Modelling the City of Melbourne Pedestrian Data

Thesis Presentation

Gavin Chin, supervised by Di Cook

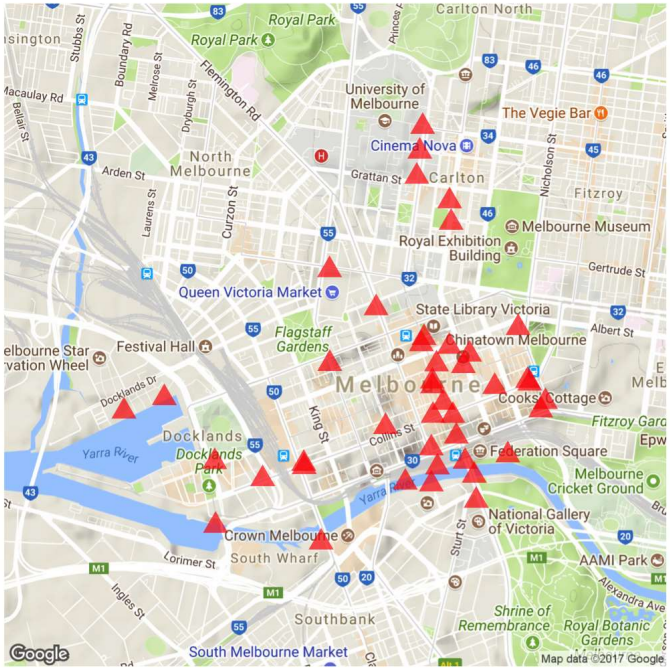
3 October 2017

Introduction

The City of Melbourne has an Open Data Platform which makes council data available to the public

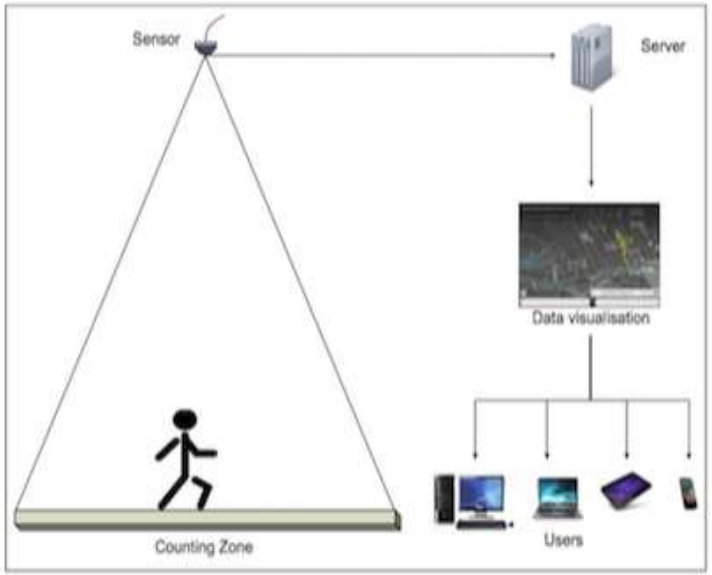
This project will focus on the pedestrian sensor data

Melbourne Pedestrian Sensor Locations



How is the data collected?

The City of Melbourne has 43 sensors installed around Melbourne's CBD. These sensors are positioned under an awning or street pole, and count the number of people passing each hour.



source: *http://www.pedestrian.melbourne.vic.gov.au/*

Accessing the data

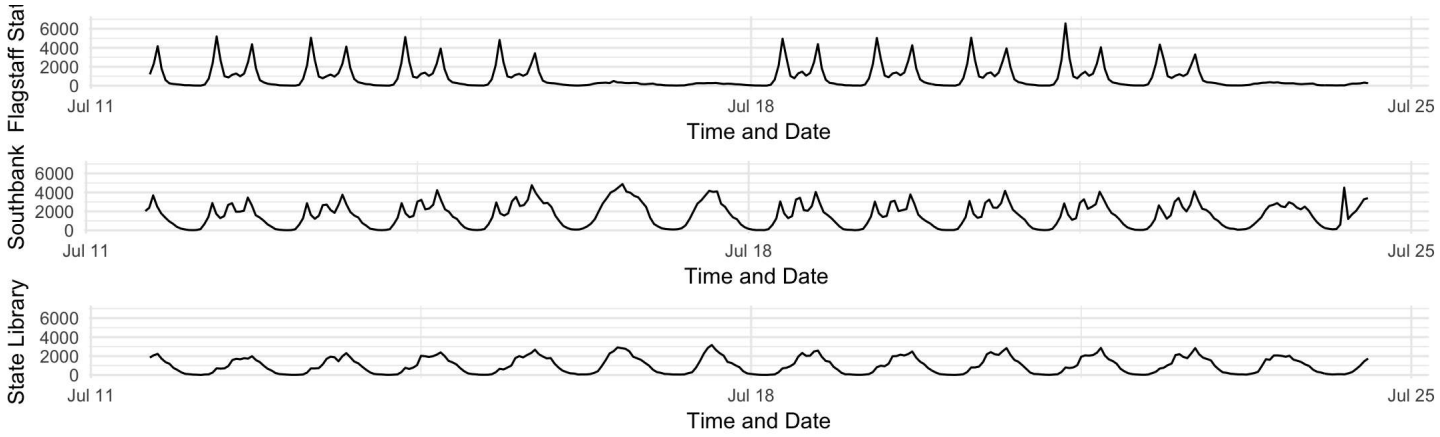
- The data is available as a `.csv` at https://data.melbourne.vic.gov.au/, and is updated monthly
- Alternatively, the data is also available as daily data from http://www.pedestrian.melbourne.vic.gov.au/
- The `rwalkr` R package by Earo Wang provides an API to access the data from both sources easily in R in a tidy format
- Both sources of data provide the same counts data, but have different forms

