for (i in 1:5)
{
  x <- window(y, start=c(i,1), end=c(i+11,168))
  fit <- tbats(x)
  fcmat[i,] <- forecast(fit, h=h)$mean
  accuracy[i,] <- accuracy(fcmat[i,], window(y, start=c(i+12,1), end=c(i+12,168)))
}

accuracy
colSums (accuracy, na.rm = FALSE, dims = 1)/5



#================= Detecting anomalies using forecasting method =================#

# 1. Moomba Festival

# Calendar view
ts_2019 %>% filter(month(Date, label = TRUE)=="Feb" & Sensor =='Flinders Street Station Underpass') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)

ts_2019 %>% filter(month(Date, label = TRUE)=="Mar" & Sensor =='Flinders Street Station Underpass') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)

ts_2019 %>% filter(month(Date, label = TRUE)=="Jan" & Sensor =='Princes Bridge') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)

ts_2019 %>% filter(month(Date, label = TRUE)=="Feb" & Sensor =='Princes Bridge') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)

ts_2019 %>% filter(month(Date, label = TRUE)=="Mar" & Sensor =='Princes Bridge') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)


# Data
train_moomba <- ts_2019 %>%
  filter(Sensor == c('Flinders Street Station Underpass', 'Princes Bridge') &
        Date_Time>=as.POSIXlt("2019-02-04 00:00", tz="Australia/Melbourne") &
          Date_Time<=as.POSIXlt("2019-03-03 23:00", tz="Australia/Melbourne"))

test_moomba <- ts_2019 %>%
  filter(Sensor == c('Flinders Street Station Underpass', 'Princes Bridge') &
    Date_Time>=as.POSIXlt("2019-03-04 00:00", tz="Australia/Melbourne") &
```

```
     Date_Time<=as.POSIXlt("2019-03-10 23:00", tz="Australia/Melbourne"))

# Moomba <- ts_2019 %>%
#  filter(Sensor == c('Flinders Street Station Underpass', 'Princes Bridge') &
#         Date_Time>=as.POSIXlt("2019-03-08 00:00", tz="Australia/Melbourne") &
#         Date_Time<=as.POSIXlt("2019-03-11 23:00", tz="Australia/Melbourne"))


# Time Series Plots
train_moomba %>%
  mutate(
    Day = lubridate::wday(Date, label = TRUE),
    Weekend = (Day %in% c("Sun", "Sat"))
  ) %>%
  ggplot(aes(x = Time, y = Count, group = Date)) +
  geom_line(aes(col = Weekend)) +
  ggtitle("Pedestrian Counts Daily Patterns") +
  facet_grid(Sensor ~ .)


test_moomba %>%
  mutate(
    Day = lubridate::wday(Date, label = TRUE),
    Weekend = (Day %in% c("Sun", "Sat"))
  ) %>%
  ggplot(aes(x = Time, y = Count, group = Date)) +
  geom_line(aes(col = Weekend)) +
  ggtitle("Pedestrian Counts Daily Patterns") +
  facet_grid(Sensor ~ .)


# Forecasting
# DATA
train_moomba_pb <- ts_2019 %>% filter(Sensor == 'Princes Bridge' &
                                Date_Time>=as.POSIXlt("2019-02-05 00:00", tz="Australia/Melbourne") &
                                Date_Time<=as.POSIXlt("2019-03-04 23:00", tz="Australia/Melbourne"))

test_moomba_pb <- ts_2019 %>% filter(Sensor == 'Princes Bridge' &
                               Date_Time>=as.POSIXlt("2019-03-05 00:00", tz="Australia/Melbourne") &
                               Date_Time<=as.POSIXlt("2019-03-11 23:00", tz="Australia/Melbourne"))


x <- msts(train_moomba_pb$Count, seasonal.periods=c(24,24*7))
# xreg <- fourier(x, K=c(12,12))
testts_moomba_pb <- msts(test_moomba_pb[,5], seasonal.periods=c(24,24*7), start=c(5,1))

# Refit

fit_moomba <- tbats(x, model=tbats)
fc_moomba <- forecast(fit_moomba, h=168)

fc_moomba$mean
fc_moomba %>% autoplot() + ylab("Counts") + xlab("Time") + autolayer(testts_moomba_pb, series = "Test
data")

checkresiduals(fc_moomba)
```

```
# Forecasting Errors
fc_moomba_mean <- fc_moomba$mean %>% as_tsibble(index = time) %>% mutate(Date_Time =
test_moomba_pb$Date_Time)
fc_moomba_mean <- fc_moomba_mean %>% as_tsibble(index = Date_Time)

error_moomba <- test_moomba_pb$Count - fc_moomba_mean$value
error_moomba <- error_moomba %>% as_tibble() %>% mutate(Date_Time = test_moomba_pb$Date_Time)
error_moomba <- error_moomba %>% as_tsibble(index = Date_Time)

error_moomba %>% ggplot() +
  geom_line(aes(x = Date_Time, y = value)) +
  ggtitle(label = "Forecast Error of Moomba Festival")



