Analysing Melbourne CBD pedestrian counts data with a focus on imputation methods

Gavin Chin

21 September 2017

Background

The Data: City of Melbourne Pedestrian Data

The City of Melbourne provides an open data platform to access council datasets, with the intention "to increase transparency, improve public services and support new economic and social initiatives"[1]. This paper will focus primarily on the pedestrian data collected from pedestrian sensors placed throughout Melbourne's CBD. In particular, we want to be able to model the pedestrian traffic at different locations in the CBD, and forecast the traffic. In addition, we aim to have an interactive, dynamic data visualisation of the model to help us to understand how Melbourne operates.

The dataset being investigated can be obtained from the official City of Melbourne's open data[2]. The pedestrian data is in the form of hourly pedestrian counts for 43 sensor locations. In total, there is 1425192 observations in the dataset, representing 26293 hours of data. It is available in `.csv` format, allowing for easy data import into R. We can also access the pedestrian data using the the `rwalkr` package by Earo Wang[3]. This package allows R to import the data from `data.melbourne.vic.gov.au` which is updated monthly, or from the data source which is used by `pedestrian.melbourne.vic.gov.au` which is updated daily.

Not even an single sensor location has complete data for the period between between 1/1/2014 and 21/10/2017.

Imputation Literature Review Notes

Reviewing current research on imputation of missing values in traffic data, a paper on imputation of missing classified traffic data during winter season (Imputation of Missing Classified Traffic Data During Winter Season - *Hyuk-Jae Roh, Satish Sharma, Prasanta K. Sahu*) stated some methods which were found to be poor for imputation. The data they were attempting imputation for were traffic counters located on the highway network in Alberta, Canada with between 40% and 60% missing data.

[1] from *'About Melbourne Data'* page at https://data.melbourne.vic.gov.au/about

[2] the specific pedestrian data is available at https://data.melbourne.vic.gov.au/Transport-Movement/Pedestrian-volume-updated-monthly-/b2ak-trbp

[3] https://github.com/earowang/rwalkr/

Replacement with "good" historical values, or using historical average values was found to have resulted in very poor fit with high MARE/MARD (mean absolute relative error/difference). They did find that using a moving average worked the best out of all the heuristic methods used. The largest problem with heuristic methods, however, is the inherent inability to reflect sudden fluctuations/shocks during abnormal periods. With such a large proportion of missing values, pattern matching methods were investigated. This involves comparing the *study curve*, the pattern at the counter which is to be imputed, to candidate patterns (patterns at other locations).

For the purposes of that paper, it was found to be inappropriate with the data being analysed. However, unlike *Roh et al.* where patterns were compared to traffic volumes in different jurisdictions (large geographical distance), the geographic distances between the sensor locations in the Melbourne CBD pedestrian data are much smaller. Another difference is the randomness of the missing values, where the periods of missing data run longer in the Melbourne CBD pedestrian data compared to that in the Alberta traffic data. They proposed using non-parametric estimation methods with k-Nearest Neighbours for their imputation, although this is unlikely to work well with the lack of data at certain sensor locations in the Melbourne CBD Pedestrian data.

Methodology: Imputation of Missing Values

Basic GLM Approach

The first method used to impute missing values is to fit generalised linear models (GLMs) at each sensor location. Estimating a GLM with a Quasi-Poisson error distribution, where specify the model as:

$$E\left[\text{HourlyCounts}|\text{Sensor}, \text{HDay}, \text{Time}\right] = \exp(\mu_{\text{Sensor,HDay,Time}})$$

$$\text{where} \quad \mu_{\text{Sensor,HDay,Time}} \sim \text{Time} \times \text{HDay} \tag{1}$$

`Time` is the time of the day, while `HDay` is the day of the week, with an additional factor level for public holidays. Both these variables are categorical/factors. We cannot treat time of the day as an integer or numeric because it does not have a linear relationship with counts.s

We use the Quasi-Poisson error distribution as opposed to normal/Gaussian errors due to the response variables being count data. The Poisson distribution allows for only non-negative integer values, while estimating a Quasi-Poisson model estimates an additional parameter for overdispersion. This allows more flexibility in the model by allowing the assumption that $E[Y] = Var[Y] = \mu$ to be relaxed. Instead, the Quasi-Poisson distribution allows for $Var[Y] = \theta\mu$.

While this works well with a small proportion of missing values, it also requires a large number of observations. Specifically, with this paramaterisation, we have $23 + 7 + (23 \times 7) + 1 = 192$ coefficients to estimate. Another disadvantage of using this model is the lack of robustness to outliers due to the sensitivity of some parameters. This becomes particularly problematic at sensor locations which have been recently installed, where there is a lack of historical data.

Improved imputation algorithm using small-large split approach

The proposed alternative approach to imputing the missing values in the data focuses on improving the models used at locations with a large proportion of missing values. A potential threshold value which could be used to class a sensor as having a "large" proportion of missing values vs a "small" proportion of missing values is 10%. At locations with a "small" proportion of missings, we can continue to use the GLM quasi-possion regression specified in (1). At the locations with a large proportion of missing values, we need to use information from neighbouring sensors.

The first issue which needs to be addressed is the potential of values which are actually missing due to sensor failure or malfunction having a zero count. This issue will cause bias in the estimates, as well as affect the classification of a sensor. Of course, we cannot simply classify all zero counts as `NA`, as true zero values are possible.

Table 1: Summary statistics of the sequence lengths of repeated values

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
2	2	2	16.98	2	19967

Instead, a simple check we can implement is to look for long sequences of zero counts. For example, we may want to classify any sequence of `Hourly_Counts = 0` running for longer than 24 hours as being considered `NA`. From the summary of lengths of repeated values (which are repeated at least once), we see that the distribution of the lengths are very positively skewed with a mean length of 16.98 hours of missing data. Classifying any sequence of repeated values, typically 0, which has a length greater than 6 as `NA` should allow true zero counts to avoid being misclassified as `NA`. We select the length of 6 hours as it would be unreasonable to assume any sensor location to have the exact same number of pedestrian counts over such a long period.

After we have replaced questionable zero values with the value `NA`, we can calculate the proportion of missing values for each sensor location. From these calculated proportion, we can classify sensors as either having a "large" or "small" proportion of missing values, where large is defined as $\texttt{NAprop}_{sensor} > \texttt{threshold}$ and `threshold = 0.1`.

Because the models for the sensors with a large proportion of missing values rely on having neighbouring sensors having complete cases (no missing values), we want to impute these values first. This will then allow us to have as much information to be available for the models which use neighbouring sensors to to train on.