We use the daily data sourced from `pedestrian.melbourne.vic.gov.au` because missing values are explicit, accessible using `rwalkr::walk_melb()`

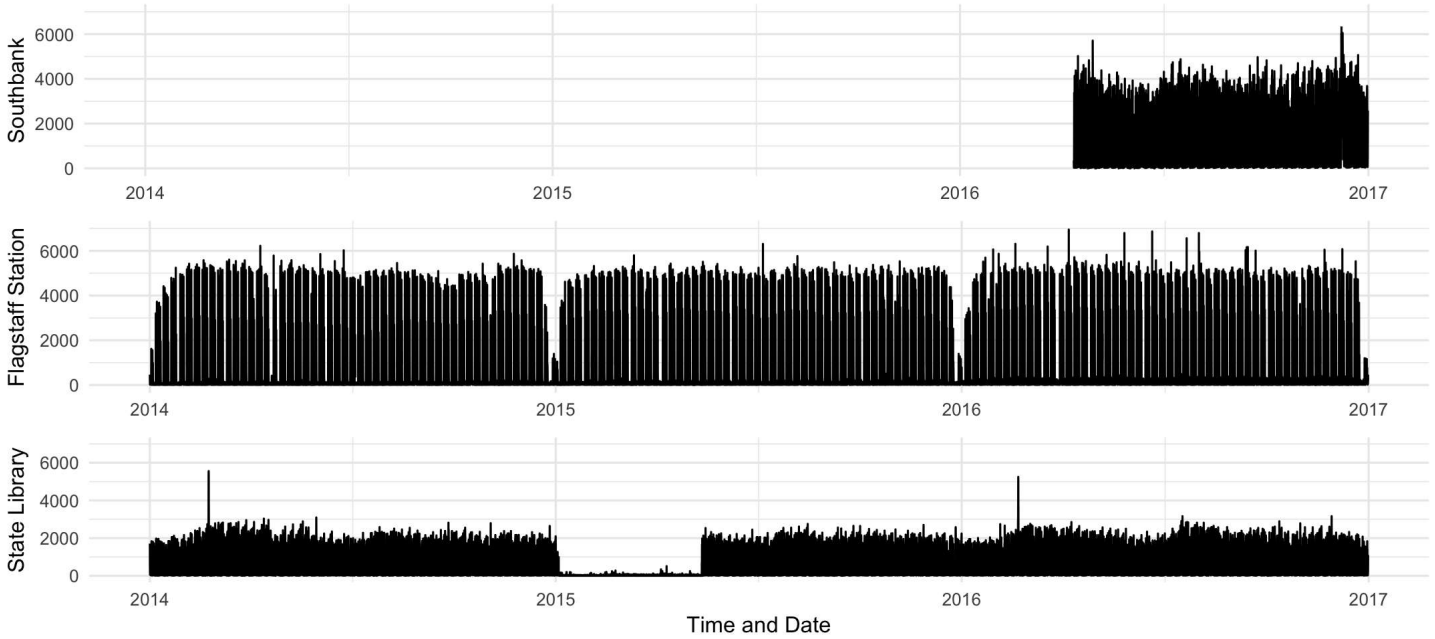
Format of the data

Time and date, sensor ID/location and hourly pedestrian count (volume)

```
## # A tibble: 1,409,712 x 5
##                   Sensor  Date_Time       Date  Time Count
##                    <chr>     <date>     <date> <int> <int>
## 1         State Library 2014-01-01 2014-01-01     0  1709
## 2 Collins Place (South) 2014-01-01 2014-01-01     0   893
## 3 Collins Place (North) 2014-01-01 2014-01-01     0   404
## 4     Flagstaff Station 2014-01-01 2014-01-01     0   462
## # ... with 1.41e+06 more rows
```



The Problem: Prediction of pedestrian traffic



- Different locations have different counts and patterns
- We want to build the prediction model using 2014-2016 data to train it

