Final Report

Group 1: Jiawen Chen, Brooke Felsheim, Elena Kharitonova, Xinjie Qian, and Jiarui Tang

4/30/2022

Introduction

As human populations are rising across the world, so is the proportion of people that live in urban areas. Estimates from the *UN World Urbanization Prospects* (2018) indicate that over 4.2 billion people (55% of the global population) currently live in urban areas, and by 2050, an additional 2.5 billion people (68% of the global population) could be living in urban areas. More people living in urban areas calls for more space-, cost-, and energy-efficient systems of transportation as an alternative to cars. One such promising transportation alternative is the implementation of bicycle sharing programs.

Bicycle sharing programs are transportation schemes that allow individuals to rent bicycles on a short-term basis for either a set rate or for free. Most bicycle sharing programs have many computer-controlled bicycle rack "hubs" dispersed across a city that keep bikes locked and release them for use when a user enters the appropriate information/payment from a station or an app (Figure 1). A user can then ride the bike and return it to any other bicycle hub that is part of the same program. Many cities across the world have begun implementing bicycle sharing programs, including Chapel Hill, which has a Tar Heel Bikes sharing system[1]. Systems like these provide convenient, inexpensive, and eco-friendly transportation options for individuals residing in a city.

Figure 1: A 'hub' of bicycles belonging to the Santander Cycles system in London. SOPA Images/Lightrocket via Getty Images

Successful implementations of bike sharing programs depend on proper management of these systems. It is important for a bike sharing program to provide a stable supply of rental bikes to its population so its users feel that they can rely on the system for their transportation needs. The analysis of bike sharing data allows for a better understanding of the demand of rental bikes in a city, which, in turn, can help inform a city about how to provide appropriate supplies of rental bikes for its population. For our project, we were interested in predicting the number of bikes rented within a given bike sharing system given information about weather, time of day, and date. We were also interested in assessing the most important variables for predicting bike rental counts. To answer these questions, we fit and evaluated a negative binomial generalized mixed model, a conditional inference tree, and a random forest model, using data from three publicly available bike sharing demand datasets.

The first dataset we use is a London bike sharing demand dataset downloaded from Kaggle[2] and provided by

[1] https://move.unc.edu/bike/bikeshare/

[2] https://www.kaggle.com/datasets/hmavrodiev/london-bike-sharing-dataset

Transport for London[3]. This dataset contains hourly bike rental count observations over two years, from Jan 04 2015 - Jan 03 2017. The first full consecutive year of data was used as the training set in the analysis, and the second full consecutive year of data was held out as a test set.

The second dataset we use is a Seoul bike sharing demand dataset downloaded from the UCI Machine Learning Repository[4] and provided by the Seoul Metropolitan Government[5]. This dataset contains hourly bike rental counts over one year, from Dec 1 2017 - Nov 30 2018. This was used as an independent test set.

The third dataset we use is a Washington, D.C. bike sharing demand dataset downloaded from Kaggle[6] and provided by Capital Bikeshare[7]. This dataset contains hourly bike rental counts over two years, from Jan 01 2011 - Dec 31 2012. This was used as an independent test set.

Each dataset contained hourly observations of bike rental count data. To simplify our analysis, we chunked the hourly data into three time blocks: [0:00 - 8:00), [8:00 - 16:00), and [16:00 - 24:00). To calculate the total bike count, we summed the hourly bike counts for each of the time blocks. Additionally, because temperature and humidity can be correlated with time of day, we chose to use the maximum and minimum daily temperature and humidity measurements for each 8-hour data point in order to avoid any issues of colinearity.

We created an R package named `bikeSharing` that includes methods for training and evaluating the negative binomial glmm and random forest models, as well as the processed source data from all three sets.

The below code can be used to install the package and load the package library. The zipped package source data, `bikeSharing_1.0.0.tar.gz` can be found in the Github repository for our project[8].

```r
if(!require("bikeSharing", quietly = TRUE))
  install.packages("package/bikeSharing_1.0.0.tar.gz", repos = NULL)
library(bikeSharing)
```

Once the package is loaded, the bike sharing data from all three sets becomes easily accessible through the variable names `london`, `seoul`, and `dc`. To directly load them into the R environment, one can simply run:

```r
data("london", "seoul", "dc")
```

All three datasets contain bike rental count data as well as 11 additional weather-, time-, and date-related variables that were used as predictors in our models:

- Hour chunk (00:00 - 8:00, 8:00-16:00, 16:00-24:00)
- Weekend status (Yes/No)
- Holiday status (Yes/No)
- Season (Winter, Spring, Summer, Autumn)
- Minimum daily temperature (C)
- Maximum daily temperature (C)
- Minimum daily humidity (%)
- Maximum daily humidity (%)
- Wind speed (m/s)
- Presence of any rain or snow (Yes/No)
- Date (mm-dd)

The way that these variables are used within our models will be further described in the Methods section. For all of the analyses performed, the `london` dataset was divided into training and testing sets, where the training set contained all "Year 1" data (Jan 04 2015 - Jan 03 2016), and the testing set contained all "Year 2" data (Jan 04 2016 - Jan 03 2017).

```r
london_train <- london[london$Year == "Year 1",]
london_test <- london[london$Year == "Year 2",]
```

[3] https://cycling.data.tfl.gov.uk
[4] https://archive.ics.uci.edu/ml/datasets/Seoul+Bike+Sharing+Demand
[5] https://data.seoul.go.kr
[6] https://www.kaggle.com/datasets/marklvl/bike-sharing-dataset
[7] https://ride.capitalbikeshare.com/system-data
[8] https://github.com/brookefelsheim/bios735-group1

Methods

Negative Binomial Generalized Linear Mixed Model

Let y_{ij} be the number of bikes rented at hour chunk j of day i. Thus i ranges from 1 to 365, and j ranges from 1 to 3, corresponding to hour chunks [00:00 - 8:00), [8:00-16:00), [16:00-24:00). We assume that the number of bikes rented for a given hour chunk within a specific day follows a negative binomial distribution, so $Y_{ij} \sim NB(\mu_{ij}, \theta)$ using the Hilbe parameterization, so:

$$P(Y_{ij} = y_{ij}|\mu_{ij}, \theta) = \frac{\Gamma(y_{ij} + \theta)}{\Gamma(y_{ij} + 1)\Gamma(\theta)} \left(\frac{\theta}{\theta + \mu_{ij}}\right)^{\theta} \left(\frac{\mu_{ij}}{\theta + \mu_{ij}}\right)^{y_{ij}} \tag{1}$$

The mean of y_{ij} for each day i at hour chunk j is μ_{ij}, which we assume follows a negative binomial generalized linear mixed model with a random intercept. Thus it is determined from the following model:

$$\log(\mu_{ij}) = x_{ij}^T \beta + b_i \tag{2}$$

Let x_{ij} = (1, I(HourChunk$_{ij}$ = [8,16)), I(HourChunk$_{ij}$ = [16,24)), Weekend$_i$, Holiday$_i$, I(Season$_i$ = Spring), I(Season$_i$ = Summer), I(Season$_i$ = Winter), Min_Temperature$_i$, Max_Temperature$_i$, Min_Humidity$_i$, Max_Humidity$_i$, Wind_Speed$_{ij}$, Rain_or_Snow$_{ij}$)^T.

HourChunk$_{ij}$ is a categorical variable corresponding to the hour chunk j of day i, with the reference hour chunk being [0:00, 8:00), Weekend$_i$ is a binary variable with 1 if day i is a weekend, 0 if it is not. Holiday$_i$ is a binary variable with 1 if day i is a holiday, 0 if it is not. Season$_i$ is a categorical variable corresponding to which season day i is in, with the reference season being Autumn. The Min_Temperature$_i$ and Max_Temperature$_i$ are the minimum and maximum temperature of day i, respectively. Similarly, the Min_Humidity$_i$ and Max_Humidity$_i$ are the minimum and maximum humidity of day i, respectively. Wind_Speed$_{ij}$ is the average wind speed of hour chunk j of day i. Rain_or_Snow$_i$ is a binary variable with 1 if during day i hour chunk j there is any rain or snow, 0 if there is not. Thus $\beta = (\beta_0, \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6, \beta_7, \beta_8, \beta_9, \beta_{10}, \beta_{11}, \beta_{12}, \beta_{13})'$.