For sensors with a small proportion of missing values, we use a GLM quasipoisson model. To improve on the previous model used, we use the model:

$$E\left[\text{HourlyCounts}|\text{Sensor}, \text{Month}, \text{DayType}, \text{Time}\right] = \exp(\mu_{\text{Sensor,Month,DayType,Time}})$$

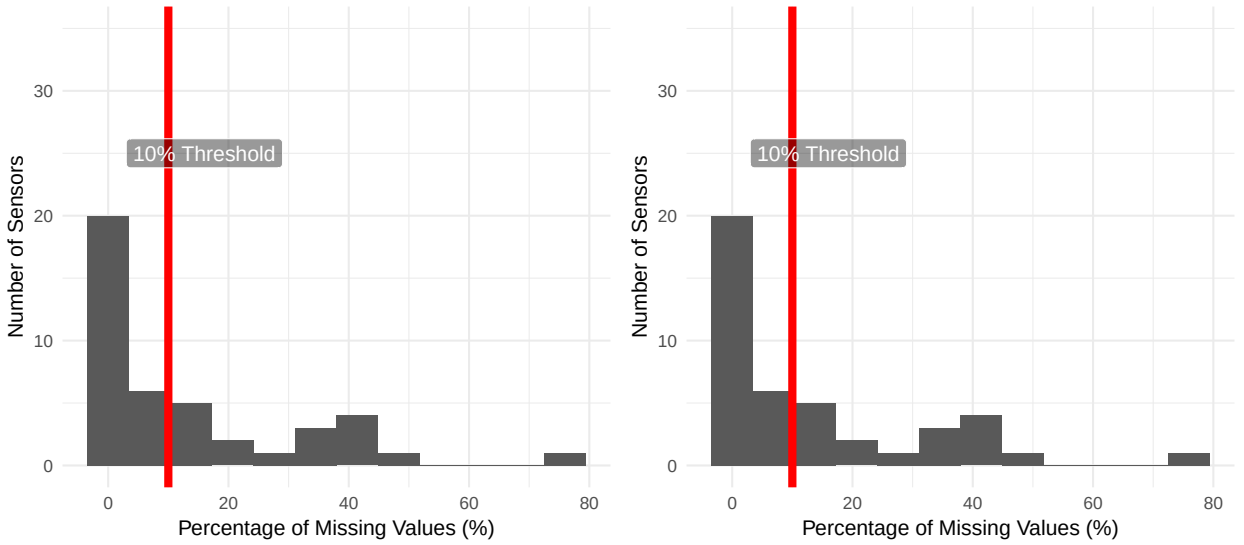


Figure 1: Histograms of the proportions of missing values at each sensor during 2014 - 2016 when calculated before and after correction for false zero counts. The distributions of proportion of missing values at each sensor is significantly different after correcting for false zero counts in the data. This emphasises the necessity to correct for false zero counts as it will affect

$$\text{where} \quad \mu_{\text{Sensor,Month,DayType,Time}} \sim \text{Month} + \text{Time} \times \text{DayType} \tag{2}$$

Instead of `HDay`, we use `DayType` which classifies Tuesday, Wednesday and Thursday as a single factor level, Midweek. We do this as the daily patterns on these weekdays are similar. This reduces the number of parameters to estimate by 46. A result of this is we also have more robust estimates for the midweek days, being less sensitive to outliers. Because this model will be estimated for sensor locations with few missing values, we have enough data to add in month as an additive effect. This will help capture the annual seasonality.

Using these estimated models, we impute the missing values to produce complete data at all these sensor locations. For the locations with high proportions of missing values can be predicted using the now complete data from all the locations which had a small proportion of missing values. Using a threshold of 10%, this gives us 25 sensor locations to use.

For each of the sensor locations with high proportions of missings, we need to decide which sensor to use for prediction. A simple method of finding a geographical closest neighbour is to take the haversine/great-circle distance between the sensor to be imputed and all the possible candidates.

Using the two geographically closest sensors with complete cases, we use another GLM specified as:

$$E\left[\text{HourlyCounts}_{\text{sensor}} | \text{Time}, \text{HourlyCounts}_{\text{neighbours}}\right] = \mu_{\text{Time,HourlyCounts}_{\text{neighbours}}}$$

$$\text{where} \quad \mu_{\text{Time,HourlyCounts}_{\text{neighbours}}} \sim \text{Time} \times \text{HourlyCounts}^{SC}_{\text{neighbour 1}} + \text{Time} \times \text{HourlyCounts}^{SC}_{\text{neighbour 2}}$$

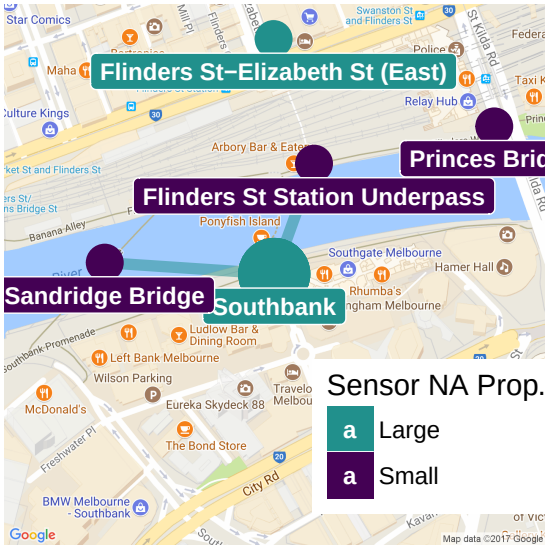
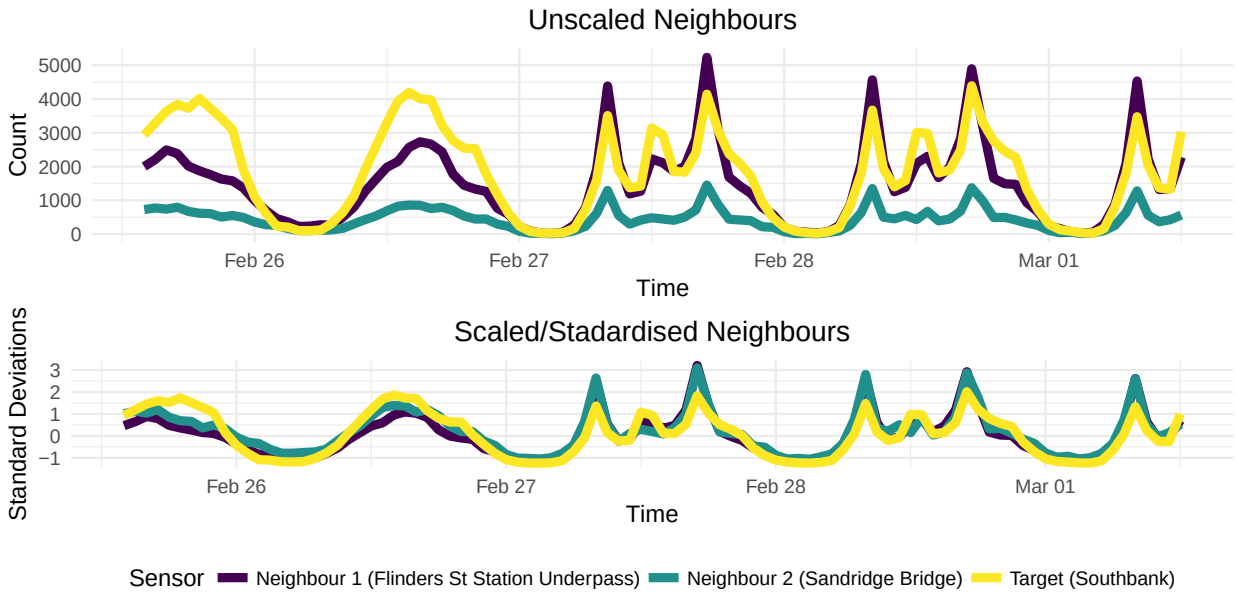


