UTILIZING ARMA MODELS FOR NON-INDEPENDENT REPLICATIONS OF POINT PROCESSES

by

Lucas M. Fellmeth

A Thesis Submitted in

Partial Fulfillment of the

Requirements for the Degree of

Master of Science

in Mathematics

at

The University of Wisconsin–Milwaukee

May 2024

ABSTRACT

UTILIZING ARMA MODELS FOR NON-INDEPENDENT REPLICATIONS OF POINT PROCESSES

by

Lucas M. Fellmeth

The University of Wisconsin–Milwaukee, 2024
Under the Supervision of Professor Daniel Gervini

The use of a functional principal component analysis (FPCA) approach for estimating intensity functions from prior work allows us to obtain component scores of replicated point processes under the assumption of independent replications. We show these component scores can be modeled using classical autoregressive moving average (ARMA) models, thus allowing us to also apply the FPCA model to non-independent replications. The Divvy bike-sharing system in the city of Chicago is showcased as an application.

To myself, and Frank.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ACRONYMS

ACF autocorrelation function. 8, 9, 16, 23, 25

AR autoregressive. 6, 7, 8, 10

ARMA autoregressive moving average. ii, vi, 1, 2, 6, 7, 8, 9, 10, 13, 16, 20, 23, 27

FPCA functional principal component analysis. ii, 1, 2, 3, 8, 9, 13, 16, 27

MA moving average. 6, 7, 8, 10, 25

MSE mean square error. 10

PACF partial autocorrelation function. 8, 9, 16, 23, 25

1 INTRODUCTION

In recent years, bicycle-sharing systems have witnessed a surge in popularity around the globe [DeM09] [SGZ10]. Major metropolitan areas like Shanghai, China and Chicago, USA have embraced these programs, offering residents and visitors a convenient and affordable way to navigate the city. These systems typically function through short-term rentals, allowing users to access bicycles from designated stations and return them at another station within the network. However, alongside the traditional station-based systems, a newer, more flexible model has emerged: dockless bike-sharing. This dockless system grants users the freedom to park bicycles at any location within the designated service area, eliminating the need to find a specific docking station. The traditional system we are going examine in this thesis, has unattended stations distributed throughout the city. Users check out a bicycle at a station near them, travel to their intended destination and return it at a station in the vicinity. There is a good reason why we look at the traditional system. While offering a convenient transportation option, traditional station-based systems face a significant challenge: to ensure a smoothly running system, it is necessary that every station has open docks and bicycles available. This motivates the analysis of data from existing stations, to uncover hidden trends and patterns that potentially allow us to improve the availability of open docks and bicycles via the placement of further bike stations.

[Ger17] proposes a functional principal component analysis (FPCA) model to estimate intensity functions of replicated point processes under the assumption of independent replications. We use this model to obtain component scores for a dataset of the Divvy bike stations. These component scores are then viewed as time series data, allowing us to model them using a class of statistical models known as autoregressive moving average (ARMA) models. ARMA models have been extensively used for data analysis in various real world applications such as stock forecasting [AN12] [RZ16] and travel forecasting [Yan05][CT14]. In this thesis, we propose a novel use case of ARMA models. Modelling the component scores with ARMA

models allows us to apply the FPCA model to non independent replications. This thesis is structured into four main chapters.

Chapter 2 provides a foundational understanding of the FPCA model and it's estimation. Chapter 3 introduces ARMA models and details the methods used for modelling these models. Then, Chapter 4 will consist of using simulations to show our ARMA models perform as intended. Finally Chapter 5 will contain the application of FPCA to the Divvy bike-sharing data, which allows us to obtain the estimated intensity functions, and their component scores, for this data. Subsequently we shall refactor the component scores to be viewed as time series data and further transforming the data to prepare it for ARMA modelling. This chapter also details the actual modeling process using classical ARMA techniques.

2 FUNCTIONAL PRINCIPAL COMPONENT ANALYSIS MODEL

2.1 Data

Specifically, we used data of bike trips in the time frame between January 1st and December 31st of 2016. For illustration we use the single station 166, "Ashland Ave and Wrightwood Ave", which is situated on the North Side of Chicago near Lincoln Park. We shall analyze the number of daily trips for this station over the time frame mentioned above. The year 2016 was a leap year, resulting in n = 366 realizations, each with a varying number of observations. In this thesis the terms "realization" and "observation" are used in the way they are used in functional data analysis, which differs from their use in point process literature. Realization refers to a realization of the entire procedure, i.e. the entire set of bike trips on any particular day. Observation refers to a single value in a realization of the procedure, i.e. a specific bike trip on a certain day.

2.2 The Model and Estimation

As mentioned in Section 1 we will use a FPCA model introduced by [Ger17] to estimate intensity functions of data from the Divvy bike-sharing system in the city of Chicago. In our data there are n = 366 days and d = 1 bike station since we only look at the single station 166 in this thesis.Therefore we let $\mathcal{X}_i$ denote the set of checkout times of day i for the station 166. Since the bike checkout times are finite but otherwise random sets, the model uses Poisson point processes to model the $\mathcal{X}_i$. Then $\mathcal{X}_i$ can be seen as a realization of a univariate point process. The temporal point processes used here are defined as a random countable set in $[0, \infty)$. For locally finite point processes, i.e $P(\mathcal{X} \bigcap B < \infty) = 1$ where B is any bounded interval, we can define $N(B) = \#\mathcal{X} \bigcap B$ as the count function. Let $\lambda(t)$ be

a nonnegative, locally integrable function. Then $\mathcal{X}$ is a Poisson process if $N(B)$ is following a Poisson distribution with rate $\int_B \lambda(t)dt$ and conditional on $N(B) = m$, the m points in $\mathcal{X} \bigcap B$ are independent and identically distributed with density $\lambda(t)/\int_B \lambda$. λ is called the intensity function of the process.

For our data the n processes $\mathcal{X}_i$ for $i = 1, \ldots, n$ each have an intensity function λ_i on the interval $[0, 24]$. Then it is assumed that for one station,

$$\log \lambda_i(t) = \mu(t) + \sum_{k=1}^{p} u_{ik}\phi_k(t), \qquad i = 1, \ldots, n \tag{2.1}$$

where $\mu(t)$ denotes the annual mean function for the station, the ϕ_k's are the components and the u_{ik}'s are the component scores. For λ_i the Model (2.1) turns into a multiplicative model:

$$\lambda_i(t) = \lambda_0(t) \prod_{k=1}^{p} \psi_k(t)^{u_{ik}}, \qquad i = 1, \ldots, n \tag{2.2}$$

where $\lambda_0(t) = \exp \mu(t)$ and $\psi_k(t) = \exp \phi_k$. We will refer to $\lambda_0(t)$ as the baseline intensity function.

We can't directly observe the λ_i, so we need to estimate the components and the mean from the data. We do this by using spline models [dB78]:

$$\mu(t) = \sum_{l=1}^{q} c_{l0}\beta_l(t) \tag{2.3}$$

$$\phi_k(t) = \sum_{l=1}^{q} c_{lk}\beta_l(t), \tag{2.4}$$

where $\{\beta_l\}_{l=1}^{q}$ is a spline basis. We use B-splines here, because they are well suited for temporal processes but other bases can be used as well[Ger17].By modelling the $\mu(t)$'s and $\phi_k(t)$'s as splines it turns the functional estimation problem into a simpler problem of estimating the basis coefficients $c_k = (c_{1k}, \ldots, c_{qk})^T$.

This estimation problem can then be solved by maximum likelihood estimation using the Poisson model, which assumes independent replications. The addition of a roughness

penalty, to avoid producing irregular estimators of μ and the ϕ_k's, becomes necessary, since the dimension q of the spline basis β can be large. The magnitude of this roughness penalty, also referred to as the degree of smoothness for the μ and the ϕ_k's respectively is determined by tuning parameters ξ_1 and ξ_2. The corresponding penalized log-likelihood function of this problem is given by

$$Pl = \frac{1}{n} l - \xi_1 \int (\mu'')^2 - \xi_2 \sum_{k=l}^{p} \int (\phi_k'')^2 \tag{2.5}$$

Solving (2.5) enables us to find the estimators $\hat{\mu}, \hat{\phi}_k$ and $\hat{u}_{ik}$ of the mean μ, the components ϕ_k and the component scores u_{ik} respectively. Using these estimators we can estimate the daily intensity functions from Model (2.1) as

$$\hat{\lambda}_i(t) = \exp \hat{\mu}(t) + \sum_{k=1}^{p} \hat{u}_{ik} \hat{\phi}_k(t) \tag{2.6}$$

For a full explanation of the model and it's estimation refer to [Ger17]. The user needs to specify a number of tuning parameters for this model: the number of components p, the degree of the B-Splines q and the number of knots used as well as the range they need to be in, and the two smoothing parameters ξ_1 and ξ_2.

3 AUTOREGRESSIVE MOVING AVERAGE MODEL FOR TIME SERIES ANALYSIS

In the mathematical sense, a time series is a sequence of data points that is indexed in time order. These data points may have an internal structure that is not immediately visible when looking at the data. Such structures may include autocorrelation, seasonal or trend patterns. Time series analysis tries to uncover these structures in order to create a better understanding of the data. We let X_t denote the observed time series. Throughout this thesis we assume the mean of the time series to be $\mu = E[X_t] = 0$. If this is not the case, we can simply replace X_t with $X_t - \mu$ in the following equations. The autoregressive moving average models we use in this thesis are a combination of two simpler models: autoregressive (AR) and moving average (MA).

3.1 Autoregressive Model

The AR model allows us to model data, where the current value linearly depends on the previous value. The order p of the model, specifies on how many previous data values (also referred to as lags) the current value depends on. Formally, an AR(p) model is described by the linear combination [CC08]

$$X_t = \sum_{i=1}^{p} \eta_i X_{t-i} + \epsilon_t \tag{3.1}$$

$\eta_1, \ldots, \eta_p$ are the parameters of the model that determine how much influence every previous value of the data has on the current value at time t and ϵ_t is an error term for the time t. Since every value X_t depends on previous values X_{t-i}, the X_t also implicitly depend on previous error terms.

The goal when fitting an AR(p) model, is to achieve $\epsilon_t \overset{iid}{\sim} N(0, \sigma^2)$ for every t by choosing the appropriate order p. Error terms distributed like this are often referred to as "white noise".

3.2 Moving Average Model

MA models model the current data value as a linear combination of the error terms ϵ_t of previous values, which ideally are white noise. Here, the order q of the model specifies how many previous error terms the current value depends on. An MA(q) model can be described as follows [CC08]

$$X_t = \sum_{i=1}^{q} \theta_i \epsilon_{t-i} + \epsilon_t, \tag{3.2}$$

where we use plus signs in front of the sum instead of minus signs, since R uses that convention. $\theta_1, \ldots, \theta_q$ are the parameters of the model that determine how much influence every previous error has on the current value at time t and ϵ_t is once again an error term for the time t. We try to achieve $\epsilon_t \overset{iid}{\sim} N(0, \sigma^2)$, by choosing the appropriate order q.