In Equation (2), b_i is the unobserved random effect that the day i has on the number of rented bikes at any given hour chunk. It is assumed that $b_i \sim N(0, \sigma_\gamma^2)$.

Thus, of interest is the estimate θ, β, and σ_γ^2. Equations 1 and 2 can be combined to obtain the following likelihood equation for a given day (i).

$$L(\theta, \beta, \sigma_\gamma^2|\mathbf{y}, \mathbf{b}) = \prod_{i=1}^{365} \prod_{j=1}^{3} p(y_{ij}) = \prod_{i=1}^{365} \left(\prod_{j=1}^{3} f(y_{ij}|\mu_{ij})\right) f(b_i|\sigma_\gamma^2) \tag{3}$$

Since the random effects b_i are unobservable, this means they must be integrated out of the above expression to obtain the likelihood, so:

$$L(\theta, \beta, \sigma_\gamma^2|\mathbf{y}, \mathbf{b}) = \prod_{i=1}^{365} \left[\int \left(\prod_{j=1}^{3} f(y_{ij}|\mu_{ij})\right) f(b_i|\sigma_\gamma^2)\, db_i\right]$$

$$= \prod_{i=1}^{365} \left[\int \prod_{j=1}^{3} \frac{\Gamma(y_{ij} + \theta)}{\Gamma(y_{ij} + 1)\Gamma(\theta)} \left(\frac{\theta}{\theta + e^{x_{ij}^T\beta + b_i}}\right)^{\theta} \left(\frac{e^{x_{ij}^T\beta + b_i}}{\theta + e^{x_{ij}^T\beta + b_i}}\right)^{y_{ij}} \frac{1}{\sqrt{2\pi\sigma_\gamma^2}} \exp\left(-\frac{b_i^2}{2\sigma^2}\right) db_i\right]$$

So thus, the log likelihood is found to be

$$l(\theta,\beta,\sigma_\gamma^2) = \sum_{i=1}^{365} log\Big[\int \prod_{j=1}^{3} \frac{\Gamma(y_{ij}+\theta)}{\Gamma(y_{ij}+1)\Gamma(\theta)} \left(\frac{\theta}{\theta + e^{x_{ij}^T\beta+b_i}}\right)^{\theta} \cdot \left(\frac{e^{x_{ij}\beta+b_i}}{\theta+e^{x_{ij}\beta+b_i}}\right)^{y_{ij}} \frac{1}{\sqrt{2\pi\sigma_\gamma^2}} \exp\left(-\frac{b_i^2}{2\sigma_\gamma^2}\right) db_i\Big]$$

This log-likelihood will be maximized to obtain estimates for $\theta, \beta, \sigma_\gamma^2$ through an MCEM approach. Assuming that the b_i's were known, we first define the complete data log likelihood as:

$$\log L_C(\theta,\beta,\sigma_\gamma^2|\mathbf{y},\mathbf{b}) = \log\left[\prod_{i=1}^{365}\left(\prod_{j=1}^{3} f(y_{ij}|\mu_{ij})f(b_i|\sigma_\gamma^2)\right)\right] = \sum_{i=1}^{365}\log\left(\prod_{j=1}^{3} f(y_{ij}|\mu_{ij})f(b_i|\sigma_\gamma^2)\right)$$

$$\text{So, } l_C(\theta,\beta,\sigma_\gamma^2|\mathbf{y},\mathbf{b}) = \sum_{i=1}^{365}\left(\sum_{j=1}^{3}\log f(y_{ij}|\mu_{ij}) + \log f(b_i|\sigma_\gamma^2)\right)$$

Thus, for the MCEM algorithm, we will be maximizing the expectation of the complete data log likelihood, otherwise known as the Q-function at step t. So the Q-function is defined as the following:

$$\begin{aligned} Q(\theta,\beta,\sigma_\gamma^2|y,\theta^{(t)},\beta^{(t)},\sigma_\gamma^{2\,(t)}) &= E[l_C(\theta,\beta,\sigma_\gamma^2|\mathbf{y},\mathbf{b})|y,\theta^{(t)},\beta^{(t)},\sigma_\gamma^{2\,(t)}] \\ &= E\left[\sum_{i=1}^{365}\left(\sum_{j=1}^{3}\log f(y_{ij}|\mu_{ij}^{(t)}) + \log f(b_i|\sigma_\gamma^{2\,(t)})\right)\right] \end{aligned}$$

$$= \sum_{i=1}^{365}\left[\int\left(\sum_{j=1}^{3}\log f(y_{ij}|\mu_{ij}^{(t)}) + \log f(b_i|\sigma_\gamma^{2\,(t)})\right) f(b_i|y,\theta^{(t)},\beta^{(t)},\sigma_\gamma^{2\,(t)})db_i\right]$$

Where $\mu_{ij}^{(t)} = e^{x_{ij}^T\beta^{(t)}+b_i}$ and $f(b_i|y,\theta^{(t)},\beta^{(t)},\sigma_\gamma^{2\,(t)})$ is the posterior distribution of b_i given the observed data and the current parameter estimates. Let $Y_i = (y_{i1}, y_{i2}, y_{i3})^T$. The density function for b_i given $Y_i, \beta^{(t)}, \theta^{(t)}, \sigma_\gamma^{2\,(t)}$ is:

$$\begin{aligned} f(b_i|Y_i,\theta^{(t)},\beta^{(t)},\sigma_\gamma^{2\,(t)}) &\propto f(Y_i|b_i,\theta^{(t)},\beta^{(t)},\sigma_\gamma^{2\,(t)})\cdot f(b_i|\sigma_\gamma^{2\,(t)}) \\ &\propto \prod_{j=1}^{3}\left(\frac{\theta^{(t)}}{\mu_{ij}^{(t)}+\theta^{(t)}}\right)^{\theta^{(t)}}\cdot\left(\frac{\mu_{ij}^{(t)}}{\mu_{ij}^{(t)}+\theta^{(t)}}\right)^{y_{ij}}\cdot\exp\left(-\frac{b_i^2}{2\sigma_\gamma^{2\,(t)}}\right) \\ &\propto \sum_{j=1}^{3}\left[(y_{ij}\log\mu_{ij}^{(t)} - (y_{ij}+\theta^{(t)})\log(\mu_{ij}^{(t)}+\theta^{(t)})) - \frac{b_i^2}{2\sigma_\gamma^{2\,(t)}}\right] \end{aligned} \tag{4}$$

If we could sample from this posterior distribution of b_i, we could approximate this integral using a montecarlo approach. So,

$$Q(\theta,\beta,\sigma_\gamma^2|y,\theta^{(t)},\beta^{(t)},\sigma_\gamma^{2\,(t)}) = \frac{1}{M}\sum_{i=1}^{365}\sum_{k=1}^{M}\left(\sum_{j=1}^{3}\log f(y_{ij}|\mu_{ijk}^{(t)}) + \log f(b_{ik}|\sigma_\gamma^{2\,(t)})\right)$$

Where b_{ik} is one of the M samples of the posterior distribution b_i and $\mu_{ijk}^{(t)} = e^{x_{ij}^T\beta^{(t)}+b_{ik}}$.

To sample from this distribution posterior distribution of b_i, we employed the metropolis hastings algorithm with a random walk. So at set t, for each new b_i, we considered $b_i^* = b_i^{(t)} + \epsilon$ where $\epsilon \sim U\left(-\frac{1}{4}, \frac{1}{4}\right)$. From this, the Metropolis Hastings ratio $R(b_i^{(t)}, b_i^*)$ was determined whether or not to accept this new b_i^*. For each new b_I^*

$$R(b_i^{(t)}, b_i^*) = \frac{f(y_{ij}|\theta, \beta, \sigma_\gamma^2, b_i^*) f(b_i^*|\sigma_\gamma^2)}{f(y_{ij}|\theta, \beta, \sigma_\gamma^2, b_i^{(t)}) f(b_i^{(t)}|\sigma_\gamma^2)}$$

Thus, the new value for the $b_i^{(t+1)}$ is:

$$b_i^{(t+1)} = \begin{cases} b_i^* & \text{with probability } \min(R(b_i^{(t)}, b_i^*), 1) \\ b_i^{(t)} & \text{otherwise} \end{cases}$$

For the M-step, we will maximize the Q-function with respect to $\theta, \beta, \sigma_\gamma^2$. The Nelder-Mead method was used to optimize all of these parameters, this was because Nelder-Mead in the M-step does not require 1st or 2nd derivatives and it is robust. After this, the E-step and M-step are repeated for each iteration until the estimates for $\theta, \beta, \sigma_\gamma^2$ converge.