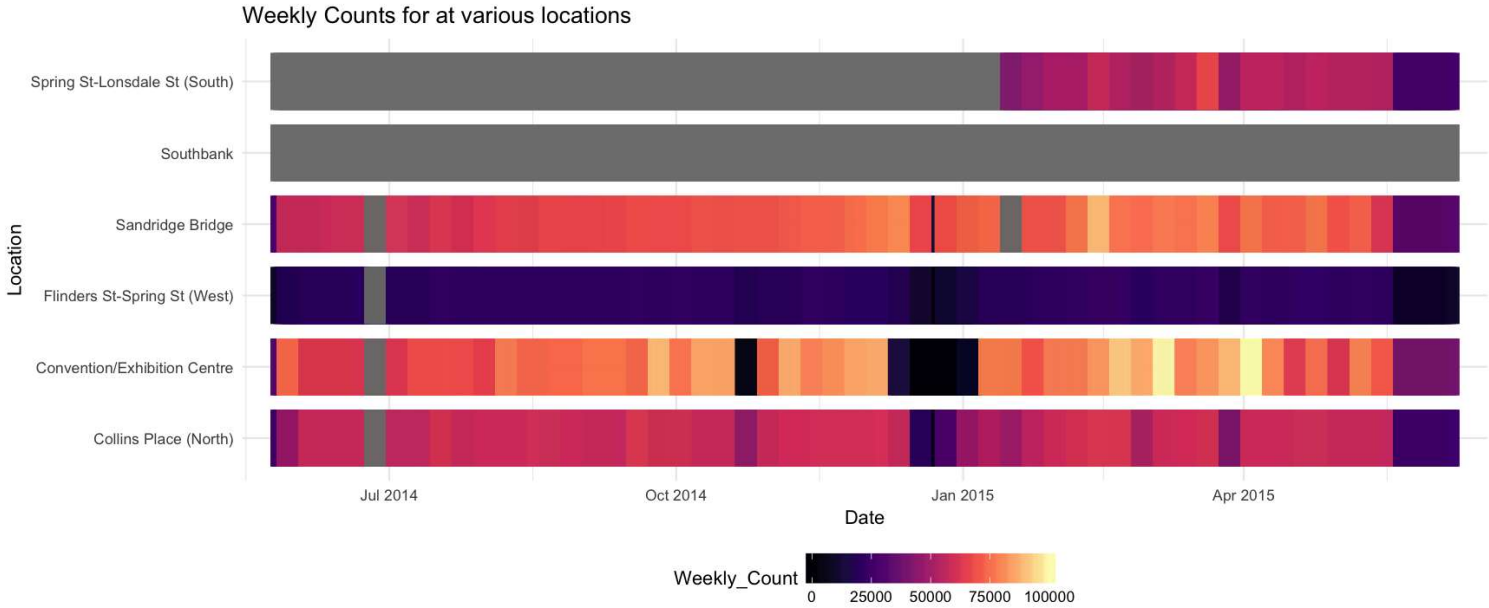
Why do we want to model pedestrian traffic?

Being able to predict how many people pass through a certain location can be used by the government sector as well as the private sector

Examples ways it can be used:

- Government example uses:
  - Infrastructure planning
  - Security planning
- Private example uses:
  - Marketing campaign planning
  - Resource management
  - Investment planning

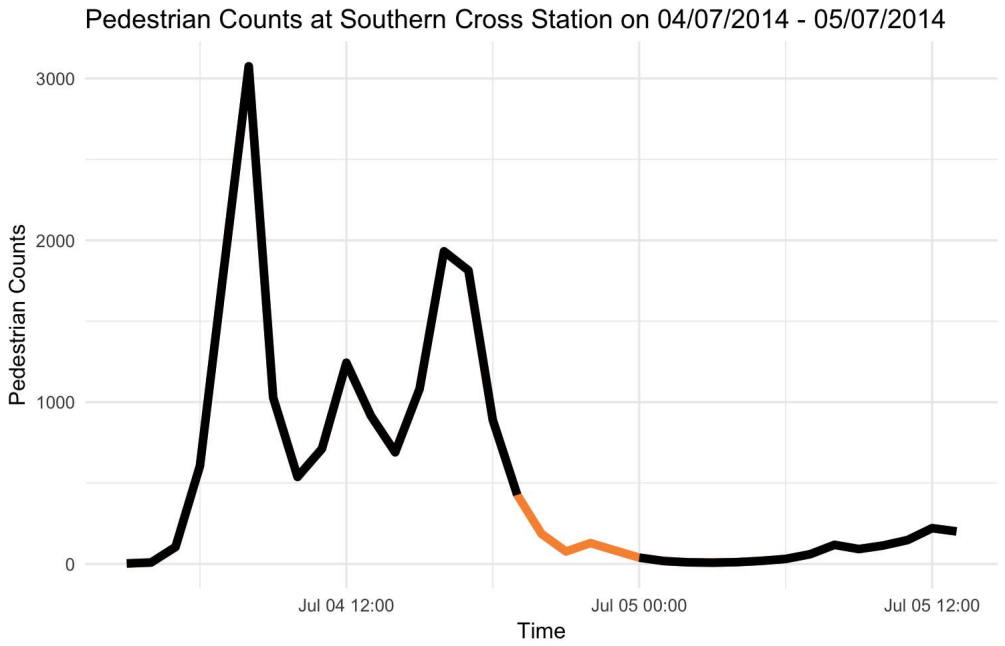
Missing Data



- Some small and large periods of missing data (grey indicates missing)
- We need to impute these values before building a predictive model