3.3 Autoregressive Moving Average Model

The ARMA model is the combination of the two models and allows us to model data which is partly AR and partly MA with one combined model. An ARMA(p, q) model is described as [CC08]

$$X_t = \sum_{i=1}^{p} \eta_i X_{t-i} + \sum_{j=1}^{q} \theta_j \epsilon_{t-j} + \epsilon_t, \tag{3.3}$$

where $\eta_1, \ldots, \eta_p$ are the parameters of the AR part of the model, $\theta_1, \ldots, \theta_q$ are the parameters of the MA part of the model and ϵ_t is again an error term for the time t, for which we aim to achieve $\epsilon_t \overset{iid}{\sim} N(0, \sigma^2)$, by choosing p and q, the respective orders of the AR part and the MA part of the model accordingly. For all of the models described above it is important to note that the error terms ϵ_t are only of theoretical nature and are unobservable. Therefore in practice, residuals are used. A residual is an observable estimate of the unobservable statistical error.

3.4 Stationarity

In general, a ARMA(p, q) model requires a stationary time series. A stationary process $\{X_t\}$ is a stochastic process where the joint distribution of $X_{t_1}, \ldots, X_{t_n}$ is the same as the joint distribution of $X_{t_1-k}, \ldots, X_{t_n-k}$ for all choices of time points $t_1, \ldots, t_n$ and all choices of time lag k[CC08]. This results in mean and variance which are constant in time. Time series consisting of stationary data are then also referred to as stationary time series.

3.5 Modelling with ARMA Models

For our purpose, the goal is to fit the ARMA model to the component scores of the components, which we obtain through the application of the FPCA model to the Divvy bike data. Before we can do that, we need to make sure the time series of the component scores is stationary. Subsequently we need to find appropriate orders p and q to achieve $\epsilon_t \overset{iid}{\sim} N(0, \sigma^2)$ for every t. To this end we utilize ACF and PACF plots. ACF is an acronym for "autocorrelation function" and it delivers exactly that. An ACF plot is a bar chart that displays estimated autocorrelation based on the sample between a time series and its past values. ACF plots prove most useful for determining the order q of MA(q) models, since here the autocorrelation function is zero for every lag $\geq q$. AR(p) models however do not drop off, but rather slowly approach zero. This motivates using a different function, to determine the order of AR(p) models. PACF plots are needed in this case, which demonstrate the values of a "partial autocorrelation function". The PACF displays the correlation between a time series and a past value with the effect of values in between removed. Then, the partial autocorrelation at lag k is the correlation between X_t and X_{t-k}, without the effects of the values lying in between $X_{t-1}, \ldots, X_{t-(k+1)}$[CC08]. A statement about the general behavior of the ACF and PACF for ARMA models can be made [CC08]. This general behavior allows us to make deductions on which model delivers the best fit for our data.

	AR(p)	MA(q)	$ARMA(p,q), p>0$ and $q>0$
ACF	Dies off	Cuts off after lag q	Dies off
PACF	Cuts off after lag p	Dies off	Dies off

Table 1: general behaviour of the ACF and PACF for ARMA models

To test the goodness of the fit implied by the general behavior we shall again use ACF and PACF plots. However in the present context the ACF and PACF of the residuals is of interest. As described above the goal when fitting these models is for the residuals to be white noise. In other words, we require the autocorrelation of the residuals to be zero or very close to zero. In ACF and PACF plots there are two horizontal dashed lines, located at $\pm 2/\sqrt{n}$, representing plus and minus two approximated standard errors of the sample (partial) autocorrelations[CC08]. We reject the fit if the sample (partial) autocorrelations plotted in the ACF and PACF plots lie outside of these bounds for small lags k, since this is statically significant evidence of a non-zero autocorrelation in the residuals. Values lying outside the bounds for large k are not statically significant evidence, since they are likely to be estimation errors.

Before we move on to the application of the described concepts to the Divvy bike data, we revisit the purpose of this thesis. As mentioned in the introduction above the FPCA model assumes independent replications. We shall show that estimated component scores resulting from the use of this model can be modeled with ARMA models. Thus, these component scores cannot be independent, since for data which can be modeled using ARMA models every data value is correlated with previous data values and therefore not independent. This then in turn allows us to remove the assumption of independent replications and extend the FPCA model to non independent replications, thus opening up extensive new possibilities of use for this model.

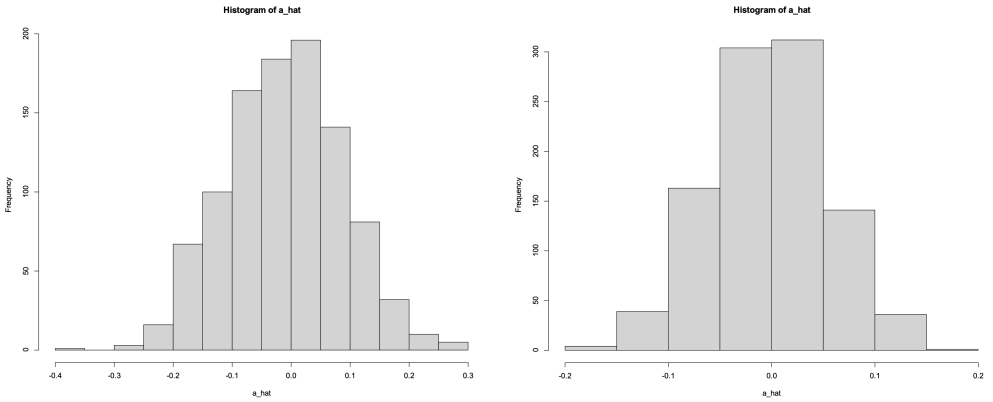
4 SIMULATIONS

This chapter is concerned with verifying that ARMA models in principle deliver the correct model, when the X_t are component scores. To confirm this we use two R programs that allow us to specify the parameter of a one component model with AR(1) or MA(1) component scores as well as the number of samples and then generates data accordingly. These simulated one component models have the components $\phi(t) = \sqrt{2}\sin 2\pi t$ and a true mean of $\mu(t) = 5\sin \pi t$. The parameter of the model with AR(1) scores is denoted by a and the parameter of the model with MA(1) scores is denoted by b. We simulate three different scenarios, a model with no correlations (i.e. $a = b = 0$), a model with relatively strong autocorrelations ($a = 0.8$) and a MA model with $b = 0.8$. Two different sample sizes $n = 100$ and $n = 300$ are considered. This gives us a total of six models. For each model we fit AR(1) or MA(1) models respectively. This allows us to compute $\hat{a}$ and $\hat{b}$, which are estimators of a and b. We will replicate each model 1000 times and find the mean square error (MSE) of $\hat{a}$ and $\hat{b}$ compared to the true values a and b respectively by averaging the errors over the replications. The MSE is defined by $MSE = \frac{1}{n}\sum_{i=1}^{n}(Y - \hat{Y}_i)^2$, where Y is the true value and the $\hat{Y}_i$ are the estimated values.

parameter	n	MSE
$a = b = 0$	100	0.0096
	300	0.0032
$a = 0.8$	100	0.0058
	300	0.0015
$b = 0.8$	100	0.0047
	300	0.0014

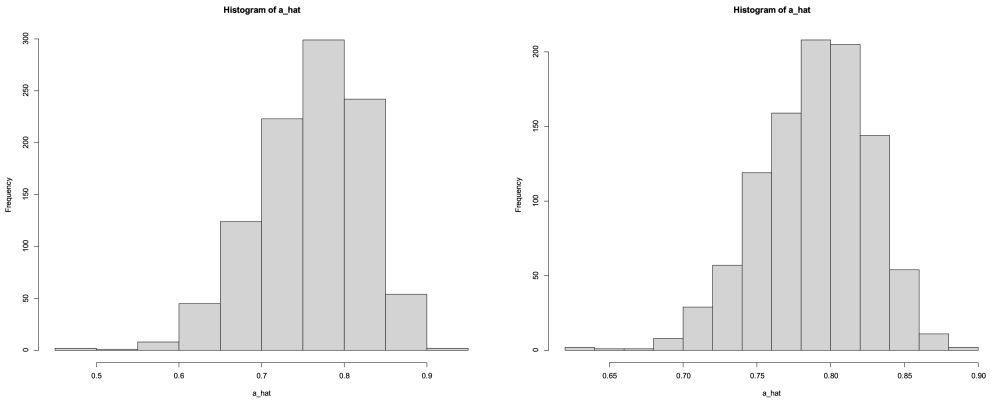
Table 2: simulation results

Table 2 showcases the results of these simulations. For each of the parameters, we can observe the MSE decreasing for increasing sample sizes n. Additionally, when looking at histograms of the estimators $\hat{a}$ and $\hat{b}$ in Fig. (1), Fig. (2) and Fig. (3) we notice they are approximately normal for all parameters and sample sizes. This is as expected because the



(a) $n = 100$ (b) $n = 300$

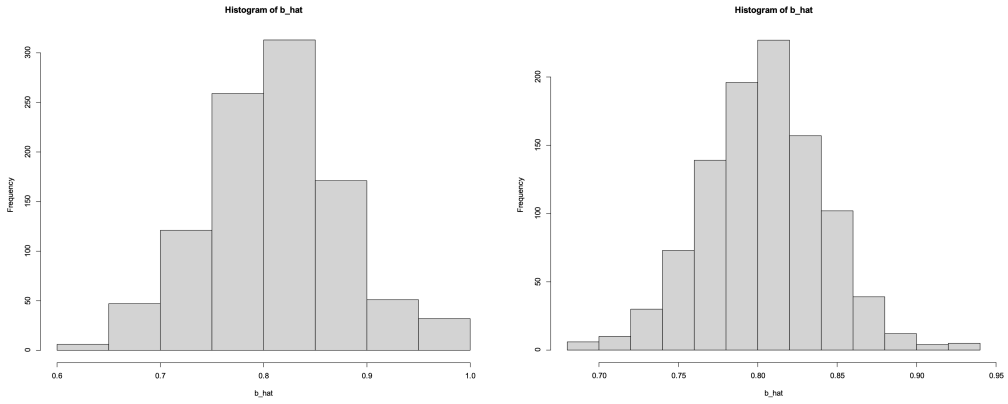
Figure 1: Histograms of $\hat{a}$ for $a = 0$ and different sample sizes



(a) $n = 100$ (b) $n = 300$

Figure 2: Histograms of $\hat{a}$ for $a = 0.8$ and different sample sizes

R function "arima()" used for fitting the models estimates the components by maximum likelihood. Maximum likelihood estimators are usually asymptotically normally distributed.



(a) $n = 100$ (b) $n = 300$

Figure 3: Histograms of $\hat{b}$ for $b = 0.8$ and different sample sizes

5 APPLYING FPCA AND ARMA TO DATA OF THE DIVVY BIKE SHARING SYSTEM

The previous chapters equipped us with a FPCA model to estimate intensity functions of replicated point processes, which allows us to obtain component scores and with ARMA models which we can use to model these component scores when viewed as a time series. Additionally we confirmed the validity of the methods we will use in this chapter through simulations. This chapter unites these forces! We'll apply both models to the Divvy dataset. The data is publicly available on the website of the Chicago data portal (https://data.cityofchicago.org).

5.1 FPCA Model

We fitted the model for the Ashland & Wrightwood station during the period between January 1st and December 31st of 2016. There were 2 components used, since they allowed us to sufficiently capture information. We used cubic splines with 10 knots in the range of (0,24) representing the hours of the day. The smoothing parameters were chosen subjectively, with visual appeal in mind, instead of extensive cross-validation since the focus of this thesis lies on time series modeling of the principal component scores.

When looking at the baseline intensity function in Fig. (4) we notice three peaks: (1) a morning peak around 7:30 a.m., which is the largest (2) a midday peak around 1 p.m., which is the smallest and (3) an evening peak around 5:45 p.m..