Figure 2: Map showing the algorithm selecting Flinders St Station Underpass and Sandridge Bridge as neighbours to be used to predict the pedestrian counts at Southbank

We would expect that the amount of people passing through the neighbouring sensors will be a strong predictor of the pedestrian counts as they are likely to also pass through. Using geographical neighbours, we can capture rare events effectively. Any large shocks to the counts at the neighbouring sensors will have an effect on the counts to be imputed.



In order to use a mixture of the patterns from two different sensors, we need to use the scaled counts at the neighbouring sensors to avoid the magnitude of the counts having an effect on the predicted counts. We scale the counts by standardising to $\mu = 0, \sigma^2 = 1$. Because the covariance with neighbouring sensors may vary over time, we also add an interaction term

between the neighbouring sensor counts and the time of the day.

Implementation of improved imputation algorithm

The exact code used for this paper can be found on http://github.com/gavinchin/honours2017

False zero-counts correction

The false zero-counts correction is performed in R using the run length encoding function, `rle()`. This function computes the length and values of runs of equal values in a vector. Unfortunately, this function does not recognise `NA` in the its computation, so all NA values are initially recoded to take the value 0 at each sensor. This results in all missing values and true zero counts to take the value zero when running `rle()`. With the returned run lengths, all sequences of repeated values longer than the defined length of 6 hours are replaced with `NA`. Any values which were also originally missing, but were not missing for longer than 6 hours (and hence are still coded as 0), are recoded to be `NA`. It must be noted that sequences of repeated values not equal to zero which run for longer that 6 hours would be incorrectly replaced with `NA`. This is highly unlikely, however, as 6 hours of the exact same pedestrian counts would be extremely rare.

Caculation of proportion of missing values and classification of sensors

In the working data, a wide format is used. This makes it easier to calculate missing value proportion at each sensor location, as it is the proportion of missing values in the column for each sensor. Using `is.na()` returns a logical vector indicating whether the sensor count is missing/`NA` or not. The sum of the instances where `is.na = TRUE` divided by the hours in the period of 2014-2016 is computed to get the proportion of missing values.

These proportions are saved for each sensor and then classified as either "large" or "small" based on a threshold value for the proportion. A list of sensors with large proportions of missing data and a list of sensors with small proportions of missing data is generated.

Estimation of GLM at sensor locations with a small proportion of missing values

A for loop is used to estimate the GLM specified:

```
glm(dfa[, i] ~ Month + DayType*Time, data = dfa,
    family = quasipoisson())
```

where `dfa` is the object name given to the data frame containing the training data after the false zero-count correction, and `i` is the i^{th} sensor in the list of sensors with a small proportion of missing values. These models are stored in a list, forming a list of models. An alternative approach to using a for loop in R is splitting the data into separate data frames

for each sensor location (forming a list of data frames) and applying the `glm()` function on each data frame in the list using the function `purrr::map()`. However, when this was attempted, performance issues were encountered due to the large size of the estimated model objects in R. This was not an issue when a for loop was used. Additional code for removing unnecessary elements of the GLM model object before storing to the list of models, reduucing the memory usage.

Imputation at sensor locations with a small proportion of missing values

The predicted values are obtained using `predict.glm()`, using the option `type = "response"` to have predicted values returned predicted counts. The default option returns the predicted values of $\hat{\lambda}$ in the quasipoisson process, not the counts, $\exp(\hat{\lambda})$.
Missing values are then replaced using the imputed values, generating a data frame of complete data as sensor locations for with a small proportion of missing values to be used for prediction at sensors with a large proportion of missing values.

Estimation of GLM at sensor locations with a large proportion of missing values

The following steps are performed on each sensor with a large proportion of missing values inside a for loop.

Finding neighbouring sensors

Using location data of the sensors provided by the City of Melbourne, a data frame containing the latitude and longitude coordinates of all the sensors which were classified as having a small proportion of missing values (and now have imputed, complete data), which are the candidates to be neighbours used for prediction.
Using the Great Circle/Haversine distance equation, implemented in the function `gcd.slc()`, an approximation of distance between each of the candidate sensors is calculated in and stored as a vector. Sorting by ascending order (using `sort()`) and taking the two smallest distances with `head(., 2)` returns the two geographically closest candidate neighbours.

Imputation Results

Firstly, we evaluate the initial model used to impute:

$$\text{where} \quad \mu_{\text{Sensor,HDay,Time}} \sim \text{Time} \times \text{HDay} \tag{1}$$

Note, this is also after the adjustments made to treat suspiciously long sequences of 0 values as `NA` in the actual counts. The simple GLM model is trained before this correction, while the improved model is trained on the adjusted data.

For the purposes of evaluating the imputation method at a location with a small proportion of missing values, we will first look at the counts from Southern Cross Station. At this sensor, the proportion of missing values (adjusted) is 0.02Firstly, we evaluate the goodness of fit of the predictions made within the training period (01/01/2014 to 31/12/2016).