In our package `bikeSharing`, we have created a function `MCEM_algorithm` that will fit a Negative Binomial Generalized Linear Mixed Model to the data using the MCEM algorithm described above.

The `MCEM_algorithm` function takes as input `beta_initial`, which are initial guesses for the β vector. `theta_initial` which is the initial guess for the θ vector. `s2gamma_initial` which is the initial guess for the σ_γ^2 vector. `M` which is the number of posterior b_i's to be sampled for the Monte Carlo integration. `burn.in` which is the number of posterior b_i's to be thrown out to account for the time it takes for the chain to converge. `tol` which is the percent difference between the Q functions of two iteration to reach convergence. `maxit` which is the maximum number of iterations allowed. `data`, which is the data to which the negative binomial GLMM model should be fit to.

```
set.seed(1)
glmm_fit <- MCEM_algorithm( beta_initial = c(8.3, 1.5, 1.5, -0.25, -0.50, 0,
                                             0, -0.25, 0, 0, 0, 0, 0, -0.25),
                         theta_initial = 10,
                         s2gamma_initial = 0.2,
                         M = 1000,
                         burn.in = 200,
                         tol = 10^-4,
                         maxit = 100,
                         data = london_train
                         )
```

```
[1] -10009.51
Maximizing -- use negfn and neggr
Iter: 1 Qf: -10009.508 s2gamma: 0.139693 Intercept: 8.288
Hour_Chunks[8,16):1.515  Hour_chunks[16,24): 1.426  Is_weekend:-0.258
Is_holiday:-0.445 SeasonSpring:0.007 SeasonSummer:0.020 SeasonWinter:-0.289
Min_temp:0.006 Max_temp:-0.005 Min_humidity:-0.000 Max_humidity:0.000
Wind_speed:-0.002 Rain_or_snow:-0.200 theta:10.380 eps:0.989990
[1] -9927.993
Maximizing -- use negfn and neggr
Iter: 2 Qf: -9927.993 s2gamma: 0.115304 Intercept: 8.288
Hour_Chunks[8,16):1.492  Hour_chunks[16,24): 1.385  Is_weekend:-0.253
Is_holiday:-0.404 SeasonSpring:0.056 SeasonSummer:0.093 SeasonWinter:-0.264
Min_temp:0.003 Max_temp:-0.003 Min_humidity:0.000 Max_humidity:-0.001
```

```
Wind_speed:0.001 Rain_or_snow:-0.183 theta:12.726 eps:0.008144
[1] -9837.183
Maximizing -- use negfn and neggr
Iter: 3 Qf: -9837.183 s2gamma: 0.081402 Intercept: 8.219
Hour_Chunks[8,16):1.529  Hour_chunks[16,24): 1.414  Is_weekend:-0.272
Is_holiday:-0.392 SeasonSpring:0.035 SeasonSummer:0.099 SeasonWinter:-0.242
Min_temp:0.008 Max_temp:-0.004 Min_humidity:-0.001 Max_humidity:-0.000
Wind_speed:0.000 Rain_or_snow:-0.192 theta:14.086 eps:0.009147
[1] -9763.714
Maximizing -- use negfn and neggr
Iter: 4 Qf: -9763.714 s2gamma: 0.070620 Intercept: 8.291
Hour_Chunks[8,16):1.547  Hour_chunks[16,24): 1.424  Is_weekend:-0.298
Is_holiday:-0.388 SeasonSpring:0.023 SeasonSummer:0.089 SeasonWinter:-0.221
Min_temp:0.013 Max_temp:-0.005 Min_humidity:-0.002 Max_humidity:-0.000
Wind_speed:-0.006 Rain_or_snow:-0.174 theta:16.304 eps:0.007468
[1] -9692.282
Maximizing -- use negfn and neggr
Iter: 5 Qf: -9692.282 s2gamma: 0.059093 Intercept: 8.240
Hour_Chunks[8,16):1.517  Hour_chunks[16,24): 1.397  Is_weekend:-0.308
Is_holiday:-0.360 SeasonSpring:0.015 SeasonSummer:0.092 SeasonWinter:-0.228
Min_temp:0.013 Max_temp:-0.001 Min_humidity:-0.003 Max_humidity:-0.000
Wind_speed:-0.001 Rain_or_snow:-0.174 theta:17.007 eps:0.007316
[1] -9655.907
Maximizing -- use negfn and neggr
Iter: 6 Qf: -9655.907 s2gamma: 0.054342 Intercept: 8.222
Hour_Chunks[8,16):1.516  Hour_chunks[16,24): 1.405  Is_weekend:-0.312
Is_holiday:-0.361 SeasonSpring:0.003 SeasonSummer:0.099 SeasonWinter:-0.219
Min_temp:0.013 Max_temp:-0.000 Min_humidity:-0.003 Max_humidity:-0.000
Wind_speed:-0.002 Rain_or_snow:-0.174 theta:17.768 eps:0.003753
[1] -9624.324
Maximizing -- use negfn and neggr
Iter: 7 Qf: -9624.324 s2gamma: 0.047064 Intercept: 8.208
Hour_Chunks[8,16):1.528  Hour_chunks[16,24): 1.409  Is_weekend:-0.322
Is_holiday:-0.383 SeasonSpring:-0.007 SeasonSummer:0.109 SeasonWinter:-0.221
Min_temp:0.014 Max_temp:-0.000 Min_humidity:-0.003 Max_humidity:-0.000
Wind_speed:-0.006 Rain_or_snow:-0.172 theta:18.032 eps:0.003271
[1] -9601.023
Maximizing -- use negfn and neggr
Iter: 8 Qf: -9601.023 s2gamma: 0.040836 Intercept: 8.171
Hour_Chunks[8,16):1.525  Hour_chunks[16,24): 1.407  Is_weekend:-0.328
Is_holiday:-0.398 SeasonSpring:-0.009 SeasonSummer:0.111 SeasonWinter:-0.205
Min_temp:0.016 Max_temp:0.001 Min_humidity:-0.004 Max_humidity:0.000
Wind_speed:-0.007 Rain_or_snow:-0.173 theta:17.536 eps:0.002421
[1] -9579.64
Maximizing -- use negfn and neggr
Iter: 9 Qf: -9579.640 s2gamma: 0.038947 Intercept: 8.160
Hour_Chunks[8,16):1.529  Hour_chunks[16,24): 1.415  Is_weekend:-0.325
Is_holiday:-0.387 SeasonSpring:-0.011 SeasonSummer:0.110 SeasonWinter:-0.194
Min_temp:0.016 Max_temp:0.001 Min_humidity:-0.004 Max_humidity:0.000
Wind_speed:-0.009 Rain_or_snow:-0.178 theta:18.146 eps:0.002227
[1] -9566.678
Maximizing -- use negfn and neggr
Iter: 10 Qf: -9566.678 s2gamma: 0.035977 Intercept: 8.140
Hour_Chunks[8,16):1.534  Hour_chunks[16,24): 1.417  Is_weekend:-0.332
```

```
Is_holiday:-0.381 SeasonSpring:-0.015 SeasonSummer:0.106 SeasonWinter:-0.190
Min_temp:0.017 Max_temp:0.001 Min_humidity:-0.004 Max_humidity:0.000