To interpret the components, the estimated baseline function $\hat{\lambda}_0$ is plotted alongside $\hat{\lambda}_0^+ = \hat{\lambda}_0\hat{\psi}_k^c$ and $\hat{\lambda}_0^- = \hat{\lambda}_0\hat{\psi}_k^{-c}$ where c is a constant. Here, twice the standard variation of the corresponding $\hat{u}_{ik}$'s is used, since that allows for convenient visualization[GK18]. We can recognize certain patterns when looking at the baseline in conjunction with the baseline multiplied by a positive and negative exponent of the component in Fig. (5). When examining the first component in Fig. (5a), we can detect a seasonal trend pattern. For

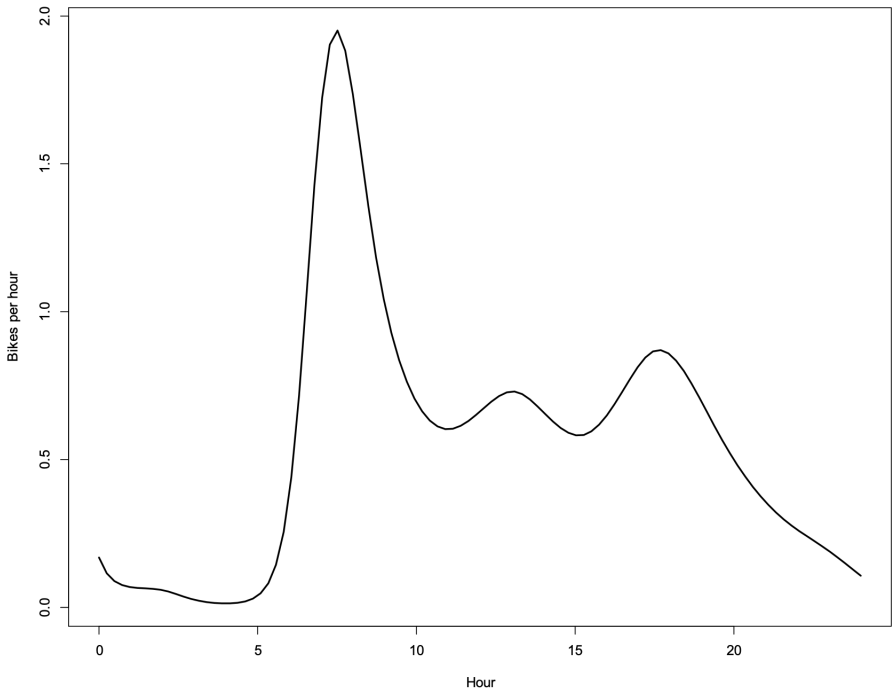
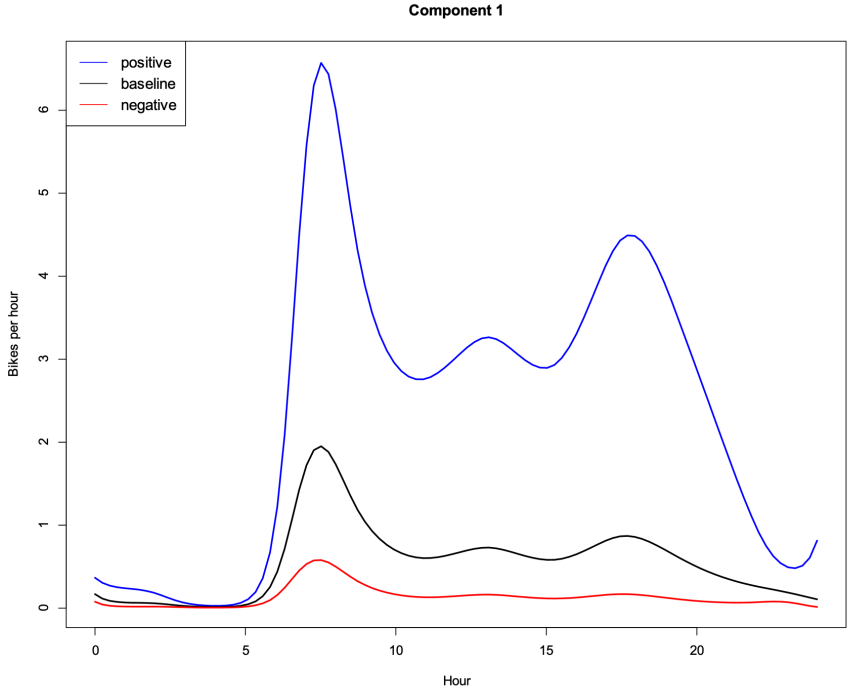
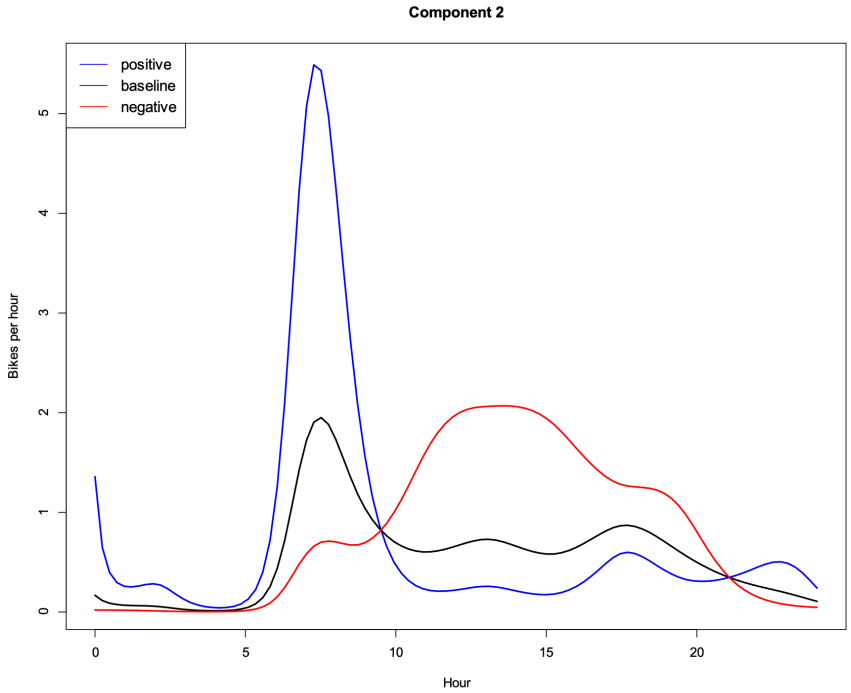


Figure 4: Baseline intensity function of daily bike demand for the Divvy bike station 166 located at the intersection of Ashland Ave and Wrightwood Ave

the negative score, the graph is lower than the baseline intensity overall suggesting fall and winter usage. For the positive score, the graph is higher than the baseline intensity overall with peaks in the morning and evening, suggesting spring and summer usage when people stay in the shade during the midday sun. For the second component in Fig. (5b) we observe a day-of-the-week effect: for the negative score, we observe a flatter morning peak and a higher afternoon peak compared to the baseline intensity, suggesting weekend/holiday use. The positive score shows a higher morning peak and a lower afternoon peak, suggesting weekday use.



(a) Component 1



(b) Component 2

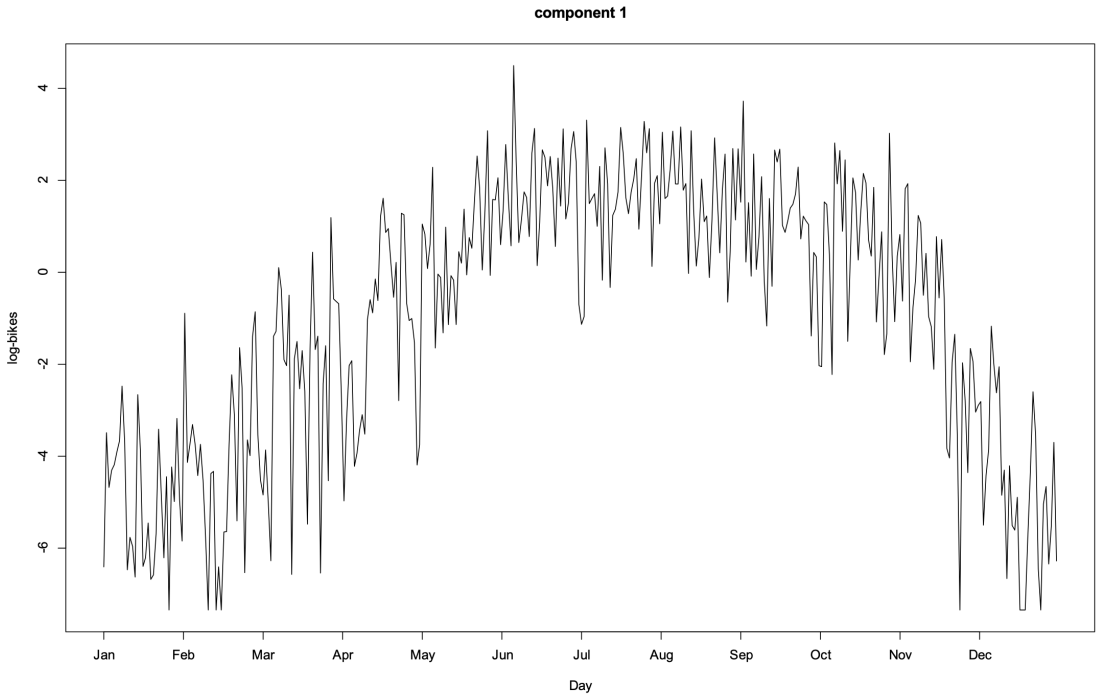
Figure 5: Multiplicative components of daily bike demands for Divvy Station 166: baseline intensity function and baseline multiplied by a positive and negative exponent of the component

5.2 ARMA Model

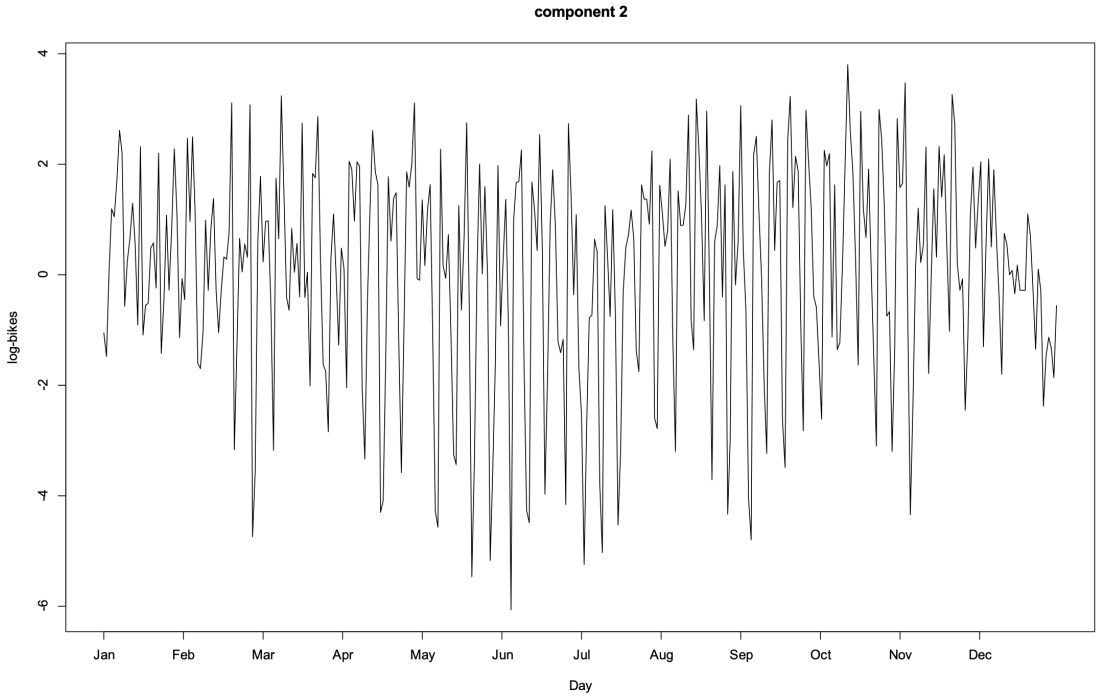
In the previous section, we successfully utilized FPCA to analyze data from the Divvy bike-sharing system in Chicago. This analysis yielded estimated intensity functions, which represent the underlying spatial distribution of bike usage across the city. Additionally, we obtained component scores. However, the component scores obtained from FPCA are not directly suitable for modeling using traditional time series techniques like ARMA models. This chapter addresses this challenge by transforming the component scores into a format that aligns with the requirements of ARMA modeling. Following the transformation, we will detail the application of ARMA models to the prepared data. In Chapter 2 we discovered different patterns for the two components. For component 1 we observed a seasonal pattern and component 2 showed signs of a day-of-the-week effect. Thus both of our components have some sort of patterns that influence their values.