Imputation of missing values

- A simple approach of using a single model specification at all locations did not work well
  - Some sensors had large proportions of missing data, particularly pre-2015
- Need to use different model for sensors with large proportion of missing values and those with a small proportion of missing values

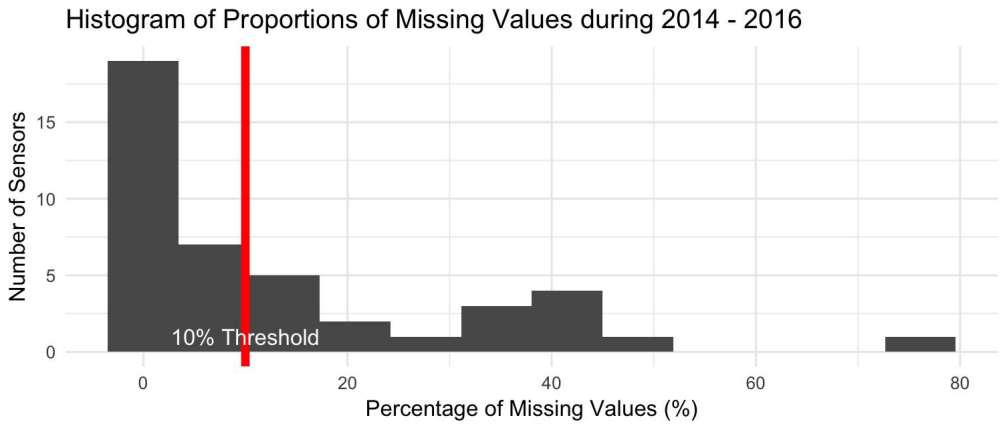


Imputation Algorithm

- **Step 1:** Check proportion of missing values
- **Step 2:** At each sensor location, run appropriate GLM model with quasipoisson error distribution for imputation
  - if "small", use GLM with `Hourly_Counts` as response variable and time and date based variables as predictors
  - if "large", use GLM with `Hourly_Counts` as response variable and neighbouring sensor counts as predictors
- **Step 3:** Replace missing values with imputed values

Step 1: Calculating proportion of missing values

- Need to first fix potential false zero-count values
  - Some observations of `Count = 0` may actually be missing values
  - Include a check for long periods with zero count, and replace with `NA` if too long
- Classify sensors as "small" or "large" proportion of missing values with a threshold proportion
  - We used 10% as the threshold proportion



Step 1: Classification of sensors by "small" and "large" proportion of missing values

Step 2a: Fit models at sensors with "small" proportion of missing values

At each sensor location with small proportion of missing values, we estimate a generalised linear model with:

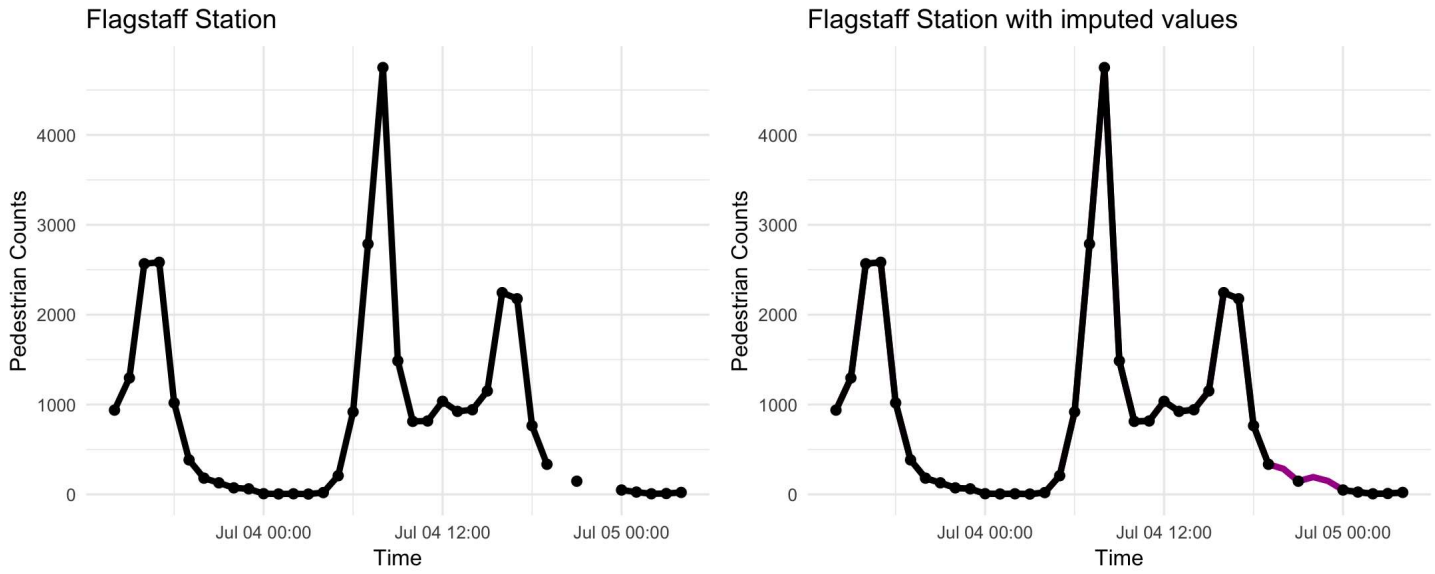
$$\mu_{\text{Sensor,Month,DayType,Time}} \sim \text{Month} + \text{Time} \times \text{DayType}$$

and a quasipoisson error distribution.