# 2. Christmas and New Year

ts_2018 %>% filter(month(Date, label = TRUE)=="Oct" & Sensor =='Flinders Street Station Underpass') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)

ts_2018 %>% filter(month(Date, label = TRUE)=="Nov" & Sensor =='Flinders Street Station Underpass') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)

ts_2018 %>% filter(month(Date, label = TRUE)=="Dec" & Sensor =='Flinders Street Station Underpass') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)


# DATA
train_xmas_flinders <- ts_2018 %>% filter(Sensor == 'Flinders Street Station Underpass' &
                                  Date_Time>=as.POSIXlt("2018-11-26 00:00", tz="Australia/Melbourne") &
                                  Date_Time<=as.POSIXlt("2018-12-23 23:00", tz="Australia/Melbourne"))

test_xmas_flinders <- ts_2018 %>% filter(Sensor == 'Flinders Street Station Underpass' &
                                 Date_Time>=as.POSIXlt("2018-12-24 00:00", tz="Australia/Melbourne") &
                                 Date_Time<=as.POSIXlt("2018-12-30 23:00", tz="Australia/Melbourne"))

# compare with previous week of xmas
xmas_flinders <- ts_2018 %>% filter(Sensor == 'Flinders Street Station Underpass' &
                                Date_Time>=as.POSIXlt("2018-11-12 00:00", tz="Australia/Melbourne") &
                                Date_Time<=as.POSIXlt("2018-12-30 23:00", tz="Australia/Melbourne"))

drop_rows <- xmas_flinders %>% filter(Date_Time>=as.POSIXlt("2018-11-19 00:00", tz="Australia/Melbourne")
&
                                  Date_Time<=as.POSIXlt("2018-11-25 23:00", tz="Australia/Melbourne"))

xmas_flinders <- setdiff(xmas_flinders,drop_rows)

# x <- msts(train_xmas_flinders$Count, seasonal.periods=c(24,24*7))
# testts_xmas_flinders <- msts(test_xmas_flinders[,5], seasonal.periods=c(24,24*7), start=c(5,1))

x <- msts(xmas_flinders$Count, seasonal.periods=c(24,24*7))

# Refit
x_train1  <- window(x, start=c(1,1), end=c(4,168))
x_test1 <- window(x, start=c(5,1), end=c(5,168))
```

```
x_train2  <- window(x, start=c(2,1), end=c(5,168))
x_test2 <- window(x, start=c(6,1), end=c(6,168))

fit_xmas1 <- tbats(x_train1)  #model=tbats
fc_xmas1 <- forecast(fit_xmas1, h=168)
fc_xmas1$mean
fc_xmas1 %>% autoplot() + ylab("Counts") + xlab("Time") + autolayer(x_test1, series = "Test data")


fit_xmas2 <- tbats(x_train2)
fc_xmas2 <- forecast(fit_xmas2, h=168)
fc_xmas2$mean
fc_xmas2 %>% autoplot() + ylab("Counts") + xlab("Time") + autolayer(x_test2, series = "Test data")



# Forecasting Errors
Date_Time = xmas_flinders %>% filter(
            Date_Time>=as.POSIXlt("2018-12-17 00:00", tz="Australia/Melbourne") &
            Date_Time<=as.POSIXlt("2018-12-30 23:00", tz="Australia/Melbourne"))
Date_Time = Date_Time$Date_Time

error_xmas1 <- x_test1 - fc_xmas1$mean
error_xmas1 <- error_xmas1 %>% as_tibble()


error_xmas2 <- x_test2 - fc_xmas2$mean
error_xmas2 <- error_xmas2 %>% as_tibble()


error_xmas <- full_join(error_xmas1,error_xmas2)
error_xmas <- error_xmas %>% as_tibble() %>% mutate(Date_Time = Date_Time)
error_xmas <- error_xmas %>% as_tsibble(index = Date_Time)

error_xmas %>% ggplot() +
  geom_line(aes(x = Date_Time, y = x)) +
  ggtitle(label = "Forecast Error of Christmas Holiday")

save.image(file="Final_Code.RData")
load(file="Final_Code.RData")










# ======================== Use Composits Package in R ================= #

# devtools::install_github("business-science/anomalize")
install.packages("anomalize")

library(ggplot2)
library(forecast)
library(dplyr)
library(tidyr)
library(stringr)
library(rgdal)
library(tidyverse)
library(tibbletime)
library(anomalize)


# 1. Moomba Festival

# Data used in forecasting models
# train_moomba <- ts_2019 %>%
#  filter(Sensor == c('Flinders Street Station Underpass', 'Princes Bridge') &
#         Date_Time>=as.POSIXlt("2019-02-04 00:00", tz="Australia/Melbourne") &
#         Date_Time<=as.POSIXlt("2019-03-03 23:00", tz="Australia/Melbourne"))