When looking at the time series of the two components in Fig. (6), a clear seasonal pattern is visible for component 1, in Fig. (6a). For component 2 in Fig. (6b) we cannot identify a clear pattern by only looking at the time series plots.

The respective ACF and PACF in Figures (7) and (8) show a clear picture for both components. For component 1, the autocorrelation and the partial autocorrelation show an exponential decay, indicating some sort of trend. The autocorrelation as well as the partial autocorrelation of component 2 show clear signs of a day-of-the-week effect, since there are always four or five negative values representing weekdays followed by three or two positive values representing weekends/holidays. Thus, the observations from the time series plots and the respective autocorrelation and partial autocorrelation plots for both components correspond with what we would expect from the intensity functions in the previous chapter, thus confirming our observations. This motivates splitting component 2 into weekdays and weekends/holidays.

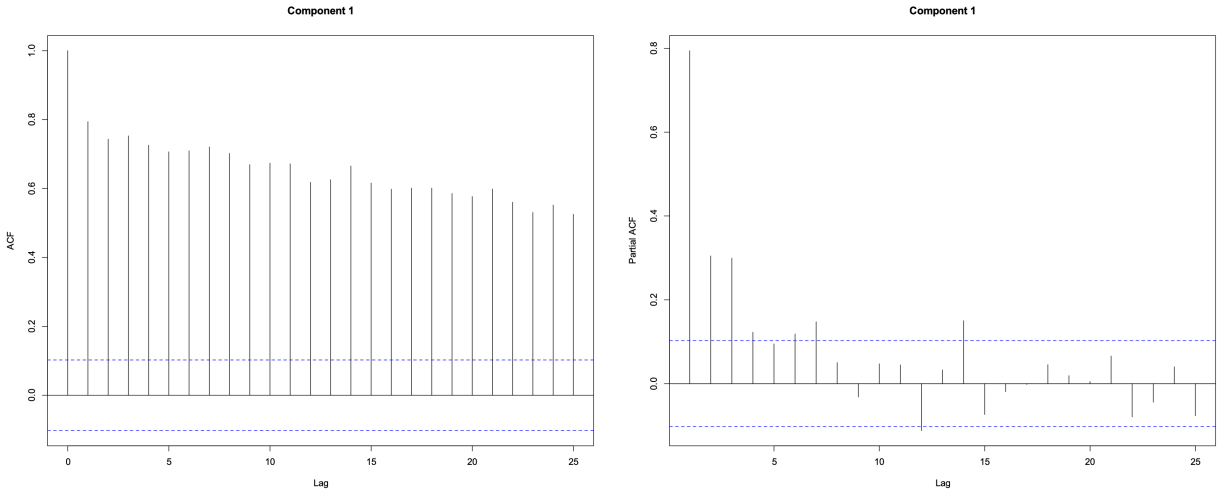


(a) Component 1



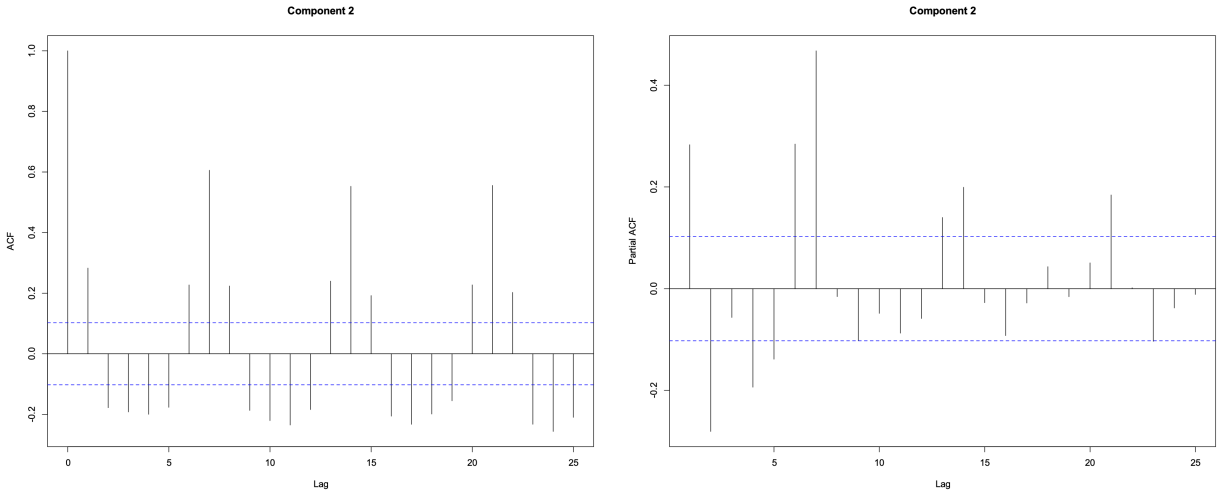
(b) Component 2

Figure 6: Time series of the components for Divvy bike station 166



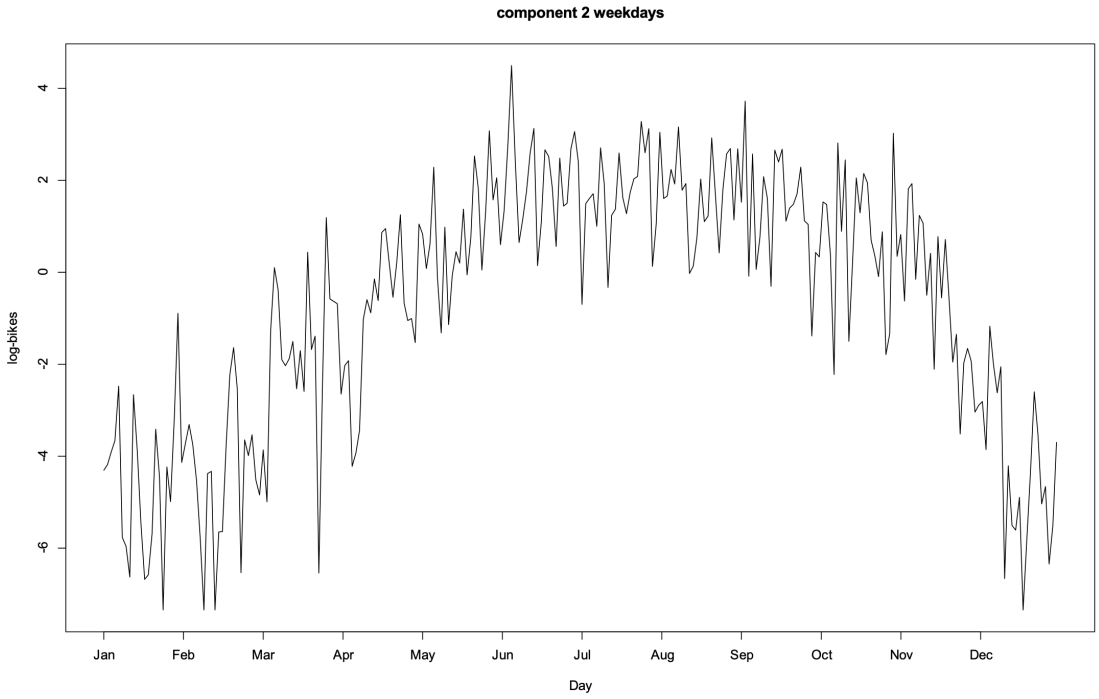
(a) ACF (b) PACF

Figure 7: ACF and PACF of the time series for the first component

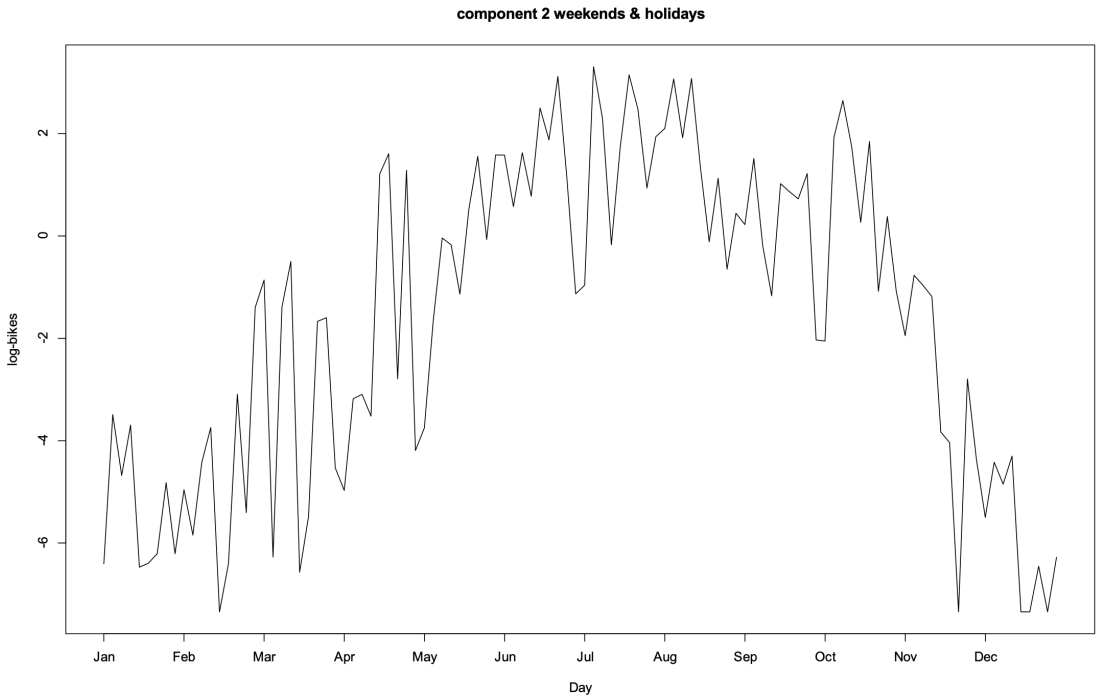


(a) ACF (b) PACF

Figure 8: ACF and PACF of the time series for the first component



(a) Weekdays



(b) Weekends and Holidays

Figure 9: Separate time series of weekdays and weekends and holidays for component 2

5.2.1 Dealing with Trends

The implementation of the proposed split uncovers an underlying seasonal pattern for weekdays and weekends respectively. Seasonal patterns like the ones prevalent in the three time series plots in Fig. (6), Fig. (9a) and Fig. (9b) can make modeling the data very difficult. The change over time induced by the trend makes it hard to identify patterns or a trends in the data itself, that are not caused by seasonality since seasonality overshadows those patterns. This motivates requiring stationarity, aside from the requirement of the ARMA model. In consequence, inference based upon non-stationary time series can produce unreliable results and forecasts. Whereas working with stationary time series allows us to make reliable statements about patterns such as autocorrelation or moving averages.