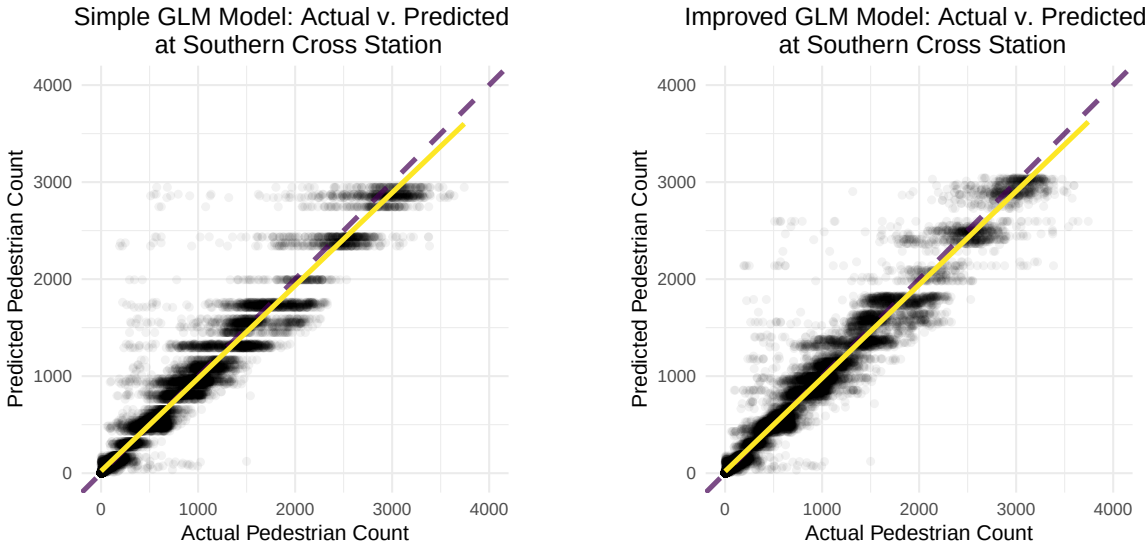


Figure 3: A plot of the in-sample Actual vs Predicted pedestrian counts at Southern Cross Station (sensor location with small proportion of missing values) for the simple GLM model and the improved GLM model. The dashed line is $x = y$, representing a reference for perfect fit, while solid line is the linear fit of the points.

Visually, we can see that the fitted values are good estimates as the relationship between the actuals vs fitted is very close to 1:1. This is seen when we perform a least squares estimate of $\widehat{Fitted}_t = \hat{\beta}_0 + \hat{\beta}_1 Actual_t$, and the estimated $\hat{\beta}_0$ is close to 0 and $\hat{\beta}_1$ is close to 1.

Here we see that the GLM model with only time and date based variables work well for fitting small periods of missing data.

However, if we try to use this same model specification at sensor locations with large proportions of missing values, the fit is poor. We use the sensor at Australia on Collins, where the proportion of missing values is 41.96%.

We can see that the simple model has bias caused by the false zero counts, resulting in the predicted values to be underestimated. This is shown by the slope of the fitted line on the actual counts against predicted counts being < 1, emphasising the importance of the first step taken in the algorithm of the improved model to class the false zero counts as missing values.

The criterion we will use to measure goodness of fit is MARE. It is calculated by:

$$MARE_{sensor} = \frac{1}{T^2} \frac{\sum_{t=1}^{T} |\widehat{\text{HourlyCounts}}_{sensor,t} - \text{HourlyCounts}_{sensor,t}|}{\sum_{t=1}^{T} \text{HourlyCounts}_{sensor,t}}$$

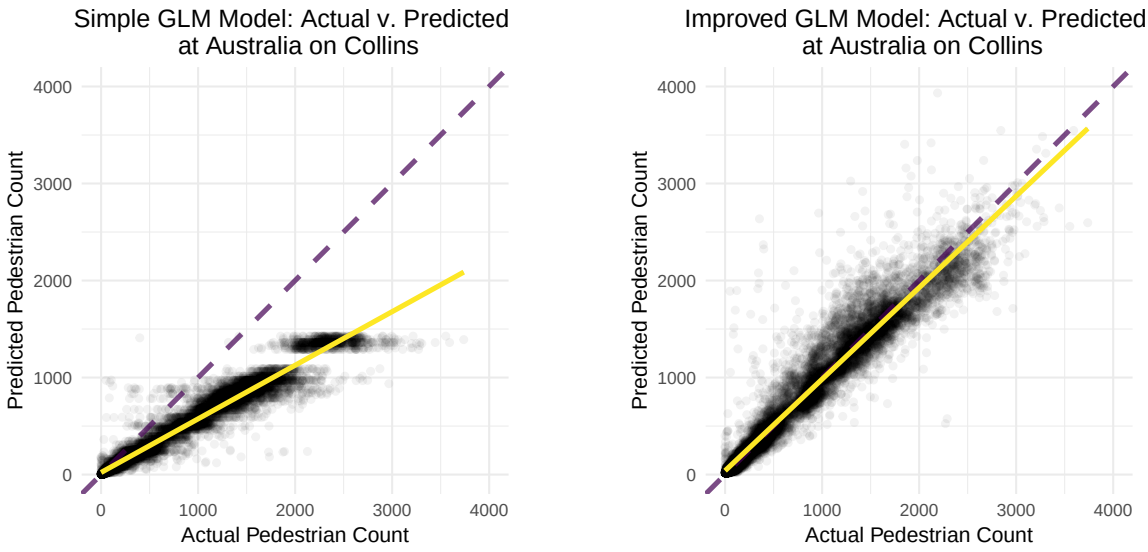


Figure 4: A plot of the in-sample Actual vs Predicted pedestrian counts at Australia on Collins (sensor location with large proportion of missing values) for the simple GLM model and the improved GLM model. The dashed line is $x = y$, representing a reference for perfect fit, while solid line is the linear fit of the points.

We calculate a seperate MARE for predictions in-sample (in the training set) and the predictions out of sample (test set of 2017 data). This will help to identify overfitting of the data. We use MARE as it will be a comparable measure between sensor locations as well as comparable between models.

While it is not very clear when comparing the distribution of in-sample MARE between imputation models, when we compare the in-sample MARE at each location we can see that the majority of locations had better in-sample fit with the improved model using the algorithm. We do see that some locations did have worse in-sample fit, but overall we see improvement with 34 having better fit with the improved model over the simple model.

Focusing on the date 04/07/2014, which has 3 hours of missing data at Southern Cross Station:

More importantly, we can evaluate the performance of the model on the out of sample observations (2017 data) as cross validation. Due to potential false zero counts in the test data sourced from `rwalkr::walk_melb()`, we need to again apply the correction by long sequences of repeated values to be able to properly evaluate the goodness of fit of out-of-sample observations. The method used for this is identical to that used on the training set in the algorithm, replacing sequences of length 6 hours or longer with `NA`.

Again, we start by comparing the fit at a sensor location with a small proportion of missing values, Melbourne Central:

We find that the out-of-sample predictions of at Melbourne Central by the simple model has a tendency to underestimate the pedestrian counts. By comparison, the predictions made by