Wind_speed:-0.011 Rain_or_snow:-0.180 theta:18.152 eps:0.001353
[1] -9558.14
Maximizing -- use negfn and neggr
Iter: 11 Qf: -9558.140 s2gamma: 0.034944 Intercept: 8.185
Hour_Chunks[8,16):1.539  Hour_chunks[16,24): 1.426  Is_weekend:-0.345
Is_holiday:-0.400 SeasonSpring:-0.017 SeasonSummer:0.100 SeasonWinter:-0.182
Min_temp:0.019 Max_temp:0.001 Min_humidity:-0.005 Max_humidity:0.000
Wind_speed:-0.013 Rain_or_snow:-0.178 theta:17.895 eps:0.000892
[1] -9550.829
Maximizing -- use negfn and neggr
Iter: 12 Qf: -9550.829 s2gamma: 0.033988 Intercept: 8.221
Hour_Chunks[8,16):1.534  Hour_chunks[16,24): 1.421  Is_weekend:-0.347
Is_holiday:-0.401 SeasonSpring:-0.018 SeasonSummer:0.101 SeasonWinter:-0.182
Min_temp:0.019 Max_temp:0.001 Min_humidity:-0.005 Max_humidity:0.000
Wind_speed:-0.014 Rain_or_snow:-0.177 theta:17.985 eps:0.000765
[1] -9545.979
Maximizing -- use negfn and neggr
Iter: 13 Qf: -9545.979 s2gamma: 0.034305 Intercept: 8.276
Hour_Chunks[8,16):1.535  Hour_chunks[16,24): 1.423  Is_weekend:-0.341
Is_holiday:-0.392 SeasonSpring:-0.033 SeasonSummer:0.096 SeasonWinter:-0.186
Min_temp:0.019 Max_temp:0.000 Min_humidity:-0.005 Max_humidity:-0.000
Wind_speed:-0.014 Rain_or_snow:-0.174 theta:18.295 eps:0.000508
[1] -9541.885
Maximizing -- use negfn and neggr
Iter: 14 Qf: -9541.885 s2gamma: 0.033014 Intercept: 8.282
Hour_Chunks[8,16):1.522  Hour_chunks[16,24): 1.412  Is_weekend:-0.342
Is_holiday:-0.389 SeasonSpring:-0.045 SeasonSummer:0.088 SeasonWinter:-0.190
Min_temp:0.019 Max_temp:0.000 Min_humidity:-0.006 Max_humidity:0.000
Wind_speed:-0.013 Rain_or_snow:-0.186 theta:18.380 eps:0.000429
[1] -9538.622
Maximizing -- use negfn and neggr
Iter: 15 Qf: -9538.622 s2gamma: 0.032616 Intercept: 8.321
Hour_Chunks[8,16):1.531  Hour_chunks[16,24): 1.411  Is_weekend:-0.346
Is_holiday:-0.399 SeasonSpring:-0.046 SeasonSummer:0.074 SeasonWinter:-0.183
Min_temp:0.020 Max_temp:0.000 Min_humidity:-0.006 Max_humidity:-0.000
Wind_speed:-0.014 Rain_or_snow:-0.181 theta:18.497 eps:0.000342
[1] -9534.839
Maximizing -- use negfn and neggr
Iter: 16 Qf: -9534.839 s2gamma: 0.032440 Intercept: 8.334
Hour_Chunks[8,16):1.536  Hour_chunks[16,24): 1.414  Is_weekend:-0.343
Is_holiday:-0.391 SeasonSpring:-0.046 SeasonSummer:0.077 SeasonWinter:-0.182
Min_temp:0.020 Max_temp:0.000 Min_humidity:-0.006 Max_humidity:-0.000
Wind_speed:-0.014 Rain_or_snow:-0.175 theta:18.442 eps:0.000397
[1] -9531.502
Maximizing -- use negfn and neggr
Iter: 17 Qf: -9531.502 s2gamma: 0.031958 Intercept: 8.335
Hour_Chunks[8,16):1.537  Hour_chunks[16,24): 1.412  Is_weekend:-0.346
Is_holiday:-0.401 SeasonSpring:-0.047 SeasonSummer:0.078 SeasonWinter:-0.182
Min_temp:0.020 Max_temp:0.000 Min_humidity:-0.006 Max_humidity:-0.000
Wind_speed:-0.013 Rain_or_snow:-0.172 theta:18.532 eps:0.000350
[1] -9526.663
Maximizing -- use negfn and neggr
```

```
Iter: 18 Qf: -9526.663 s2gamma: 0.031245 Intercept: 8.339
Hour_Chunks[8,16):1.535  Hour_chunks[16,24): 1.410  Is_weekend:-0.346
Is_holiday:-0.405 SeasonSpring:-0.050 SeasonSummer:0.078 SeasonWinter:-0.180
Min_temp:0.020 Max_temp:0.000 Min_humidity:-0.006 Max_humidity:-0.000
Wind_speed:-0.013 Rain_or_snow:-0.174 theta:18.638 eps:0.000508
[1] -9525.671
Maximizing -- use negfn and neggr
Iter: 19 Qf: -9525.671 s2gamma: 0.030310 Intercept: 8.340
Hour_Chunks[8,16):1.527  Hour_chunks[16,24): 1.405  Is_weekend:-0.347
Is_holiday:-0.415 SeasonSpring:-0.052 SeasonSummer:0.073 SeasonWinter:-0.177
Min_temp:0.021 Max_temp:0.001 Min_humidity:-0.007 Max_humidity:-0.000
Wind_speed:-0.012 Rain_or_snow:-0.179 theta:18.449 eps:0.000104
[1] -9523.732
Maximizing -- use negfn and neggr
Iter: 20 Qf: -9523.732 s2gamma: 0.029895 Intercept: 8.338
Hour_Chunks[8,16):1.528  Hour_chunks[16,24): 1.410  Is_weekend:-0.344
Is_holiday:-0.401 SeasonSpring:-0.052 SeasonSummer:0.074 SeasonWinter:-0.177
Min_temp:0.020 Max_temp:0.001 Min_humidity:-0.007 Max_humidity:-0.000
Wind_speed:-0.013 Rain_or_snow:-0.179 theta:18.310 eps:0.000204
[1] -9518.461
Maximizing -- use negfn and neggr
Iter: 21 Qf: -9518.461 s2gamma: 0.029774 Intercept: 8.343
Hour_Chunks[8,16):1.531  Hour_chunks[16,24): 1.411  Is_weekend:-0.342
Is_holiday:-0.393 SeasonSpring:-0.053 SeasonSummer:0.074 SeasonWinter:-0.176
Min_temp:0.020 Max_temp:0.001 Min_humidity:-0.007 Max_humidity:-0.000
Wind_speed:-0.013 Rain_or_snow:-0.177 theta:18.560 eps:0.000553
[1] -9520.844
Maximizing -- use negfn and neggr
Iter: 22 Qf: -9520.844 s2gamma: 0.029349 Intercept: 8.348
Hour_Chunks[8,16):1.533  Hour_chunks[16,24): 1.415  Is_weekend:-0.340
Is_holiday:-0.399 SeasonSpring:-0.063 SeasonSummer:0.067 SeasonWinter:-0.177
Min_temp:0.020 Max_temp:0.001 Min_humidity:-0.007 Max_humidity:-0.000
Wind_speed:-0.014 Rain_or_snow:-0.177 theta:18.519 eps:0.000250
[1] -9520.354
Maximizing -- use negfn and neggr
Iter: 23 Qf: -9520.354 s2gamma: 0.029582 Intercept: 8.353
Hour_Chunks[8,16):1.534  Hour_chunks[16,24): 1.415  Is_weekend:-0.337
Is_holiday:-0.393 SeasonSpring:-0.064 SeasonSummer:0.067 SeasonWinter:-0.179
Min_temp:0.020 Max_temp:0.001 Min_humidity:-0.007 Max_humidity:-0.000
Wind_speed:-0.014 Rain_or_snow:-0.176 theta:18.422 eps:0.000051
There were 22 warnings (use warnings() to see them)
```