When inspecting the three time series plots in Fig. (6), Fig. (9a) and Fig. (9b) we clearly find that none of them are stationary time series, since they vastly change over time. Examining the component scores, we might observe a recurring seasonal pattern that resembles a quadratic function. There are different methods to refactor time series with such trend into a stationary format, which is suitable for ARMA modeling. We are going to use differencing in this case, but estimation and subsequent removal of the underlying quadratic function could also be used. Differencing involves calculating the difference between subsequent data points in the time series. There are various differencing orders that each tend to different trends in the data. For our purpose, we will use second-order differencing.

First-order differencing represents the difference between consecutive data points, and, in our example, satifies the equation

$$Z_i^{FO} = X_i - X_{i-1}, \qquad i = 2, \ldots, n$$

This can prove useful for linear trends, but for the quadratic patterns in our data, second-order differencing is necessary to reach stationarity. Second-order differencing represents the

difference between consecutive first-order differences. Thus in our example we get:

$$\begin{aligned} Z_i &= Z_i^{FO} - Z_{i-1}^{FO} \\ &= (X_i - X_{i-1}) - (X_{i-1} - X_{i-2}) \\ &= X_i - 2X_{i-1} + X_{i-2} \qquad\qquad i = 3, \ldots, n \end{aligned}$$

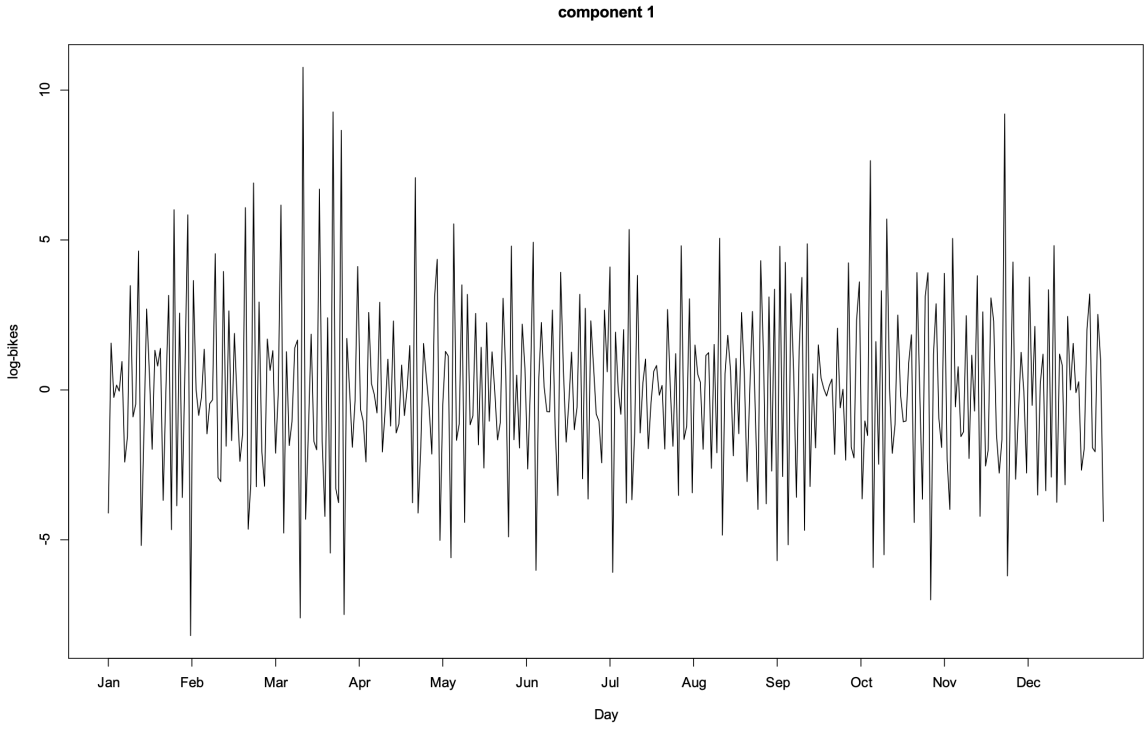
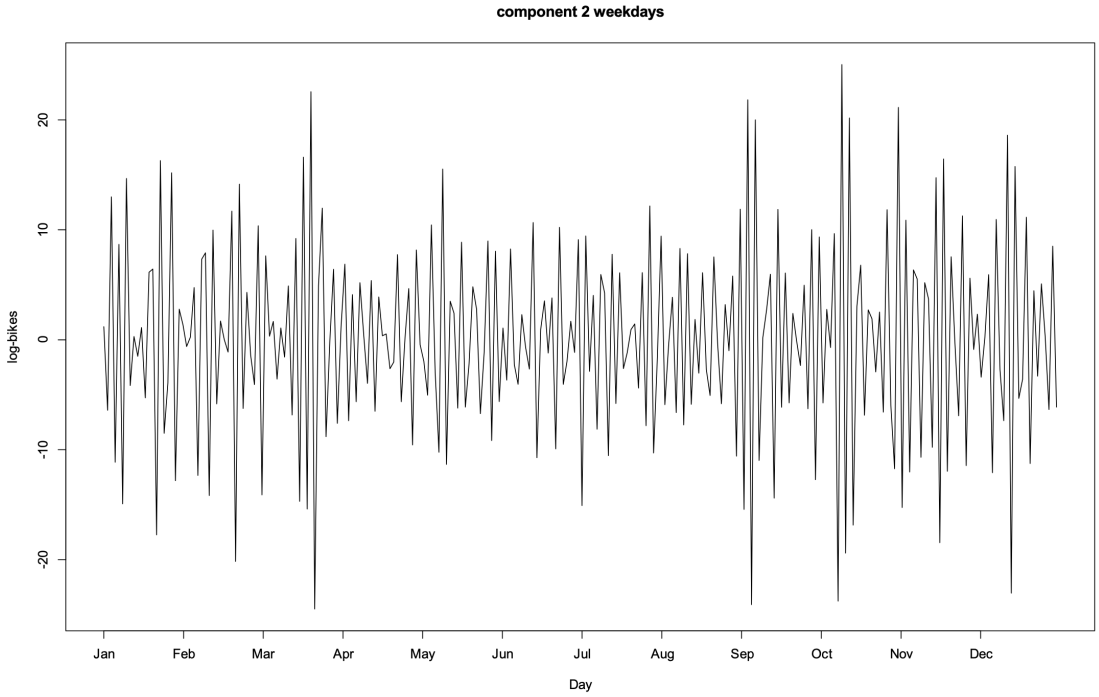
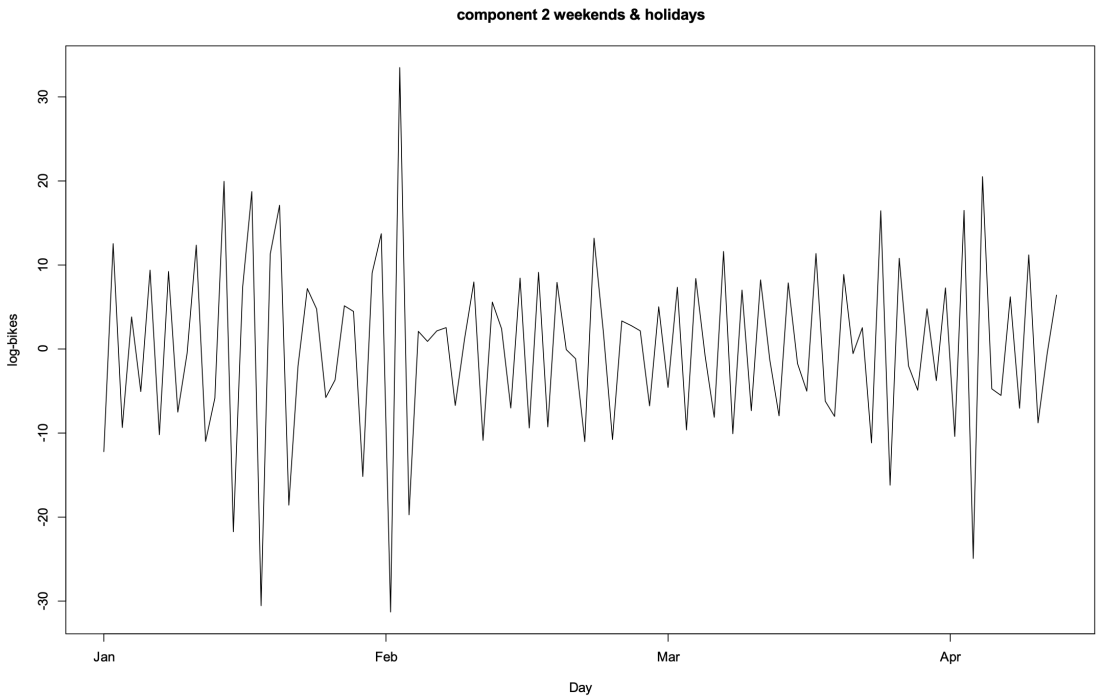


Figure 10: Stationary time series of component 1



(a) Weekdays

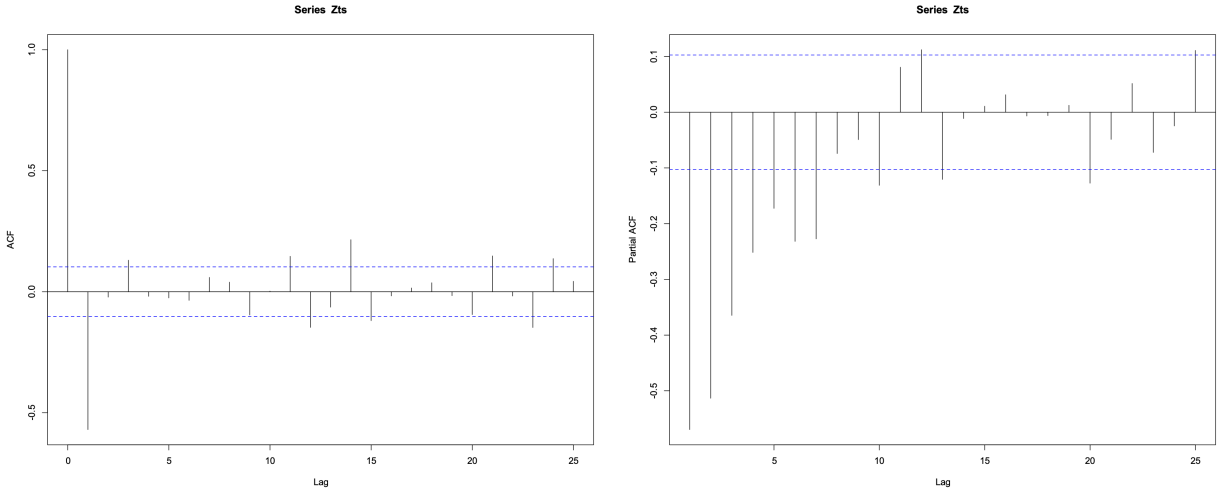


(b) Weekends and Holidays

Figure 11: Separate stationary time series of weekdays and weekends and holidays for component 2