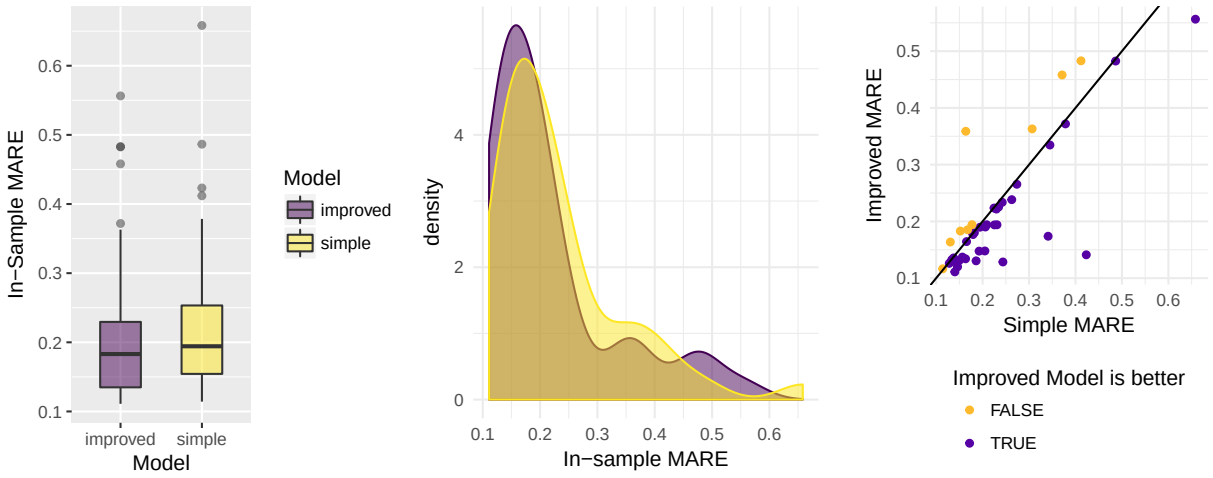


Figure 5: Distribution of in-sample (2014-2016 data) MARE by imputation model

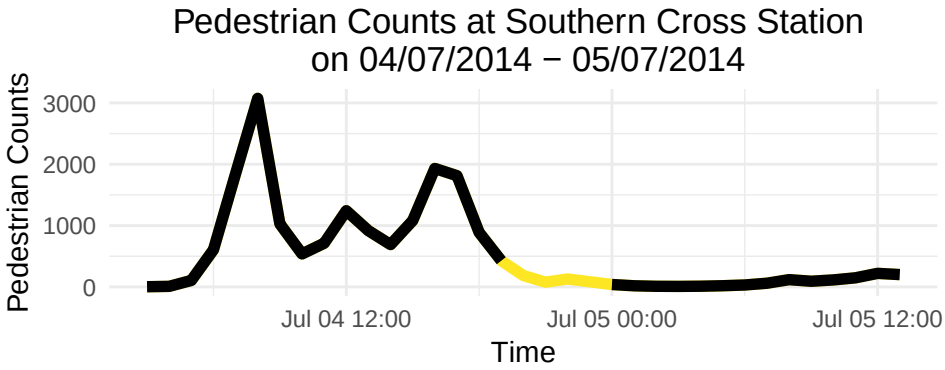


Figure 6: A plot of the pedestrian counts at Southern Cross Station. The black line is the observed counts, where the yellow line is the imputed counts.

the improved imputation model does not underestimate the counts as much. Comparing the out-of-sample MARE between models at Melbourne Central, MARE decreases from 21.43% to 18.55% with the use of the improved model.

At a sensor location with a large proportion of missing values, Collins Place (South), we again see a tendency to underestimate counts with the simple imputation model. The improved imputation method reduces the underestimation of the predictions and provides better predictions. The MARE is greatly improved, where it decreases from 21.39% to 14.65% with the use of the improved model.

Spencer St-Collins St (North)/(South) and Flinders St-Swanston St (West) was omitted from the the plots above as the out-of-sample MARE is extremely high at 2584.12%, 3278.74% and 3042.22% respectively.

As these sensor locations were both classified as locations with a large proportion of missing values, it would appear that the relationship between these locations and their neighbouring sensors has changed. This is because the in-saple MARE at these sensor locations were quite

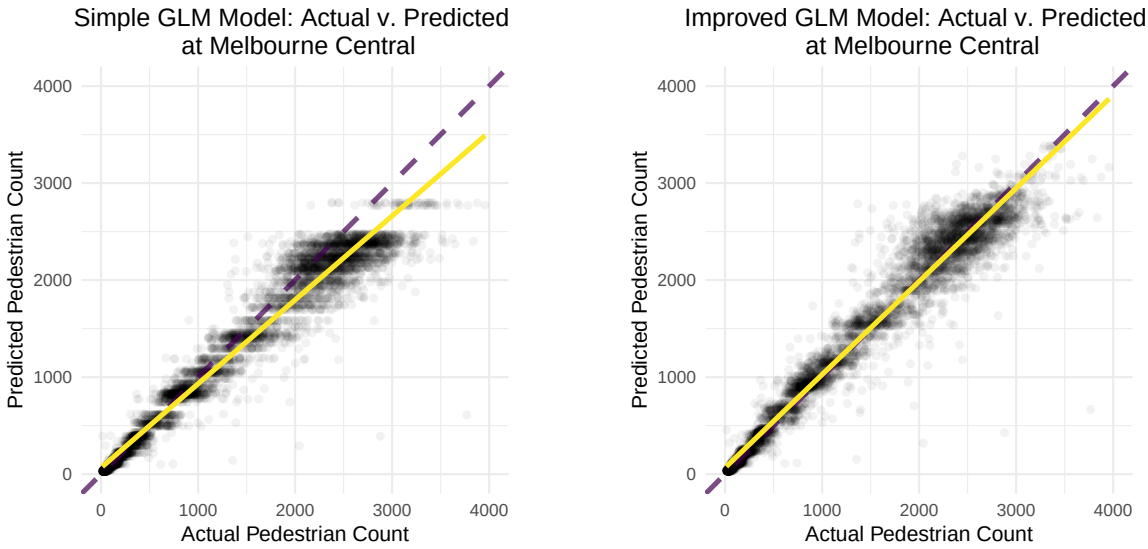


Figure 7: A plot of the out-of-sample (2017) Actual vs Predicted pedestrian counts at Melbourne Central (sensor location with small proportion of missing values) for the simple GLM model and the improved GLM model. The dashed line is $x = y$, representing a reference for perfect fit, while solid line is the linear fit of the points.

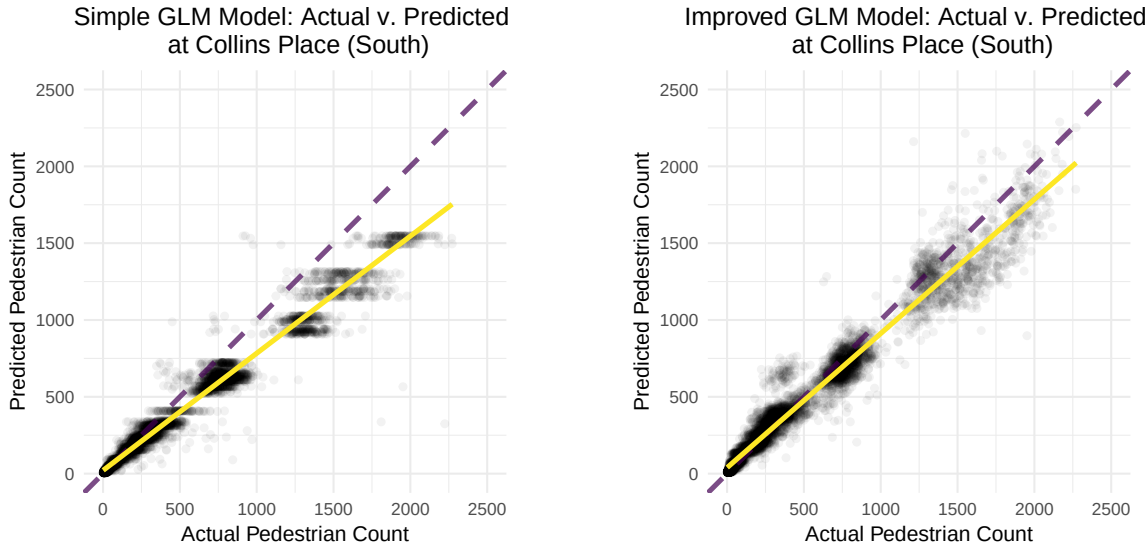


Figure 8: A plot of the out-of-sample (2017) Actual vs Predicted pedestrian counts at Collins Place (South) (sensor location with large proportion of missing values) for the simple GLM model and the improved GLM model. The dashed line is $x = y$, representing a reference for perfect fit, while solid line is the linear fit of the points.