- Quasipoisson regression allows for $Var[\text{HourlyCount}] = \theta\mu$ (ie. overdispersion)
- These models only use time and date based variables to predict pedestrian counts
  - `DayType` is a factor level/categorical variable based on the day of the week and public holidays
  - `Time` is treated as a factor level, as it has a non-linear relationship with counts

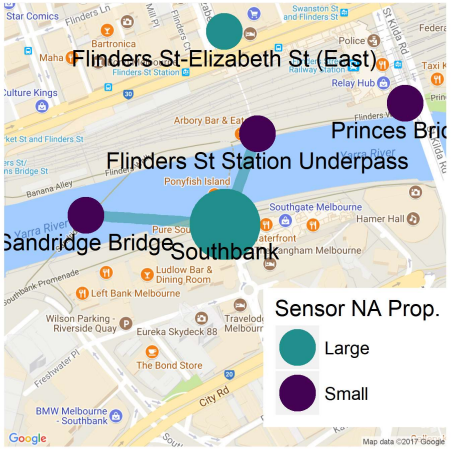
Step 2b: Replacing missing values with imputed values

Need to fill the missing data at the sensors with small proportions of missing values first



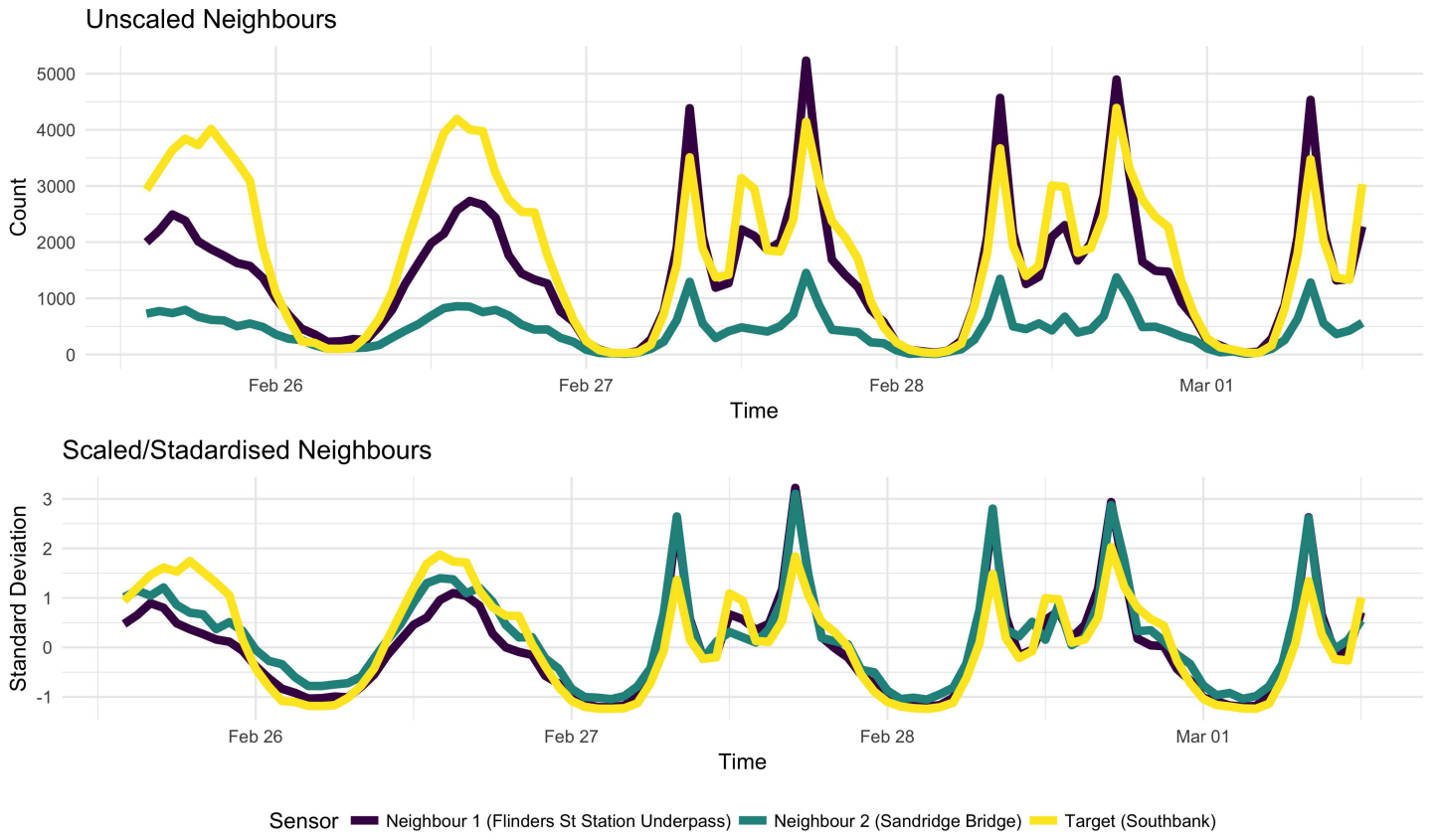
Step 2c: Find neighbours for sensors with "large" proportion of missing values