5.2.2 Fitting the ARMA Model to the Divvy Data

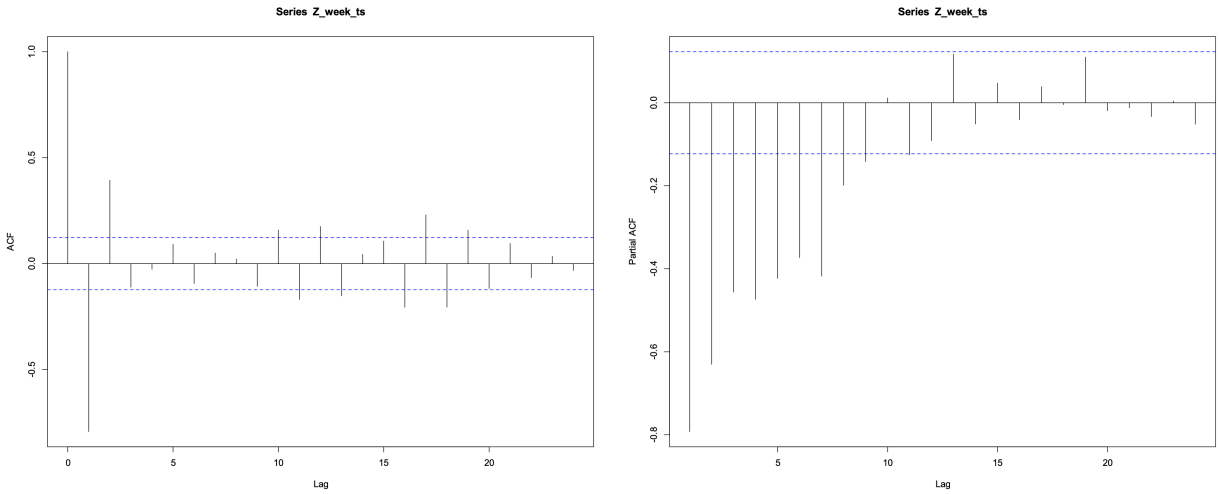
After this transformation, all three of our time series plots in Fig. (11), Fig. (11a) and Fig.(11b) indicate that second-error differencing successfully transformed our time series with trends into stationary time series. Thus, we now fulfill the requirements of ARMA modelling and can commence with fitting an appropriate ARMA(p, q) model to our data. For this purpose, we commence by investigating the ACF and PACF plots of our time series.



(a) ACF (b) PACF

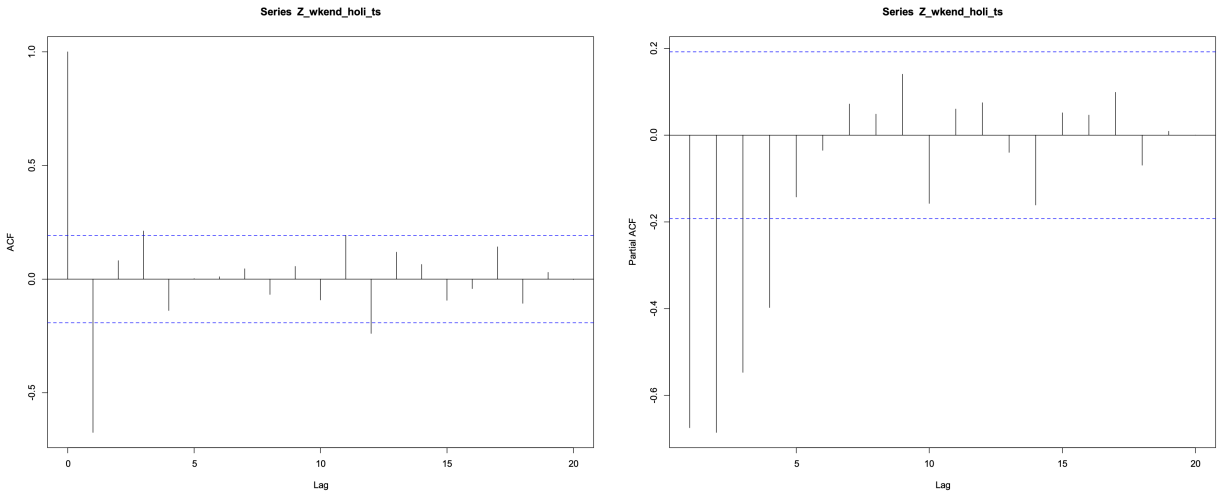
Figure 12: ACF and PACF of the stationary time series for the first component

We shall use the statements made about the general behaviour of ACF and PACF for ARMA models in Table 1 from an earlier chapter of this thesis to fit ARMA models to the components scores of the Divvy bike data. For the first component the ACF in Fig. (12a) cuts off after lag 1 and the PACF in Fig. (12b) dies off after lag 1 as well. This clearly indicates a MA(1) model. For the weekdays of the second component the ACF in Fig. (13a) cuts off after lag 2 and the PACF in Fig. (13b) dies off after lag 1. This indicates either a MA(1) or a MA(2) model. We'll consult the residuals to determine which model will have a better fit. For the weekends and holidays of the second component the ACF in Fig. (14a) cuts off after lag 1 and the PACF in Fig. (14b) dies off after lag 2. This again indicates either a MA(1) or a MA(2) model. We will also use the residuals to determine the goodness



(a) ACF (b) PACF

Figure 13: ACF and PACF of the stationary time series for the weekdays of the second component



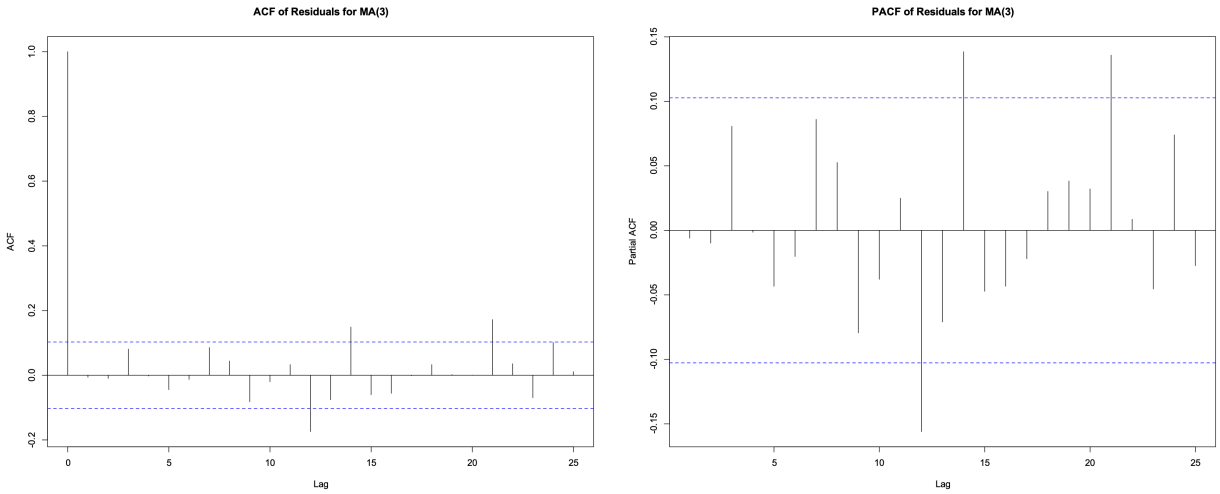
(a) ACF (b) PACF

Figure 14: ACF and PACF of the stationary time series for the weekends and holidays of the second component

of each fit in this case.

5.2.3 Goodness of the Fit

To ensure our proposed fit is correct we shall investigate the ACF and PACF of the residuals for each of the components. We'll begin by looking at the first component. The smallest order q for which the ACF in Fig. (15a) and the PACF in Fig. (15b) for the residuals indicate a good fit is $q = 3$. Only for MA(3) the residuals look like white noise. For the MA(1) and MA(2) models the residuals do not seem to be white noise.

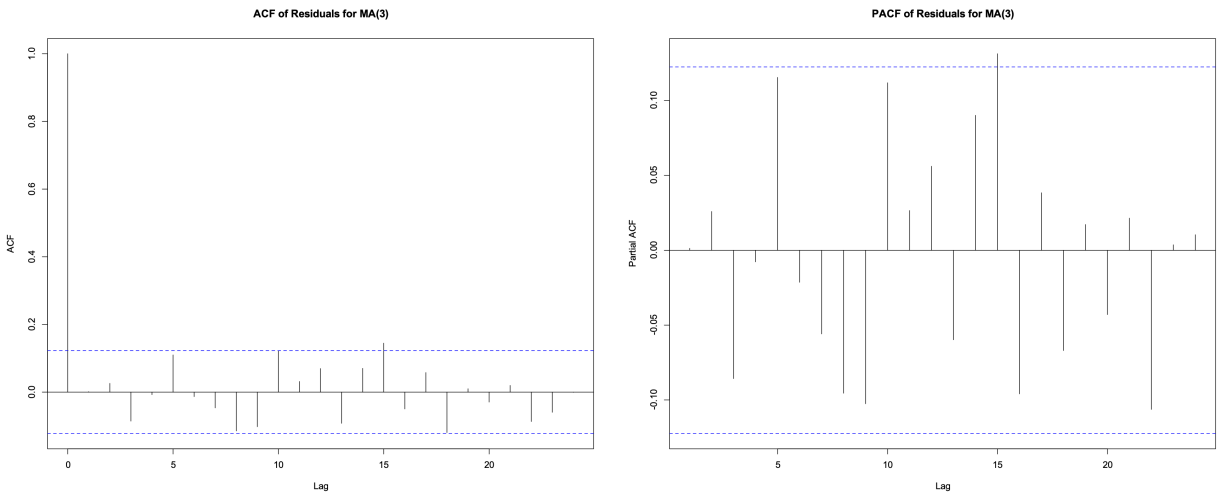


(a) ACF (b) PACF

Figure 15: ACF and PACF of the residuals of a MA(3) model for the first component

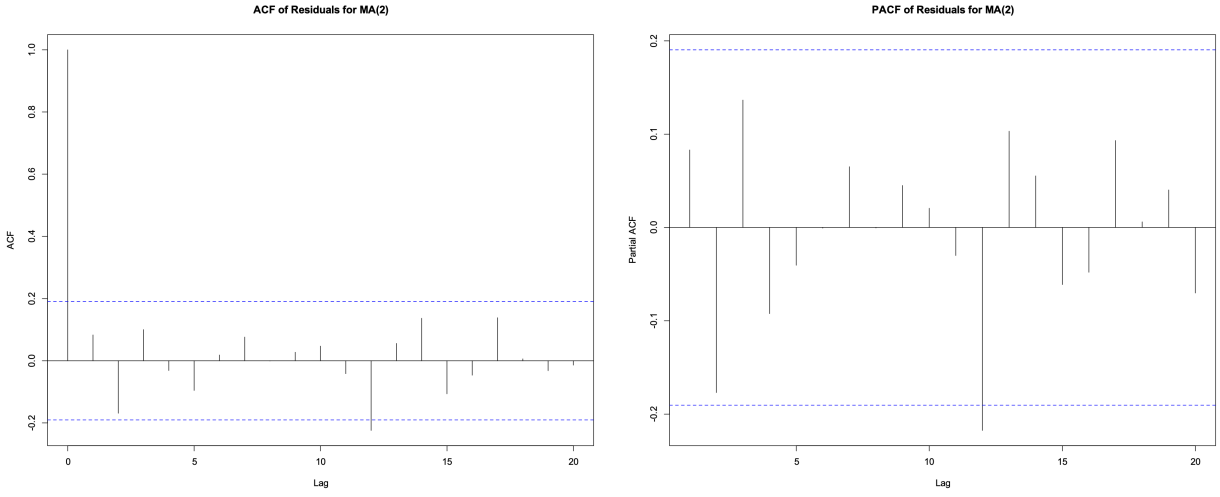
For the weekdays of the second component we observe a similar picture. The smallest order q for which the ACF in Fig. (16a) and the PACF in Fig. (16b) for the residuals indicate a good fit is again $q = 3$. Only for MA(3) the residuals look like white noise. For the MA(1) and MA(2) models the residuals do not seem to be white noise.