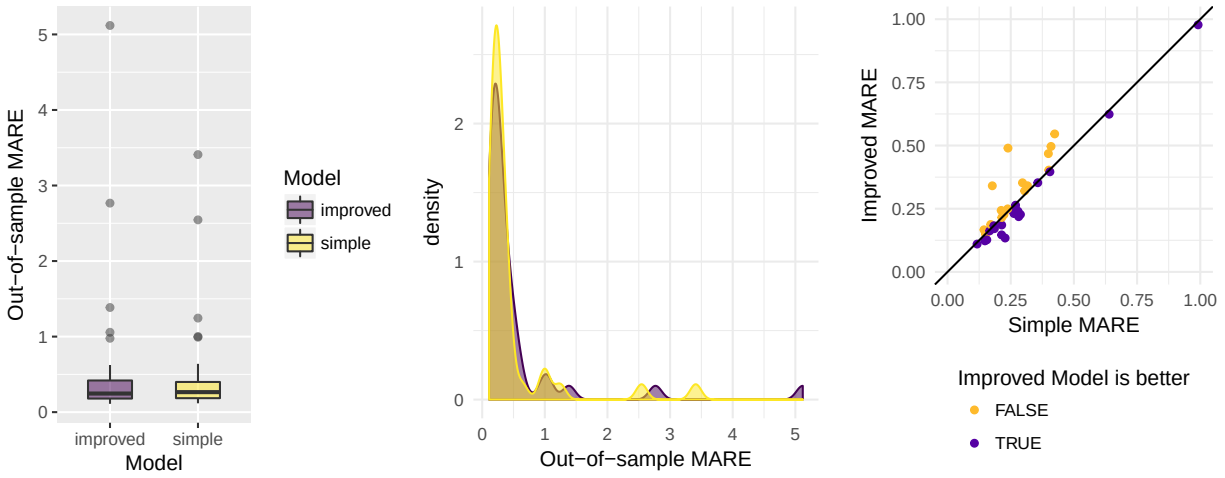


Figure 9: Distribution of out-of-sample (2017 data) MARE by imputation model, excluding Spencer St-Collins St (North)/(South) and Flinders St-Swanston St (West) sensors

acceptable at 13.05%, 19.41% and 11.65% respectively.

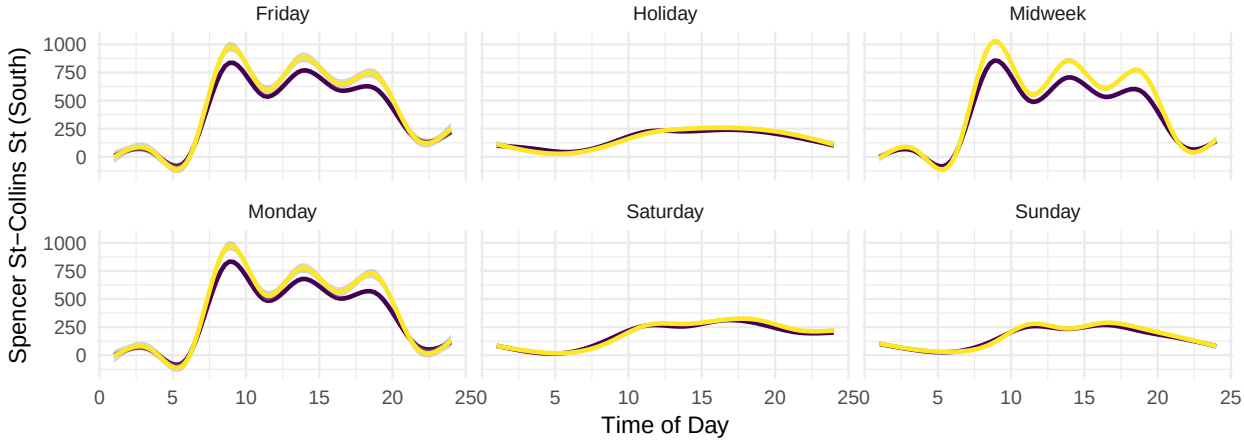


Figure 10: Average pedestrian counts by time of day and type of day (DayType) at Spencer St-Collins St (South) for training data (dark line) and test data (light line)

The plot of the average counts by time of day and type of day and data set shows that this location saw an increase in counts on weekdays which did not occur at the neghbouring sensors. We also see that the pattern is slightly different during weekdays, with a higher afternoon/evening peak. These types of change in pedestrian traffic behaviour is hard to capture in our imputation model, especially when its neighbours did not have a similar change. While we can add a year variable to allow for year to year growth, the short period of training data doesn't make this appropriate at this stage.

Another way we can perform cross-validation is to remove data at a sensor, and evaulate the predictions made. This will allow us to evaluate the imputation model's performance within sample, which is the primary concern with them imputation process.

We select a sensor which has a small proportion of missing values, Bourke Street Mall (South), which has 99.981% of data available. As observed in the data, missing data is present in different ways: random, short periods of missing data (no more that 6 hours) or long periods of missing data (multiple months). We will simulate both types of missingness at this sensor location.

First, data is randomly removed such that 20% of the training data is missing, then the imputation model is estimated. Using the estimated model, the MARE within the training data is calculated (in-sample MARE).