- The imputation models at sensors with large proportions of missing values are based on neighbouring sensors
- Requires complete data at neighbours to work properly



- Need to use data from neighbouring sensors, defined as geographical neighbours for simplicity
- Find two closest sensors by great-circle (haversine) distance

Step 2c: Fit find neighbours sensors with "large" proportion of missing values



Need to standardise counts at neighbours to match patterns

Step 2c: Fit models using neighbours

At each sensor location with "large" proportion of missing values, we estimate a generalised linear model with:

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

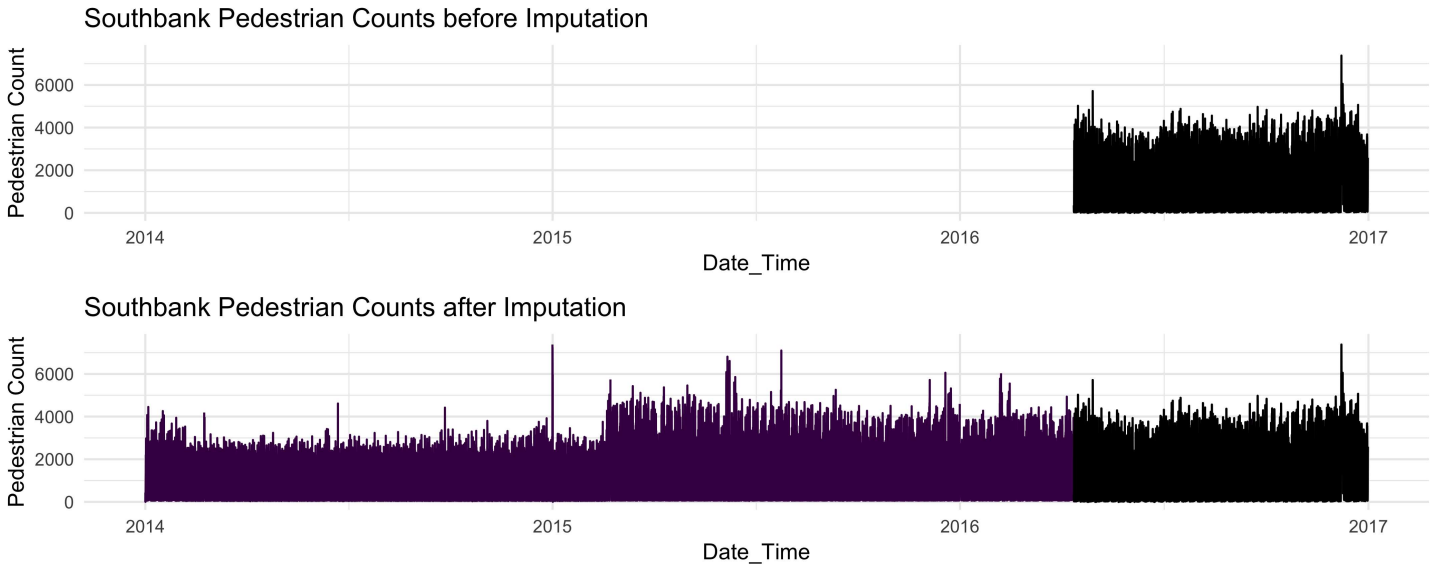
and a quasipoisson error distribution.

- use interaction with `Time`, the hour of the day, and the scaled counts at neighbours
  - Again, `Time` is treated as a factor variable
  - Counts scaled by standardising each sensor to $\mu = 0$ and $\sigma^2 = 1$

Step 3: Replacing missing values with imputed values

Using the models estimated, we can impute the missing values

- For sensors with large proportion of missing values, we needed to have complete data at the neighbours
  - Can't use neighbours with large proportion of missing values
  - Use imputed data from sensors with small proportion of missing values