# test_moomba <- ts_2019 %>%
#  filter(Sensor == c('Flinders Street Station Underpass', 'Princes Bridge') &
#         Date_Time>=as.POSIXlt("2019-03-04 00:00", tz="Australia/Melbourne") &
#         Date_Time<=as.POSIXlt("2019-03-10 23:00", tz="Australia/Melbourne"))

# Data for anomoly detection

Moomba_pb <- ts_2019 %>% filter(Sensor == 'Princes Bridge')
Moomba_pb <- Moomba_pb %>% filter(Date_Time>=as.POSIXlt("2019-02-04 00:00",
tz="Australia/Melbourne") &
                    Date_Time<=as.POSIXlt("2019-03-17 23:00", tz="Australia/Melbourne"))

# Princes Bridge
# ----- Anomaly detection using anomalize ------#

Moomba_pb_1 <- Moomba_pb[, c(2,5)]

# Change the time scale template
get_time_scale_template()

# time_scale_template() %>%
#  mutate(trend = ifelse(time_scale == "hour", "2 week", trend)) %>%
#  set_time_scale_template()

time_scale_template() %>%
  mutate(frequency = ifelse(time_scale == "hour", "4 week", trend)) %>%
  set_time_scale_template()


# STL decomposition method #
p1 <- Moomba_pb_1 %>%
  time_decompose(Count) %>%
  anomalize(remainder) %>%
  plot_anomaly_decomposition() +
  ggtitle("Anomaly Detection Using Anomalize")

p1

# Twitter decomposition method #
p1.1 <- Moomba_pb_1 %>%
  time_decompose(Count, method = "twitter") %>%
  anomalize(remainder) %>%
  plot_anomaly_decomposition() +
  ggtitle("Anomaly Detection Using Anomalize")

p1.1


# 2. Christmas and New Year

ts_2018 %>% filter(month(Date, label = TRUE)=="Oct" & Sensor =='Flinders Street Station Underpass') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)

ts_2018 %>% filter(month(Date, label = TRUE)=="Nov" & Sensor =='Flinders Street Station Underpass') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)

ts_2018 %>% filter(month(Date, label = TRUE)=="Dec" & Sensor =='Flinders Street Station Underpass') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)

ts_2019 %>% filter(month(Date, label = TRUE)=="Jan" & Sensor =='Flinders Street Station Underpass') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)

ts_2018 %>% filter(month(Date, label = TRUE)=="Nov" & Sensor =='Princes Bridge') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)

ts_2018 %>% filter(month(Date, label = TRUE)=="Dec" & Sensor =='Princes Bridge') %>%
  ggplot(aes(x = Time, y = Count)) + geom_line() + facet_calendar(Date)


# DATA
# train_xmas_flinders <- ts_2018 %>% filter(Sensor == 'Flinders Street Station Underpass' &
#                                  Date_Time>=as.POSIXlt("2018-11-26 00:00", tz="Australia/Melbourne") &
#                                  Date_Time<=as.POSIXlt("2018-12-23 23:00", tz="Australia/Melbourne"))

# test_xmas_flinders <- ts_2018 %>% filter(Sensor == 'Flinders Street Station Underpass' &
#                                 Date_Time>=as.POSIXlt("2018-12-24 00:00", tz="Australia/Melbourne") &
#                                 Date_Time<=as.POSIXlt("2018-12-30 23:00", tz="Australia/Melbourne"))

# compare with previous week of xmas
 xmas_flinders <- ts_2018 %>% filter(Sensor == 'Flinders Street Station Underpass' &
                          Date_Time>=as.POSIXlt("2018-11-26 00:00", tz="Australia/Melbourne") &
                          Date_Time<=as.POSIXlt("2018-12-30 23:00", tz="Australia/Melbourne"))

# drop_rows <- xmas_flinders %>% filter(Date_Time>=as.POSIXlt("2018-11-19 00:00",
tz="Australia/Melbourne") &
#                              Date_Time<=as.POSIXlt("2018-11-25 23:00", tz="Australia/Melbourne"))

#xmas_flinders <- setdiff(xmas_flinders,drop_rows)



# Anomaly detection

# ------- use anomalize ------ #
xmas_flinders <- xmas_flinders[, c(2,5)]

time_scale_template() %>%
  mutate(frequency = ifelse(time_scale == "hour", "1 week", trend)) %>%
  set_time_scale_template()

p2 <- xmas_flinders %>%
  time_decompose(Count) %>%
  anomalize(remainder) %>%
  plot_anomaly_decomposition() +
  ggtitle("Anomaly Detection Using Anomalize")

p2

# Twitter decomposition method #
p2.1 <- xmas_flinders %>%
  time_decompose(Count, method = "twitter") %>%
  anomalize(remainder) %>%
  plot_anomaly_decomposition() +
  ggtitle("Anomaly Detection Using Anomalize")

p2.1

save.image(file="Anomoly Detection.RData")
```