For the weekends and holidays of the second component, the smallest order q for which the ACF in Fig. (17a) and the PACF in Fig. (17b) for the residuals indicate a good fit is $q = 2$. For $q = 1$ the residuals do not seem to be white noise.



(a) ACF (b) PACF

Figure 16: ACF and PACF of the residuals of a MA(3) model for the weekdays of the second component



(a) ACF (b) PACF

Figure 17: ACF and PACF of the residuals of a MA(2) model for the weekends and holidays of the second component

6 CONCLUSION

This thesis is based on prior work on FPCA models. [Ger17] introduced a FPCA model which can be used to estimate the intensity functions of replicated point processes. However the model assumes independent replications, because of the log-likelihood function used to find the estimates of intensity functions. The purpose of this thesis was expanding the possibility of use for this model to non-independent replications. To achieve this goal we applied the FPCA model to data from the Divvy bike-sharing system in the city of Chicago. This enabled us to find component scores for this data. We found that the component scores of every component can be modeled using ARMA models. This allowed us to also apply the FPCA model to non-independent replications, thus achieving our goal. It would be interesting to see if other statistical models assuming independence in the data could be expanded to non-independent data by using a similar method.

REFERENCES

[AN12] Mohamad Foad Anaghi and Yaser Norouzi. A model for stock price forecasting based on ARMA systems. In *2012 2nd International Conference on Advances in Computational Tools for Engineering Applications (ACTEA)*, pages 265–268. IEEE, 2012.

[CC08] J.D. Cryer and K.S. Chan. *Time Series Analysis: With Applications in R*. Springer Texts in Statistics. Springer New York, 2008.

[CT14] Oscar Claveria and Salvador Torra. Forecasting tourism demand to Catalonia: Neural networks vs. time series models. *Economic Modelling*, 36:220–228, 2014.

[dB78] Carl de Boor. *A Practical Guide to Spline*, volume Volume 27. 01 1978.

[DeM09] Paul DeMaio. Bike-sharing: History, impacts, models of provision, and future. *Journal of public transportation*, 12(4):41–56, 2009.

[Ger17] Daniel Gervini. Multiplicative component models for replicated point processes. *arXiv preprint arXiv:1705.09693*, 2017.

[GK18] Daniel Gervini and Manoj Khanal. Exploring patterns of demand in bike sharing systems via replicated point process models, 2018.

[RZ16] Mohammad Mahdi Rounaghi and Farzaneh Nassir Zadeh. Investigation of market efficiency and financial stability between s&p 500 and london stock exchange: monthly and yearly forecasting of time series stock returns using arma model. *Physica A: Statistical Mechanics and its Applications*, 456:10–21, 2016.

[SGZ10] Susan A Shaheen, Stacey Guzman, and Hua Zhang. Bikesharing in Europe, the Americas, and Asia: past, present, and future. *Transportation research record*, 2143(1):159–167, 2010.

[Yan05] Jiann-Shiou Yang. A study of travel time modeling via time series analysis. In *Proceedings of 2005 IEEE Conference on Control Applications, 2005. CCA 2005.*, pages 855–860, 2005.

APPENDIX

A R code

Main code:

```
### FPCA approach ------------------------------------------------------------------

rm(list = ls())
load("divvy_data_2016_AW.Rdata")
source("mcatpp.R")
library(MASS)
library(splines2)

#variables
p <- 2
rng <- c(0, 24)
nkt <- 10
itmax <- 50
sm1 <- 10^-5
sm2 <- 10^-5

out <- mcatpp(x = x, p = p, rng = rng, nkt = nkt, sm1 = sm1, sm2 = sm2, itmax = itmax)

c0 <- unlist(out$c0, recursive = F) #c0: Basis coefficients of the mean (q x 1).
C <- unlist(out$C, recursive = F)   #C: Basis coefficients of the components (q x p).
s2 <- unlist(out$s2, recursive = F) #s2: Component variances (p vector).
u <- unlist(out$u, recursive = F)   #u: Individual component scores (n x p).

 t <- seq(from = rng[1], to = rng[2], length.out = 100)
knots <- seq(from = rng[1], to = rng[2], length.out = nkt + 2)[-c(1, nkt + 2)]

B <- bSpline(x = t, knots = knots, intercept = T)

plot(t, exp(B %*% c0), type = "l", lwd = 2, col = "black", xlab = "Hour",
ylab = "Bikes per hour", xlim = c(0, 24))
for (i in 1:p) {

  lmb0 <- exp(B %*% c0)
  lmbpos <- exp(B %*% c0 + 2 * sqrt(s2[i]) * B %*% C[, i])
  lmbneg <- exp(B %*% c0 - 2 * sqrt(s2[i]) * B %*% C[, i])


  plot(t, lmbpos, type = "l", lwd = 2, col = "blue", main = paste("Component", i),
  xlab = "Hour", ylab = "Bikes per hour")


  lines(t, lmb0, lwd = 2, col = "black")
  lines(t, lmbneg, lwd = 2, col = "red")
  legend("topleft", legend = c("positive", "baseline", "negative"),
         col = c("blue", "black", "red"), lty = 1, cex = 1.2)
}
```

```r
### time series analysis with ARMA models ------------------------------------------

uts <- ts(u)
acf(uts[, 1], main = "Component 1")
pacf(uts[, 1], main = "Component 1")
acf(uts[, 2], main = "Component 2")
pacf(uts[, 2], main = "Component 2")

for (i in 1:p) {
  plot.ts(uts[, i], main = paste("component", i), xlab="Day", ylab = "log-bikes",
  xaxt = "n",)
  axis(1,seq(1,nrow(u),nrow(u)/12),c("Jan","Feb","Mar",'Apr',"May", "Jun", "Jul",
  "Aug","Sep","Oct","Nov","Dec"))
}

## removing seasonal trend from Component 1 to make it stationary

Z <- vector("numeric", 364)

for (i in 3:366) {

  Z[i - 2] <- u[i, 1] - 2 * u[i - 1, 1] + u[i - 2, 1]

}
ARi <- list()
Zts <- ts(Z)
acf(Zts)
pacf(Zts)
AR <- arima(Zts, order = c(1, 0, 0))
ARi[[1]] <- AR

plot.ts(Zts, main = paste("component 1"), xlab="Day", ylab = "log-bikes",xaxt = "n",)
axis(1,seq(1,nrow(u),nrow(u)/12),c("Jan","Feb","Mar",'Apr',"May", "Jun", "Jul","Aug",
"Sep","Oct","Nov","Dec"))

## splitting Component 2 in weekdays & weekends/holidays to deal
# with the day-of-the-week effect

# Week

Z_week <- u[i_week]
Z_wkend_holi <- u[i_wkend_holi]

Z_week_ts <- ts(Z_week)
acf(Z_week_ts)
pacf(Z_week_ts)
plot.ts(Z_week_ts, main = paste("component 2 weekdays"), xlab="Day",
ylab = "log-bikes",xaxt = "n",)
    axis(1,seq(1, length(Z_week_ts), length(Z_week_ts)/12),c("Jan","Feb","Mar",'Apr',
    "May", "Jun", "Jul","Aug","Sep","Oct","Nov","Dec"))

# remove quadratic trend from the weekdays of Component 2 to make the TS stationary

temp <- vector("numeric", length(Z_week) - 2)
```

```
for (i in 3:length(Z_week)) {

  temp[i - 2] <- Z_week[i] - 2 * Z_week[i - 1] + Z_week[i - 2]

}
Z_week <- temp
Z_week_ts <- ts(Z_week)
acf(Z_week_ts)
pacf(Z_week_ts)
AR <- arima(Z_week_ts, order = c(1, 0, 0))
ARi[[2]] <- AR
plot.ts(Z_week_ts, main = paste("component 2 weekdays"), xlab="Day",
ylab = "log-bikes",xaxt = "n",)
  axis(1,seq(1, length(Z_week_ts), length(Z_week_ts)/12),c("Jan","Feb","Mar",'Apr',
  "May", "Jun", "Jul","Aug","Sep","Oct","Nov","Dec"))

# Weekend

Z_wkend_holi_ts <- ts(Z_wkend_holi)
acf(Z_wkend_holi_ts)
pacf(Z_wkend_holi_ts)
plot.ts(Z_wkend_holi_ts, main = paste("component 2 weekends & holidays"), xlab="Day",
ylab = "log-bikes",
xaxt = "n",)
    axis(1,seq(1, length(Z_wkend_holi_ts), length(Z_wkend_holi_ts)/12),c("Jan","Feb",
    "Mar",'Apr',"May", "Jun","Jul","Aug","Sep","Oct","Nov","Dec"))

# remove quadratic trend from the weekends/ holidays
#of Component 2 to make the TS stationary

rm(temp)
temp <- vector("numeric", length(Z_wkend_holi) - 2)

for (i in 3:length(Z_wkend_holi)) {

  temp[i - 2] <- Z_wkend_holi[i] - 2 * Z_wkend_holi[i - 1] + Z_wkend_holi[i - 2]

}
Z_wkend_holi <- temp
Z_wkend_holi_ts <- ts(Z_wkend_holi)
acf(Z_wkend_holi_ts)
pacf(Z_wkend_holi_ts)
AR <- arima(Z_wkend_holi_ts, order = c(1, 0, 0))
ARi[[3]] <- AR
plot.ts(Z_wkend_holi_ts, main = paste("component 2 weekends & holidays"), xlab="Day",
ylab = "log-bikes",
xaxt = "n",)
  axis(1,seq(1,nrow(u),nrow(u)/12),c("Jan","Feb","Mar",'Apr',"May", "Jun", "Jul",
  "Aug","Sep","Oct","Nov","Dec"))

#Simulations-------------------------------------------------------------------

source(file = "simul.R")
```

```
r <- 1000
a <- 0
a_hat <- replicate(n = r, {
  sim <- gen1(n = 100, a = a)
  comp_scores_sim_ts <- ts(sim\$w)
  ARsim <- arima(comp_scores_sim_ts, order = c(1, 0, 0))
  coef(ARsim)[1]
})
mean((a_hat - a)^2)
hist(a_hat)

a_hat <- replicate(n = r, {
  sim <- gen1(n = 300, a = a)
  comp_scores_sim_ts <- ts(sim\$w)
  ARsim <- arima(comp_scores_sim_ts, order = c(1, 0, 0))
  coef(ARsim)[1]
})
mean((a_hat - a)^2)
hist(a_hat)

a <- 0.8
a_hat <- replicate(n = r, {
  sim <- gen1(n = 100, a = a)
  comp_scores_sim_ts <- ts(sim\$w)
  ARsim <- arima(comp_scores_sim_ts, order = c(1, 0, 0))
  coef(ARsim)[1]
})
mean((a_hat - a)^2)
hist(a_hat)

a <- 0.8
a_hat <- replicate(n = r, {
  sim <- gen1(n = 300, a = a)
  comp_scores_sim_ts <- ts(sim\$w)
  ARsim <- arima(comp_scores_sim_ts, order = c(1, 0, 0))
  coef(ARsim)[1]
})
mean((a_hat - a)^2)
hist(a_hat)

b <- 0.8
b_hat <- replicate(n = r, {
  sim <- gen2(n = 100, b = b)
  comp_scores_sim_ts <- ts(sim\$w)
  MAsim <- arima(comp_scores_sim_ts, order = c(0, 0, 1))
  coef(MAsim)[1]
})
mean((b_hat - b)^2)
hist(b_hat)

b_hat <- replicate(n = r, {
  sim <- gen2(n = 300, b = b)
  comp_scores_sim_ts <- ts(sim\$w)
  MAsim <- arima(comp_scores_sim_ts, order = c(0, 0, 1))
```

```
  coef(MAsim)[1]
})
mse <- mean((b_hat - b)^2)
hist(b_hat)


#ACF and PACF for the residuals of different order MA models--------------------------

MAi <- list()
MA <- arima(Zts, order = c(0, 0, 1))
MAi[[1]] <- MA

acf(resid(MAi[[1]]), main = "Component 1 - ACF of Residuals for MA(1)")
pacf(resid(MAi[[1]]), main = "Component 1 - PACF of Residuals for MA(1)")

MA <- arima(Zts, order = c(0, 0, 2))
MAi[[1]] <- MA

acf(resid(MAi[[1]]), main = "Component 1 - ACF of Residuals for MA(2)")
pacf(resid(MAi[[1]]), main = "Component 1 - PACF of Residuals for MA(2)")

MA <- arima(Zts, order = c(0, 0, 3))
MAi[[1]] <- MA

acf(resid(MAi[[1]]), main = "Component 1 - ACF of Residuals for MA(3)")
pacf(resid(MAi[[1]]), main = "Component 1 - PACF of Residuals for MA(3)")

MA <- arima(Z_week_ts, order = c(0, 0, 1))
MAi[[2]] <- MA

acf(resid(MAi[[2]]), main = "Component 2 weekdays - ACF of Residuals for MA(1)")
pacf(resid(MAi[[2]]), main = "Component 2 weekdays - PACF of Residuals for MA(1)")

MA <- arima(Z_week_ts, order = c(0, 0, 2))
MAi[[2]] <- MA

acf(resid(MAi[[2]]), main = "Component 2 weekdays - ACF of Residuals for MA(2)")
pacf(resid(MAi[[2]]), main = "Component 2 weekdays - PACF of Residuals for MA(2)")

MA <- arima(Z_week_ts, order = c(0, 0, 3))
MAi[[2]] <- MA

acf(resid(MAi[[2]]), main = "Component 2 weekdays - ACF of Residuals for MA(3)")
pacf(resid(MAi[[2]]), main = "Component 2 weekdays - PACF of Residuals for MA(3)")

MA <- arima(Z_wkend_holi_ts, order = c(0, 0, 1))
MAi[[3]] <- MA

acf(resid(MAi[[3]]), main = "Component 2 weekends/holidays - ACF of Residuals for MA(1)")
pacf(resid(MAi[[3]]), main = "Component 2 weekends/holidays - PACF of Residuals for MA(1)")

MA <- arima(Z_wkend_holi_ts, order = c(0, 0, 2))
MAi[[3]] <- MA
```

```r
acf(resid(MAi[[3]]), main = "Component 2 weekends/holidays - ACF of Residuals for MA(2)")
pacf(resid(MAi[[3]]), main = "Component 2 weekends/holidays - PACF of Residuals for MA(2)")
```