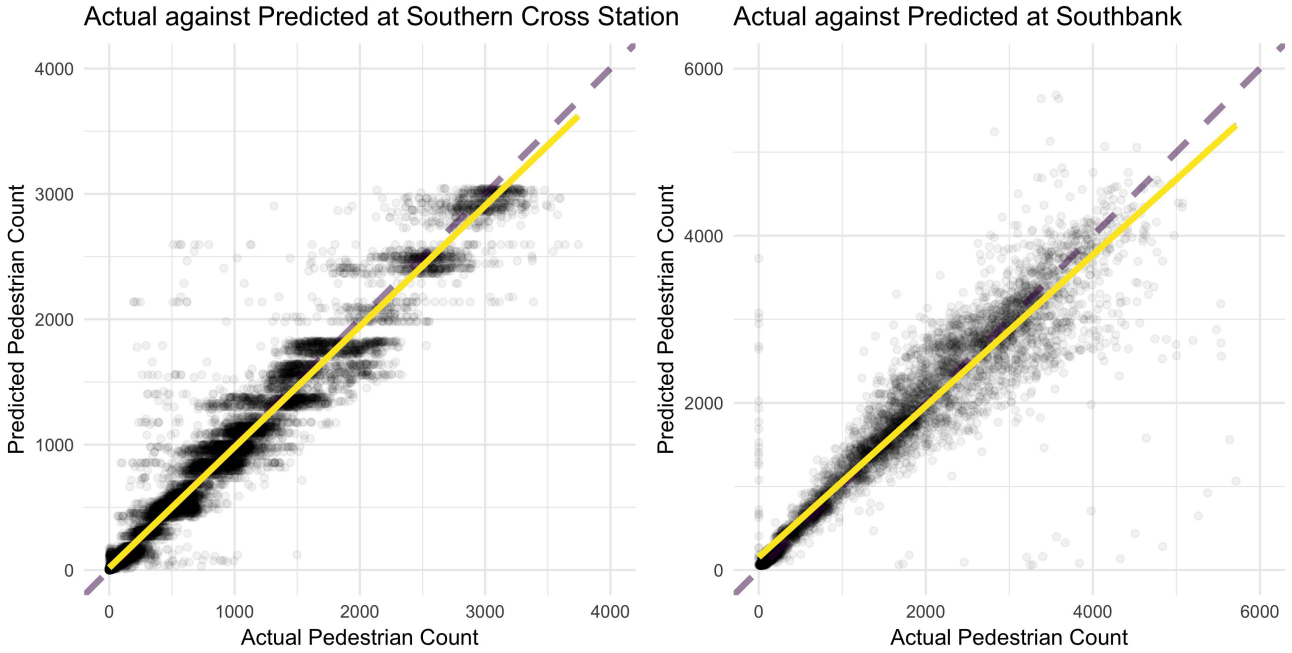


Implementation of the imputation algorithm

- general code which will run the algorithm
- inputs required:
  - data in wide format
  - threshold value for missing values proportion (*default:* `threshold = 0.1`)
  - max length of repeated values (*default:* `maxlength = 17`)
- use R and RStudio with packages:
  - `dplyr`, `lubridate` and `tidyr` for data manipulation
  - `foreach` and `doSNOW` for parallel computing