The output of the `MCEM_algorithm` function contains the list of `beta`, the final iteration of the fitted $\hat{\beta}$ vector. It contains `s2gamma` which is $\hat{\sigma_\gamma^2}$ at the final iteration. `theta` which is the final iteration of $\hat{\theta}$. `eps` which is the difference between this iteration and the last. `qfunction` which is the value of the Q function at the final iteration. `day_ranef` which contains all of the fitted b_i's. `iter` is the iteration at which convergence was reached.

```r
str(glmm_fit)
```

```
## List of 7
##  $ beta     : num [1:14] 8.353 1.534 1.415 -0.337 -0.393 ...
##  $ s2gamma  : num 0.0296
##  $ theta    : num 18.4
##  $ eps      : num 5.15e-05
```

```
##  $ qfunction: num -9520
##  $ day_ranef: num [1:365] 0.0653 -0.398 -0.5165 -0.2612 -0.0374 ...
##  $ iter     : num 23
```

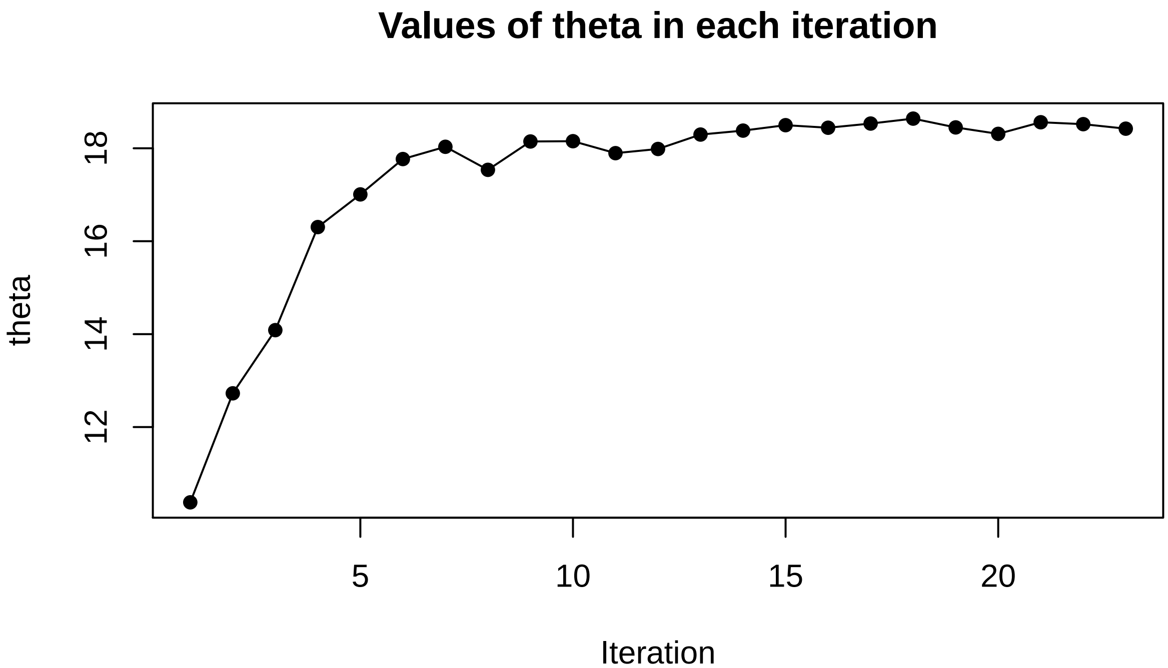


Figure 2: Values of theta in each iteration of the run of `MCEM_algorithm` on the London training set.

As we can see in Figure 2, the estimation of $\hat{\theta}$ converges quickly and from 9^{th} iteration, it starts to float near the true θ.

The size of the coefficients of the binary and categorical variables can be compared directly to determine how important that variable is in determining bike count. As we can see from above, Hour Chunks [8,16) and Hour Chunks [16,24) are the covariates with the largest coefficients in magnitude, implying that they are very important in determining the number of bikes rented. As we can see, between 8:00 - 16:00 and 16:00 - 24:00, many more people rent bikes in London than between 0:00 - 8:00. This result is not surprising. Weekend and Holiday are the next binary variables with the largest coefficients in magnitude. Both holidays and weekends have a similar effect on bike count; less people rent bikes on holidays and weekends. Additionally, rain or snow causes people to rent less bikes. Winter has a larger effect on bike counts than spring or summer compared to autumn, with significantly less people renting bikes in winter than autumn. Less bikes are rented in spring than autumn although the effect is not as large as the effect of winter. Summer leads to an increase in bike rentals compared to autumn. Min and Max Temperature do not play as significant of a role in determining the number of bike counts, but this effect may be masked by the effects of the season. Humidity does not seem to play a large role in determining bike counts. Wind speed has a small negative coefficient, meaning that with increased wind speed, less people rent bikes.

Machine learning models

Next, we applied two machine learning methods to predict the bike count. The first model we used is a conditional inference tree. In traditional partitioning, all possible splits are investigated to find the best split, which results in overfitting and selection. The conditional inference tree embeds the partition step with a permutation test, thereby enabling this method to be robust to covariates of different scales. Furthermore, it is capable of stopping when no significant correlation exists between the covariates and the response (Hothorn, Hornik, and Zeileis 2006). We employed the ctree funciton in the partykit to fit the conditional inference tree.

We visualized the three layer conditional inference tree in order to verify the relationship between bike count and variables (Figure 2). It is apparent that people tend to rent fewer bikes between midnight and 8 am. Maximum temperatures and humidity also affect bike rental rates. The splits that are employed in the conditional reference tree provide well reasoned explanations of the data structure, which means high accuracy predictions are made.

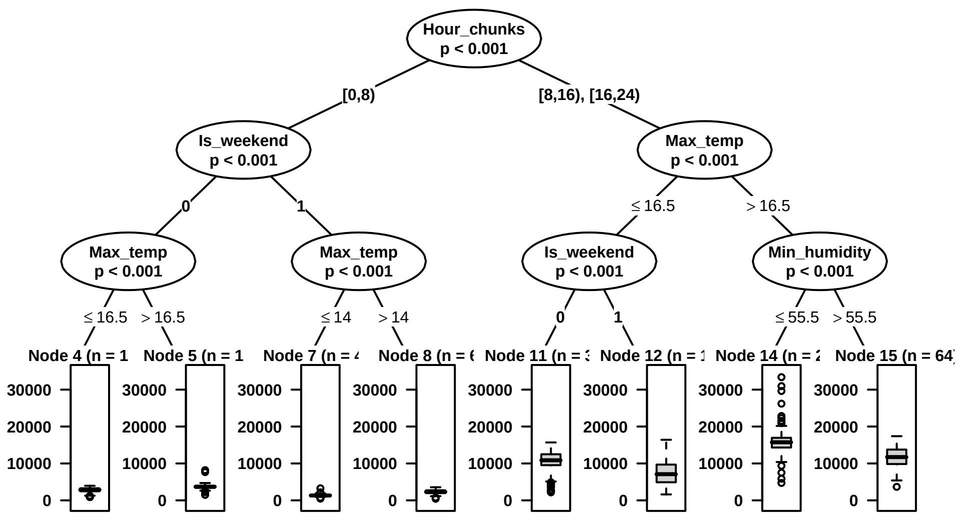


Figure 3: Three layer conditional inference tree

A random forest model was then applied to predict the number of bikes that would be rented. Random forest is composed of many decision trees as opposed to a single tree, resulting in a more accurate result. Incorporating the randomness allows random forest to protect against overfitting and can be applied more effectively to other data sets. The disadvantage of the random forest is its computational complexity. To fit the random forest, we created a method `train_random_forest()` within our `bikeSharing` R package that leverages the the `train()` function within the `caret` R package. The optimal tuning parameter mtry was determined using a 5-fold cross validation. We visualized a random selected tree.

```
rf_fit <- train_random_forest(data = london_train)
```

Random forest sample trees share similar splits with conditional inference trees (Figure 4). The estimated bike rental count is much higher in the hour chunks that are not 0-8 am. Additionally, the estimated count is high when the temperature is more than 22.25 degrees.

Variable Importance in machine learning models

To investigate which variables affect the prediction most, we calculate the importance of the variables in the conditional inference tree and the random forest. In conditional inference tree, we calculated the mean decrease in accuracy when deleting a variable (Table 1).