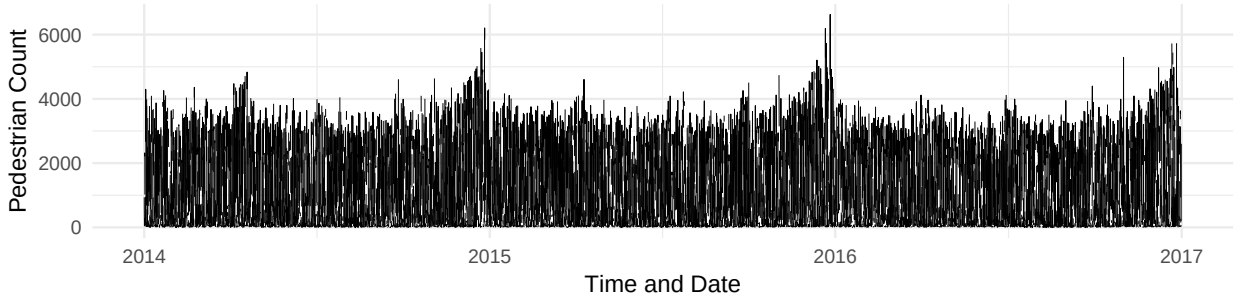


Figure 11: Plot of simulated missingness method 1 at Bourke Street Mall (South). 20% of the observations are removed from the traning data for the imputation model at random.

With random hours of missing data, the predictions made by using neighbouring sensors are estimated with good accuracy with a low MARE of 11.04%. Unfortunately, this type of missingness is not consistent with what is generally observed at most sensors with large proportions of missing data.

Instead, a more meaningful method is to remove 20% of the data in a single period to simulate what is typically observed. A random date is taken and then 2629 hours of data is removed before and after this date (10% before, 10% after). The imputation model is estimated again in order to calculate a MARE.

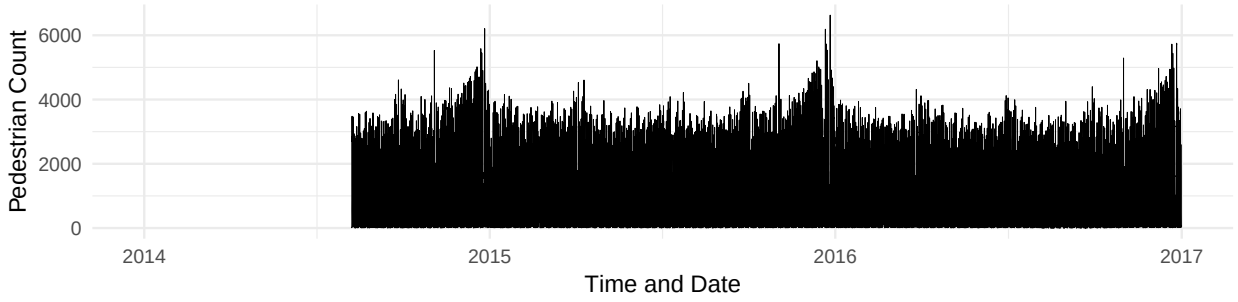


Figure 12: Plot of simulated missingness method 2 at Bourke Street Mall (South) with 20% of the training data removed as a sequence. This method of simulating missing data is a better representation of the missing data observed at sensors with a large proportion of missing data compared to randomly removing hourly observations from the training data.

The MARE of the estimated model is 11.06%, which is similar to what was obtained when the missing values were random. This demonstrates that the model quite well when we remove 20% of the observations from the training data.

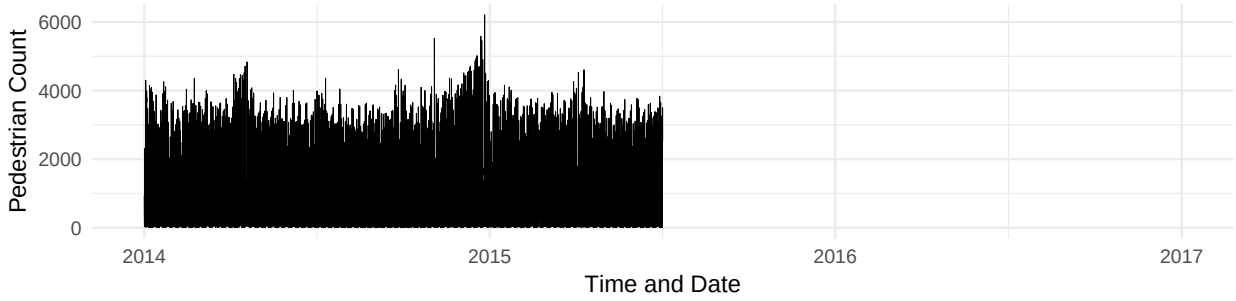


Figure 13: Plot of simulated missingness method 2 at Bourke Street Mall (South) with 50% of the training data removed as a sequence. A large proportion of the training data is removed to test the robustness of the imputation method used at sensors with extremely large proportions of missing data during the training period defined.

Removing 50% of the data using the same method, it is found that the in-sample MARE is not affected significantly by the larger proportion of missing training data where it is only 11.26%.

Predictive Model

For the purpose of prediction, the use of the most simplistic model possible to make quick predictions of the expected pedestrian counts at each location, accessible to the public. The model is not expected to be able to predict large deviations from time and date based estimates.

Similar to the simple imputation model used, a generalised linear model with quasi-poisson errors is estimated using time and date based variables as predictors. The model specification used is:

$$E\left[\text{HourlyCounts}|\text{Sensor}, \text{Month}, \text{DayType}, \text{Time}\right] = \exp(\mu_{\text{Sensor,Month,DayType,Time}})$$

$$\text{where} \quad \mu_{\text{Sensor,Month,DayType,Time}} \sim \text{Month} + \text{Time} \times \text{DayType} \tag{2}$$

Using a GLM predicts well compared to estimating with other models such as AR(p) or other time series models when considering the tradeoff between model complexity and prediction accuracy. Due to multiple seasonalities, AR(p) models would need to be considered on a hour of the week basis at each sensor location. Prediction with time series models would also require of historical data. For the objective of this predictive model, the additional imputs

required by end users to predict pedestrian counts when making out of sample predictions is not desirable.