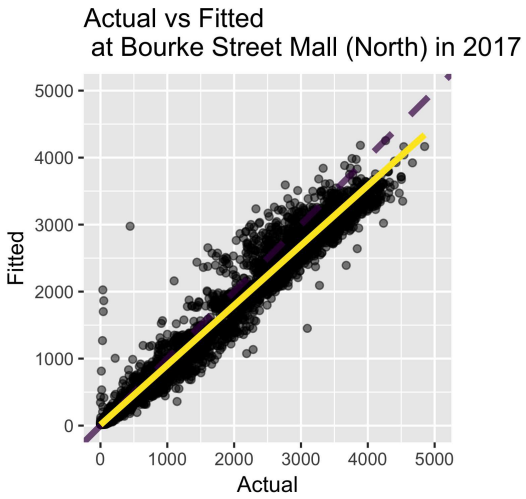
Imputation Model Evaluation

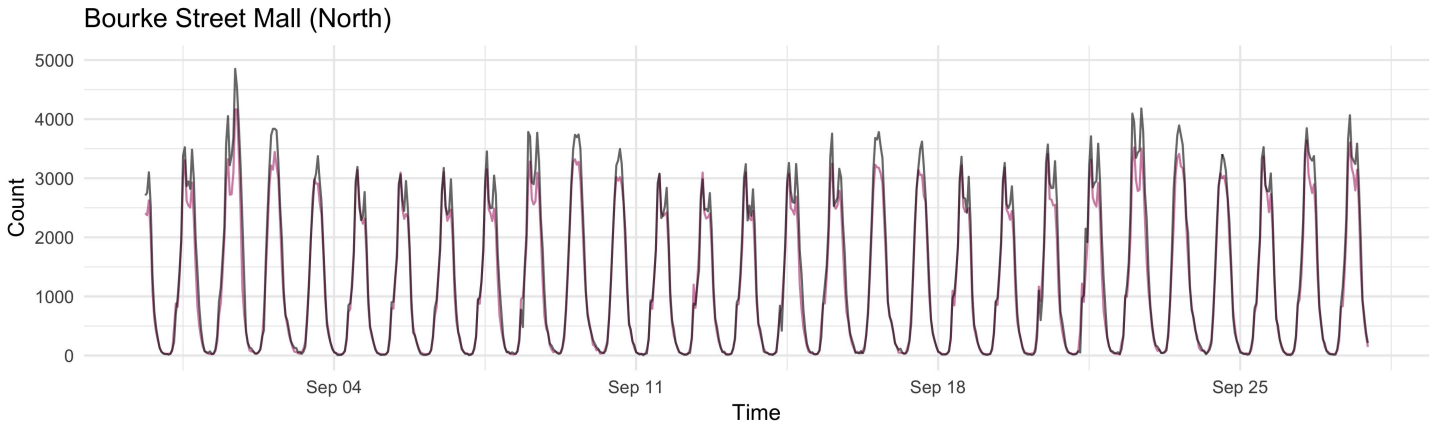
In-sample fit



Imputation Model Evaluation

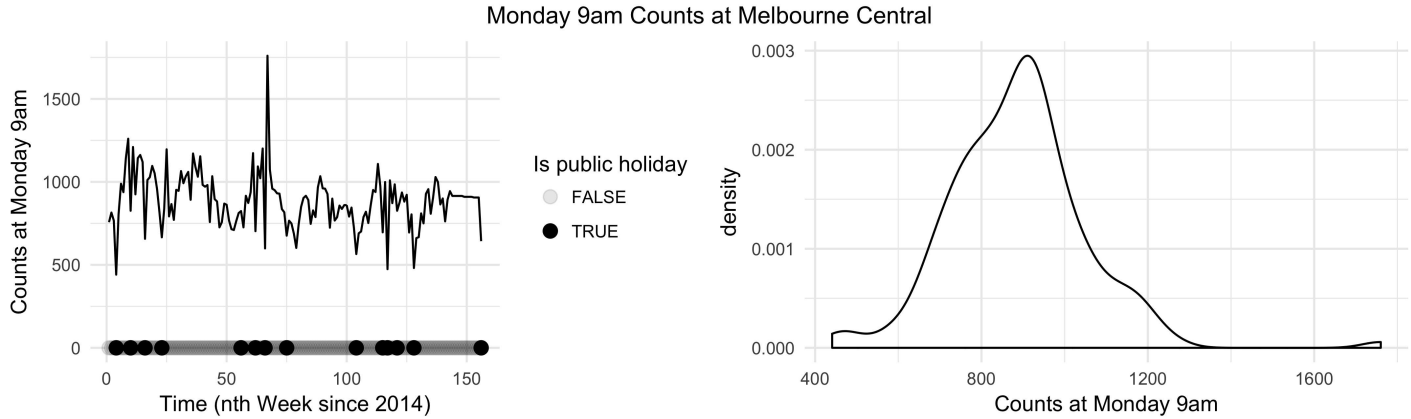
Out-of-sample cross validation





Predictive model using imputed data

- can perform analysis at all locations now with complete data



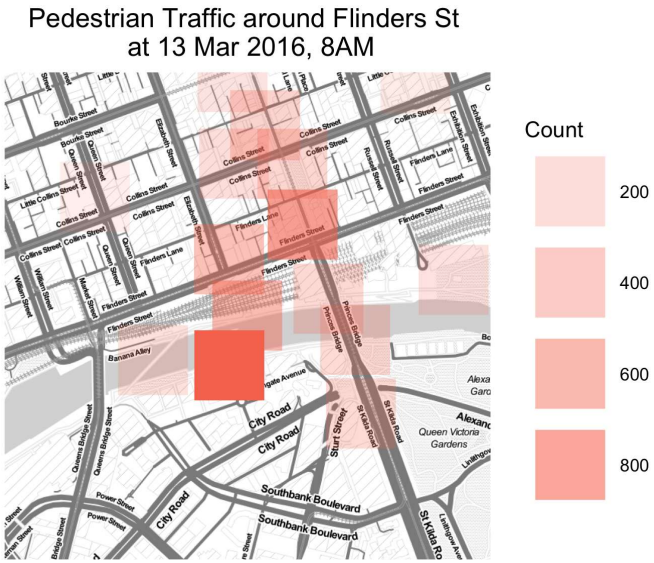
- can treat each specific hour of the week as a time series
  - find covariates outside the counts data to help predict counts
  - eg. weather data, public events

Objective: On-the-fly predictions of pedestrian traffic

- On-the-fly predictions based on glm computed on complete (imputed) training data (2014-2016), for each location
- Model objects are hundreds of megabytes
- Requires efficient data storage
- Modify the model object, to minimise the structure containing only what is necessary for prediction.

Future steps for finishing

- package into a Shiny app for easy on-the-fly predictions
  - make the predictive models more accessible
  - provide visualisations of the predictions



- perform simulations to test imputation algorithm for robustness
- explore predictions outside of counts data

Summary of my contributions

- Explored issues in the Melbourne CBD pedestrian data available from the City of Melbourne Open Data Portal
- Algorithm for imputing missings
- Efficient training model structures

Acknowledgements

- R and RStudio
- Packages used for this work were:
  - `dplyr`, `lubridate` and `tidyr` for data manipulation
  - `foreach` and `doSNOW` for parallel computing
  - `rwalkr` for accessing latest data (Thanks Earo!)
  - `ggplot2` and `ggmap` for visualisations
  - `xaringan` for compliling this presentation
- Code, presentation and thesis at github.com/gavinchin/honours2017

I'd like to thank Di Cook and Earo Wang for helping me with the project, as well as my fellow honours classmates.

Questions or comments?

Questions or comments?

Thanks for listening!