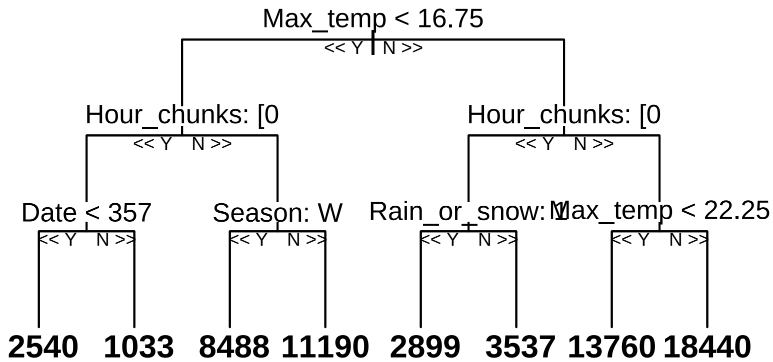


Figure 4: A random selected tree in the trained random forest model

Table 1: Mean decrease in accuracy of variables in the conditional inference tree

Variable	Mean decrease in accuracy
Hour_chunks	42292743.475
Max_temp	7305786.627
Is_weekend	2819028.041
Rain_or_snow	1622451.011
Min_humidity	1014266.239
Season	905987.474
Max_humidity	184868.121
Wind_speed	35027.139
Is_holiday	25815.318
Date	3737.395

In random forest, we calculate the increase in MSE when deleting a variable (Figure 5). The code to do this was implemented as a function `plot_rf_importance` in the `bikeSharing` R package.

```r
plot_rf_importance(london_train)
```

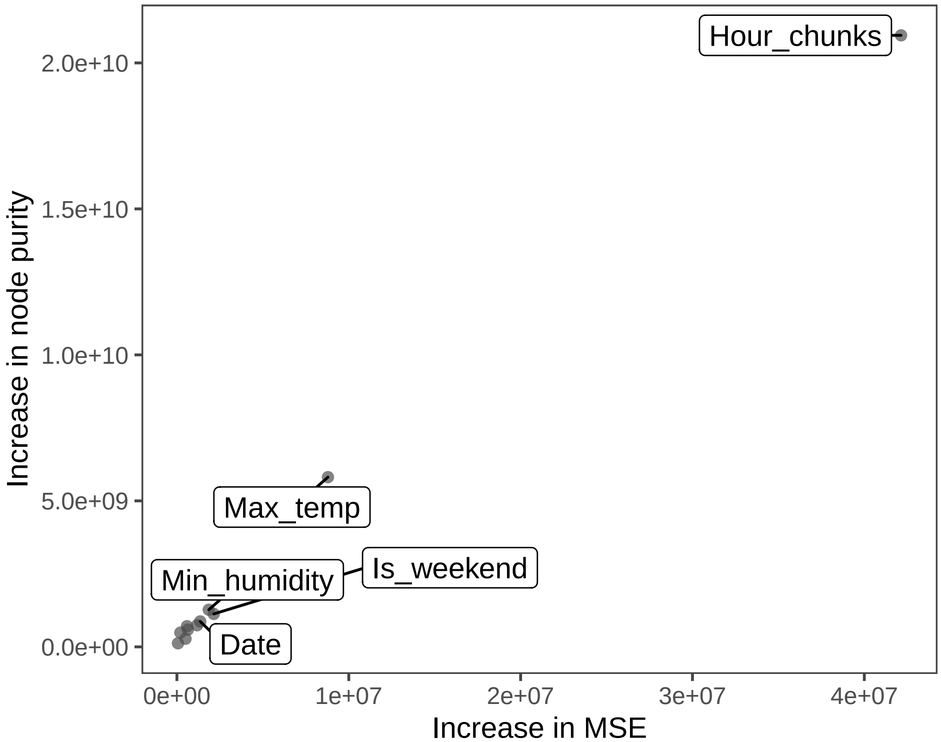


Figure 5: Feature importance in the random forest model

The top important variables of the conditional inference tree and random forest are similar. The hour chunk is the most important variable in both models, which agrees with the results from the negative binomial GLMM method. For both models, the next most important variable is the Maximum temperature, which is different from the negative GLMM model. Additional details of the variables are discussed in the discussion section. We further compare the performance of random forest and conditional inference tree using the second year bike renting data in London. Conditional inference tree results in a $R^2 = 0.81$ and random forest has $R^2 = 0.91$. Due the similarity in these two models and the better fit of the random forest model to the training set, we selected random forest tree for use in further comparison.

Methods for Model Fit Assessment

The accuracy of the random forest model and the negative binomial GLMM model will be assessed by using the models to predict the values of bike counts for for the London (training set), London (test set), Seoul, and DC data sets. The predicted values will be compared to the actual values to determine how well the models predict the bike counts for a city on a given day and time.

Three metrics of accuracy will be calculated: Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Coeficient fo Determination (R^2). They will be calculated using the following formulas.

- RMSE $= \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}$
- MAE $= \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$
- $R^2 = 1 - \frac{\sum_{i=1}^{N}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{N}(y_i - \bar{y}_i)^2}$

Results

Both the negative binomial GLMM and the random forest model were used to predict the the values of bike counts for for the London (training set), London (test set), Seoul, and DC data sets. Since Seoul and DC have a different population size than London, it is possible that the predicted value based on the trained London data set and the actual city bike count may be on different scale. In order to account for this, the predicted values for the Seoul and DC data sets were scaled by $\frac{\text{Average Bike Count of City}}{\text{Average Bike Count of London}}$, so that the predicted value for DC and Seoul would be on the same scale as the actual observed bike counts.

The assessment of model fit for the GLMM model was wrapped into the function `glmm_model_fit()`. It takes as input a negative binomial model `model_glmm` which is the output of the `MCEM_algorithm` function, `data`, the data set for which bike counts will be predicted, `scale_to_reference_mean` to determine whether the predicted values should be scaled, and `reference` which is the city that the model was fit to. The output of this function is a vector of RMSE, MAE, and R^2 values. We apply this function to the London training, the London test, the DC data, and the Seoul data.

```
glmm_model_fit(model_glmm = glmm_fit, data = london_train, scale_to_reference_mean = "no",
               reference = london)
```

```
##       RMSE      MAE        R2
## 1 1886.267 1291.106 0.8831618
```

```
glmm_model_fit(model_glmm = glmm_fit, data = london_test, scale_to_reference_mean = "no",
               reference = london)
```

```
##       RMSE      MAE        R2
## 1 2491.293 1647.064 0.8142036
```

```
glmm_model_fit(model_glmm = glmm_fit, data = dc, scale_to_reference_mean = "yes",
               reference = london)
```

```
##      RMSE     MAE       R2
## 1 845.741 605.217 0.521788
```

```
glmm_model_fit(glmm_fit, data = seoul, scale_to_reference_mean = "yes",
               reference = london)
```

```
##       RMSE      MAE        R2
## 1 3519.413 2719.021 0.4999935
```

Unsurprisingly, we see that the GLMM model fits the London test data set the best out of the test sets. There does not seem to be a large difference between the R^2 values for the DC and Seoul data set, however the DC data set has a smaller RMSE and MAE, implying that the DC bike counts may be less spread out than the Seoul bike counts.

A similar assessment for the fit of the random forests model was wrapped into the function `rf_model_fit()`. It takes as input a random forest model, `model`, which is the output of the `train_random_forest` function, `data`, a data set for which bike counts will be predicted applied to, `scale_to_reference_mean` to determine whether the predicted values should be scaled, and `reference` which is the city that the model was fit to. The output of this function is a vector of RMSE, MAE, and R^2 values. We apply this function to the London training, the London test, the DC data, and the Seoul data.

```
rf_model_fit(model = rf_fit, data = london_train, scale_to_reference_mean = "no",
             reference = london)
```

```
##       RMSE      MAE        R2
## 1 752.2336 471.5412 0.9822717
```

```
rf_model_fit(rf_fit, data = london_test, scale_to_reference_mean = "no",
             reference = london)
```

```
##       RMSE     MAE        R2
## 1 1789.24 1191.32 0.9084533
```

```r
rf_model_fit(rf_fit, data = dc, scale_to_reference_mean = "yes",
             reference = london)
```

```
##        RMSE      MAE        R2
## 1 664.5474 492.4647 0.6737107
```

```r
rf_model_fit(rf_fit, data = seoul, scale_to_reference_mean = "yes",
             reference = london)
```

```
##        RMSE      MAE        R2
## 1 3163.613 2479.835 0.5658297
```

Overall, the random forest model performed better than the negative binomial GLMM in all cases. The random forest model fits the initial training data significantly better than the negative binomial GLMM does. This is not surprising as random forests can account for interactions between covariates while the GLMM model does not. Additionally, the random forest model fits the London test data set better than the GLMM model. Overall, both models seem to perform relatively well in predicting London bike rental counts of the next year (the London training dataset). The random forest model performs significantly better than the GLMM model for DC, however both models perform relatively poorly for the Seoul data set. Thus, it seems that both the GLMM model and the random forest model can predict bike counts in the same city as the testing data set relatively well, however both models do not extrapolate well to other cities.