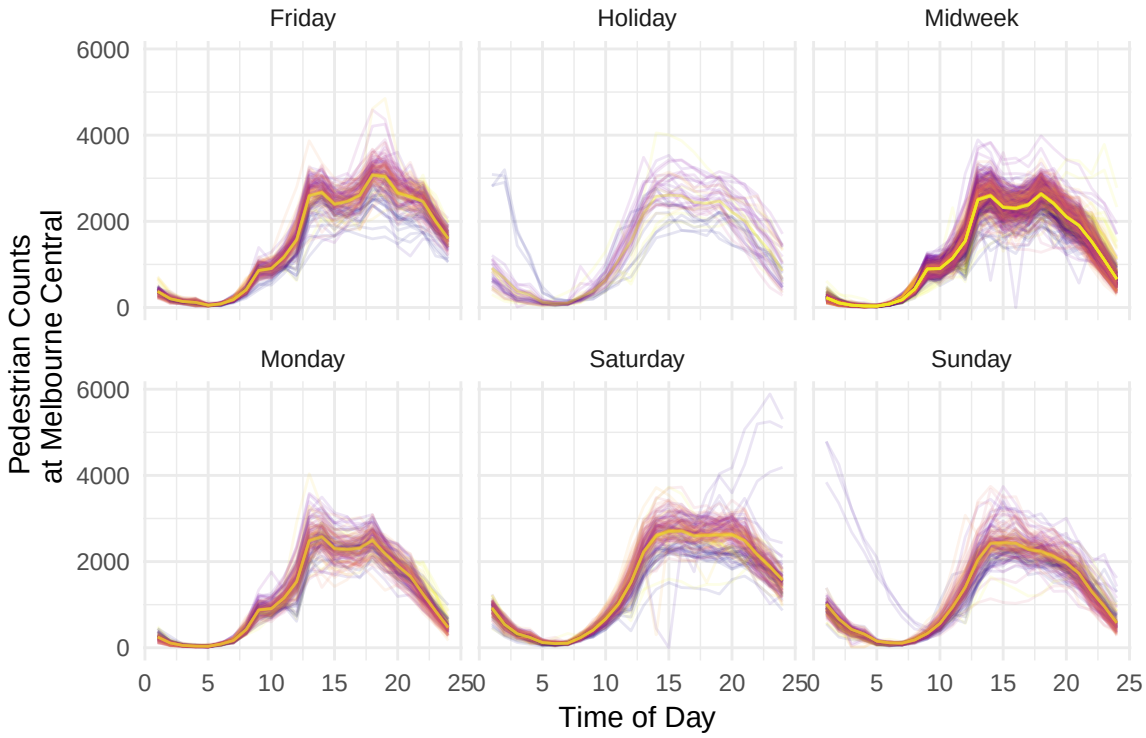


Figure 14: Plot of daily pedestrian counts series at Melbourne Central by DayType and Month. Month of the year is mapped to colour to help identify annual seasonal patterns, where month has an additive effect on the pedestrian counts. At Melbourne Central, different months can be seen to have different mean counts for a given type of day and hour of the day.

A plot of the daily pedestrian counts series at Melbourne Central by type of day and month show that time and date variables as deterministic predictors adequately explains most variation in the pedestrian counts. It can be seen that month has an additive effect as the pattern of the hourly counts for a given date are not affected by month. The six different factor levels that `DayType`, the type of day, is seen to be a sufficiently explain six different patterns. The type of day, Midweek, shows evidence that no additional information is gained from identifying the day to be Tuesday, Wednesday or Thursday.

When estimating the GLM at each sensor location with R, only the necessary elements in the model object which are required in order to retain the functionality of R's `predict.glm()` function are saved to file. Saving the entire GLM object as given by the base R `glm()` function is highly inefficient when working with large datasets. This is due to the model object providing redundancies. In particular, elements which are unnecessarily saved for the purposes of the predictive model: the training data, the predictors in a model matrix format (the matrix of $\mathbf{X}$ after being formatted to be used for regression, such as recoding categorical variables into dummy variables), the response variable as another vector ($\mathbf{Y}$), the fitted values of the model after estimation, the residuals of the estimates, as well as the estimated effects.

The result of removing all the unnecessary redundant data elements in the GLM object is a storage size reduction from 29.9 Mb to 510 Kb. This is only 1.66% of the size of the original GLM object. With 43 sensors (and models), this represents a reduction in model object size from 1.3 Gb to 21.4 Mb.

The estimated model parameters are saved so that they can be used for on-the-fly predictions by end users without needing to estimate the model themselves. Using the estimated model parameters, a new function is written to facilitate the predictions in an easy to use format.

```
ped_predict(pred_date = "2017-12-26", t_hour = 13, is_pub_hol = TRUE)
```

This function will return tibble (trimmed down version of `data.frame()` in R) containing the predicted counts at each sensor location for the 13th hour (13:00/1:00 PM) on Boxing Day (26 December) 2017. This tibble can then be easily used for visualisations or analysis. Due to the lack of data on future public holiday dates, the function assumes the date given is not a public holiday unless `is_pub_hol = TRUE` is provided.
Another function which uses the `ped_predict()`, `ped_predict_day()`, returns a tibble containing the predicted counts at each sensor location for all 24 hours of the date given to predict.

Using these function as a basis, visualisations of pedestrian counts can be easily generated in R:

```
ped_predict_day("2018-05-21") %>%
  ggplot() + geom_line(aes(x = Date_Time, y = Southbank))
```

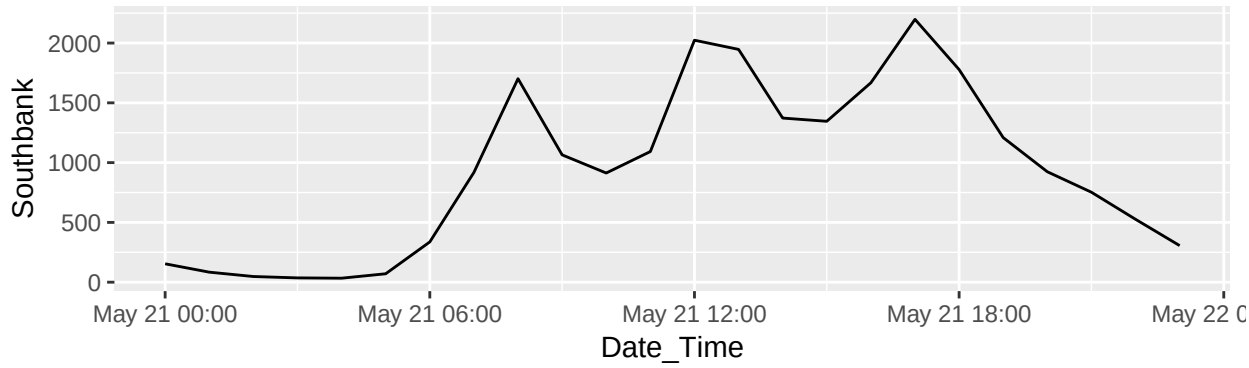


Figure 15: Prediction for 21/05/2018 in R using `ped_predict_day()`, then plotting the time series using `ggplot2`. This shows the user friendly nature of the function, where the function returns predictions in a tidy, wide format. No data manipulation is required in order to plot the predicted counts.

Conclusion

Analysis of the Melbourne CBD pedestrian data made publicly available by the City of Melbourne has revealed issues with the dataset with relation to missing values in particular. A major issue which was found when using data sourced from the City of Melbourne's official pedestrian data visualisation (also accessible using `rwalkr::walk_melb()`) is potential for false zero-counts where missing values were given the value of zero. As outliers have major implications on analysis, a correction for the false counts in the data was proposed, by checking for lengths of repeated value sequences.

An imputation method for the pedestrian data was developed with a theory of sensor locations with a large proportion of missing values require modelling with non-deterministic predictors due to the lack of data. This method involved using neighbouring sensor counts data for prediction. Sensors with a small proportion of missing values used imputation models with only time and date variables as predictors. The proposed algorithm was evaluated in comparison to a single model specification of a generalised linear model with only time and date based variables at each sensor location and uncorrected training data (false zero counts not corrected). This method was found to have improved prediction accuracy, as well as being robust at high levels of missingness.