To determine why this is the case, the average of the covariates for each of the cities was examined to see if there was a major difference between the covariates between cities. (Table 2 below)

Table 2: Average of Covariates From All Data Sets

City	Average Number of Holidays	Average Min Temp	Average Max Temp	Average Min Humidity	Average Max Humidity	Average Wind Speed	Average Rain/Snow Days
London - Train	1.021918	9.576712	15.63836	55.44795	86.58356	4.636123	1.346119
London - Test	1.022018	9.413303	15.44908	56.60183	87.92477	4.227260	1.321101
DC	1.028807	11.384115	19.63523	43.61088	82.14952	3.545157	1.222222
Seoul	1.048159	8.831445	17.45694	37.41643	77.88385	1.726357	1.205855

As we can see, there does not appear to be any major differences in the average of the covariates between the cities. This implies that the issue in applying the fitted models to the new cities is not the range of the covariates, such as the temperature range, humidity range, wind speed range, the number of holidays or the number of rain/snow days. Therefore, to further determine the cause of the poor fit of the model in DC and Seoul, the distribution of the bike counts is examined for each of the data sets below (Figure 6). The DC bike counts and the seoul bike counts were scaled by $\frac{\text{Average Bike Count of London}}{\text{Average Bike Count of City}}$. This was done so that the distribution of bike counts would be on the same scale as the London data set, so that the distributions could be compared more directly.

The London training and test set have a very similar distribution, which explains why both the random forest model and the negative binomial glmm model predict the bike counts well for the London test data set. The DC bike counts have a more similar distribution to the London data set than the Seoul bike counts do, which is likely the reason why the random forest model performs better for the DC data compared to the Seoul data. However, overall the distributions between the Seoul bike counts and the DC bike counts are not too different from the London data set. This implies that the effects that the covariates have on bike counts are likely different from city to city. For example, in London, more people ride bikes in the Summer than in the

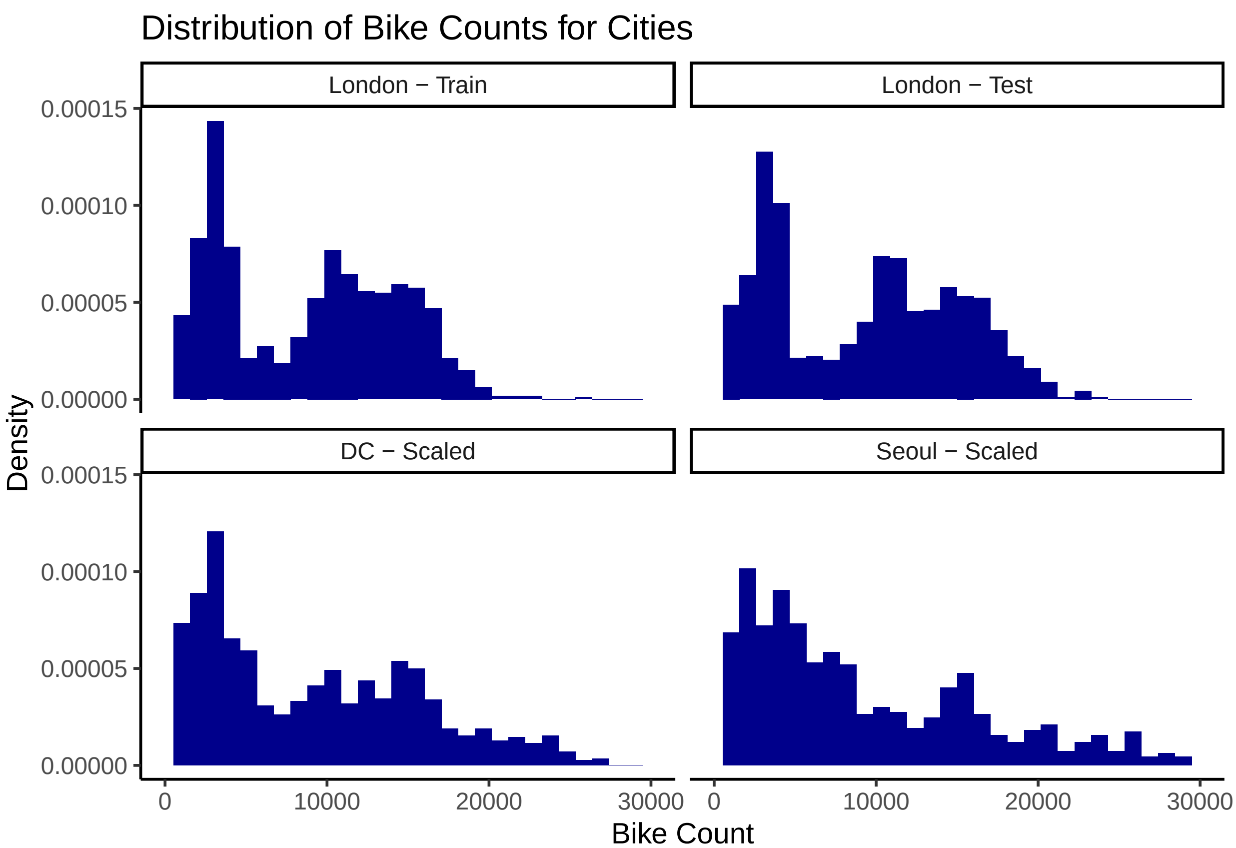


Figure 6: Density plots showing the distribution of bike count rental observations for each training and testing set.

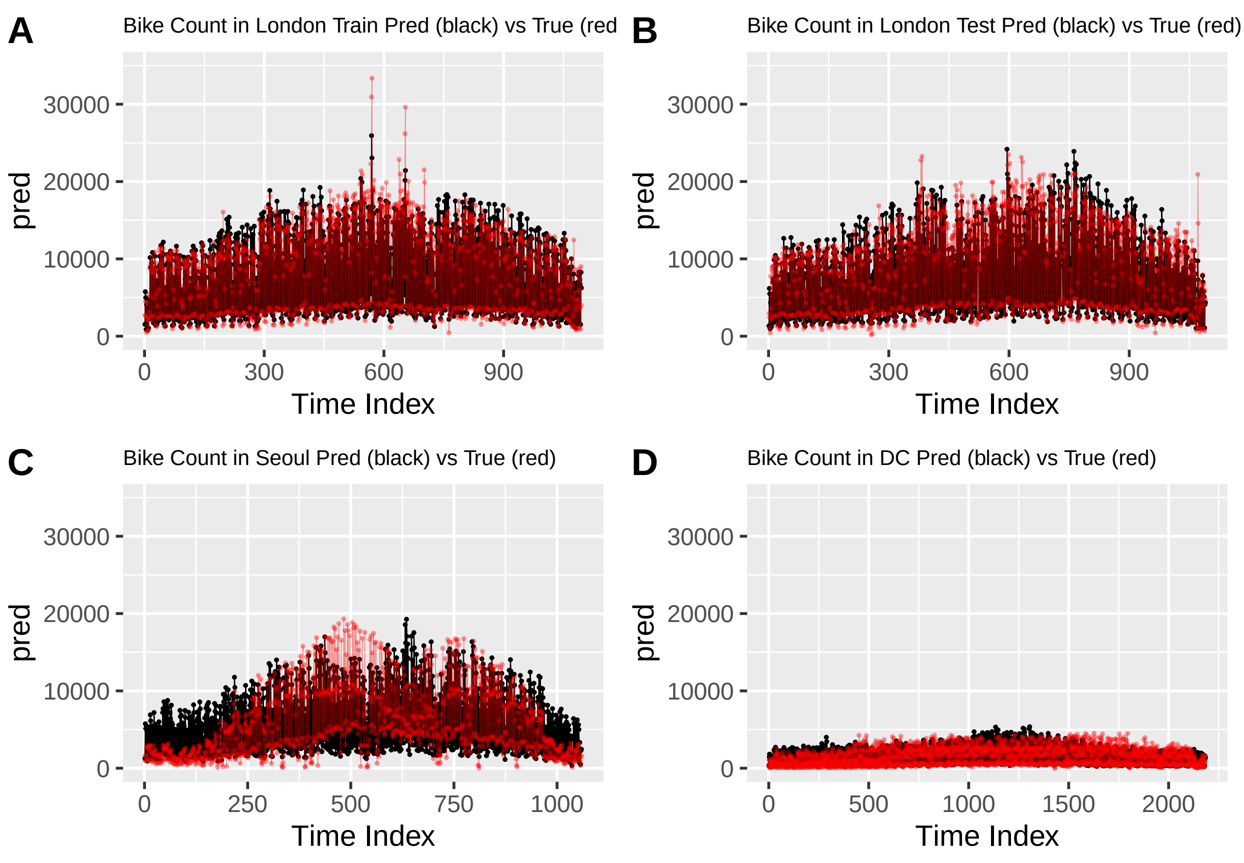
autumn, likely due to the fact that it is warmer. However it is possible that in Seoul or DC, the relationship is the opposite and more people ride bikes in Autumn compared to the Summer because it is too hot in the summer to ride a bike. Thus, since it seems likely that each covariate may influence bike counts differently in different cities, which means our model is not particularly useful in predicting bike counts in cities other than the one that it is trained on. This may be a result of the fact that our cities are very different from each other since they are all from different countries. Further work in measuring bike counts in different cities within a country needs to be done to determine how well our model performs for different cities within the same country as the city that the model was trained on.

Discussion and Conclusion

In conclusion, factors including hours, holidays, min temperature, max temperature, min humidity, max humidity, average wind speed and presence of rain/snow all have inference on the number of bike rented in London. Hours from 4pm to midnight have the largest effect on the number of bike rented. Hours from 8am to 4pm, winter season, rain/snow also have a large impact on the bike rent count. Random forest has a better prediction results in all the training and testing data, compared to the results of GLMM.

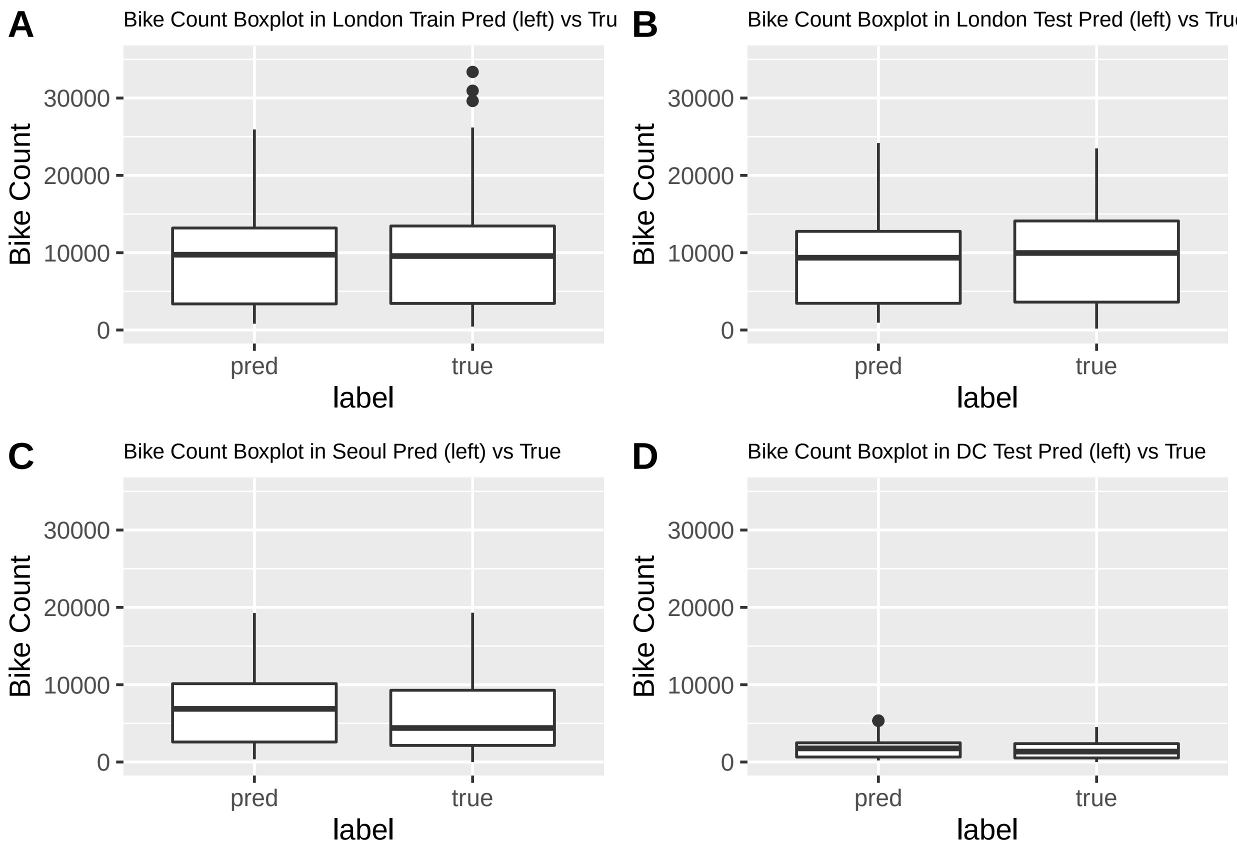
In addition to the conclusions above, we also have some findings that need more discussion. The RMSE for Seoul is much larger than that for the other two cities. The results for the non London test datasets are worse than the results for either of the two London data sets. This may imply that our model may have some limitations across cities. One possible reason is that in our model, we only considered the effects of days, hours and weather on the bike count but did not take information about cities into consideration (e.g. population, GDP), which may cause bias in prediction.

To get a better understanding on how our negative binomial GLMM model predicts the sequences of bike count in different dataset, we plotted out the predicted bike count and the ground truth for each of the four datasets. Below is the longitudinal plots of the predicted bike count (in black) and the true bike count (in red) for the three cities. As we can see, for Seoul, our models have a tendency to underestimate the predicted bike count in the earlier and later time points, however the model overestimates the bike counts in the middle time points. This is likely why the model has such a larger RMSE and a poor overall fit.



From these time series plots, we can see that our model performs well on the testing set of London except for some accidentally extremely overestimated points at the time index around 400 (around May 08th), 650 (around June 24th), and 1060 (around Dec 25th). We define the predictions whose distance to the true value is two standard deviation above the average distance to true value as the **extremely inaccurate predictions**. Most extremely inaccurate predictions on the London testing dataset happened at summer. The prediction resuls on Seoul dataset are much worse than the others. Most extremely inaccurate predictions happened at summer when the humidity is high and the temperature is high, similar to the prediction on London testing dataset. It is worth noting that the prediction on the DC dataset is different from that on the two dataset above. Most extremely inaccurate predictions are from [8,16) hour chuncks (111 out of 139). It is possibly because the bike count distribution in the training dataset of London between 8am to 4pm is much different from that in the DC dataset. Also different from Seoul and London, most extremely inaccurate predictions of DC dataset happened at Autumn (64 out of 139). But similar to Seoul and London, most extremely inaccurate predictions of DC dataset happened at weather with higher temperature.

To understand the bias and variance of our prediction results, we also drew the boxplots of predicted bike count and the true bike count of the four datasets. The plots are shown below:



From the boxplots, we can see that the predictions on Seoul dataset have noticable bias and the predictions on DC dataset also have some bias, which can help explain the poor RMSE and R2 in the two dataset.

Overall, our result is consistent with the published research by Sylwia et al. (2021) in "Impact of environment on bicycle travel demand-Assessment using bikeshare system data". In their study, they used the data from the bike sharing system in Cracow, Polan and conducted an ordinal least square regression model to analyze the effect of daily air temperature, daily rainfall, public holidays, and school holidays on the daily number of bike rented from bike sharing system. Their study result indicated that weather conditions, especially air temperature and daily rainfall, have large impact on the number of bike sharing system, which is consistent with ours result. Compared to their research, our study further found that the maximum temperature and the minimum humidity have more impact on the bike count. We did not restrict our study on daily level but cut one day into different hour chunks and found that different hour chunks in one day can also have large effect on the number of bike rented from bike sharing system.

Reference

Hothorn T, Hornik K, Zeileis A (2006). "Unbiased Recursive Partitioning: A Conditional Inference Framework." Journal of Computational and Graphical Statistics, 15(3), 651–674. doi:10.1198/106186006X133933.

Sylwia P., Mariusz K., Carmelo D (2021). "Impact of environment on bicycle travel demand—Assessment using bikeshare system data." Sustainable Cities and Society, 67, [102724]. https://doi.org/10.1016/j.scs.2021.102724

United Nations, Department of Economic and Social Affairs, Population Division (2018). World Urbanization Prospects: The 2018 Revision, Online Edition.