"mcatpp.R" function:

```r
  mcatpp <- function(x,p=0,rng,dgr=3,nkt=1,sm1=0,sm2=0,itmax=100,errmax=1e-4){
# Multiplicative Component Analysis for Temporal Point Processes
#   (PCA of log-intensities)
#
# INPUT:
#   x: Observed time points (list, length n).
#       Each x[[i]] is a vector containing the data from replication i.
#   p: Number of components (integer>=0; default p=0, mean-only model).
#   rng: Time range (vector, length 2).
#   dgr: Spline degree (integer; default dgr=3, cubic splines).
#   nkt: Number of equally-spaced basis knots (integer; default nkt=1).
#   sm1: Smoothing parameter for the mean (scalar>=0; default sm1=0).
#   sm2: Smoothing parameter for the components (scalar>=0; default sm2=0).
#         Note that all components have norm 1 but the mean does not,
#         so different sm's may be needed to attain the same smoothness).
#   itmax: Maximum number of iterations (integer; default itmax=100).
#   errmax: Maximum error tolerance for convergence (scalar>0; default errmax=1e-4)
#
# OUTPUT: List with the following elements
#   c0: Basis coefficients of the mean (q x 1).
#   C: Basis coefficients of the components (q x p).
#   s2: Component variances (p vector).
#   u: Individual component scores (n x p).
#   logf: Individual log-densities (n x 1).
#   tg: Output time grid (ng vector)
#   mu: Estimated mean (ng vector)
#   phi: Estimated components (norm 1) (ng x p)
#
# Version: 5 Feb 2023

# Loading required packages
require(splines2)
require(MASS)

### Input check and elimination of data out of RNG --------
n <- length(x)
m <- rep(0,n)
for (i in 1:n){
  in_range <- (x[[i]]>=rng[1])&(x[[i]]<=rng[2])
  x[[i]] <- x[[i]][in_range]
  m[i] <- length(x[[i]])
}
if (any(m>=200)){
  cat('Warning: Some x{i}s have more than 200 observations,\n
        which may cause NaNs in the pdfs.\n
        This method is intended for relatively small m(i)s,\n
        for large m(i)s you can just use kernel smoothing.')
}
```

```
### Initialization ------------
tg <- seq(rng[1],rng[2],length.out=500)
dt <- tg[2]-tg[1]
knt <- seq(rng[1],rng[2],length.out=nkt+2)[2:(nkt+1)]
B0 <- bSpline(tg,knots=knt,degree=dgr,intercept=T,Boundary.knots=rng)
J0 <- (t(B0)%*%B0)*dt
q <- dim(B0)[2]
if (dgr>=3){
  B2 <- bSpline(tg,knots=knt,degree=dgr,intercept=T,Boundary.knots=rng,derivs=2)
  J2 <- (t(B2)%*%B2)*dt
} else {
  J2 < matrix(0,q,q)
}
sumB <- matrix(0,n,q)
for (i in 1:n){
  if (m[i]>0){
    B_i <- bSpline(x[[i]],knots=knt,degree=dgr,intercept=T,Boundary.knots=rng)
    sumB[i,] <- colSums(B_i)
  }
}

#### Initial mean-only model -------------
c0 <- matrix(log(mean(m)/(rng[2]-rng[1])),q,1)
logf <- complogf0(sumB,c0,dt,B0,m)
OF <- mean(logf)-sm1*t(c0)%*%J2%*%c0
cat('---> Computing mean\n')
cat('Iteration: 0, Pen. loglik: ',OF,'\n')
err <- 1
iter <- 0
while ((err>errmax) & (iter<itmax)){
  iter <- iter + 1
  c00 <- c0
  OF0 <- OF
  derivs <- derivc0(sumB,c0,dt,B0)
  gpll <- derivs$gc - 2*sm1*J2%*%c0
  Hpll <- derivs$Hc - 2*sm1*J2
  direction <- solve(Hpll,gpll)
  OF <- -Inf
  k <- 0
  while ((OF<=OF0)&(k<6)){
      step <- 0.7^k
      c0 <- c00-step*direction
      logf <- complogf0(sumB,c0,dt,B0,m)
      OF <- mean(logf)-sm1*t(c0)%*%J2%*%c0
      k <- k+1
  }
  if (OF<=OF0){
      cat('No further improvement in obj. func. is possible\n')
      c0 <- c00
      OF <- OF0
  }
  err <- sqrt(sum((c0-c00)^2))/sqrt(sum(c00^2))
  cat('Iteration:',iter,', Pen. loglik:',OF,', Error:',err,"\n")
```

```
}

#### Sequential PC estimation ----------
if (p==0){
  C <- NULL
  s2 <- NULL
  u <- NULL
  u2 <- NULL
} else {
  C <- matrix(0,q,p)
  s2 <- rep(0,p)
  u <- matrix(0,n,p)
  u2 <- matrix(0,n,p)
  for (ic in 1:p){
    if (ic==1){
      P <- diag(1,nrow=q)
    } else {
      P <- Null(J0%*%C[,1:(ic-1)])
    }
    # Initial estimators
    C[,ic] <- (P%*%t(P))%*%matrix(1,q,1)
    C[,ic] <- C[,ic]/c(sqrt(t(C[,ic])%*%J0%*%C[,ic]))
    if (ic==1){
      Ilmb0 <- sum(exp(B0%*%c0))*dt
      u[,ic] <- sqrt(rng[2]-rng[1])*log(pmax(m,1)/Ilmb0)
      s2[ic] <- sd(u[,ic])^2
    } else {
      s2[ic] <- s2[ic-1]/2
    }
    eff <- compeff(sumB,c0,C[,1:ic],s2[1:ic],dt,B0,m)
    u[,1:ic] <- eff$u
    u2[,1:ic] <- eff$u2
    logf <- eff$logf
    OF <- mean(logf) - sm1*t(c0)%*%J2%*%c0
        - sm2*sum(diag(t(C[,1:ic])%*%J2%*%C[,1:ic]))
    cat('---> Computing component ',ic,"\n")
    cat('Iteration: 0, Pen. loglik: ',OF,"\n")
    err <- 1
    iter <- 0
    # Iterations
    while ((err>errmax)&(iter<itmax)){
      iter <- iter + 1
      # Update C
      c00 <- C[,ic]
      u00 <- u
      u200 <- u2
      OF0 <- OF
      derivs <- derivc(sumB,c0,C[,1:ic],dt,B0,u[,1:ic],u2[,1:ic])
      gpll <- derivs$gc - 2*sm2*J2%*%C[,ic]
      Hpll <- derivs$Hc - 2*sm2*J2
      direction <- P%*%( solve(t(P)%*%Hpll%*%P,t(P)%*%gpll) )
      OF <- -Inf
      k <- 0
      while ((OF<=OF0)&(k<6)){
```

```
          step <- 0.7^k
          C[,ic] <- c00-step*direction
          C[,ic] <- C[,ic]/c(sqrt(t(C[,ic])%*%J0%*%C[,ic]))
          eff <- compeff(sumB,c0,C[,1:ic],s2[1:ic],dt,B0,m)
          u[,1:ic] <- eff$u
          u2[,1:ic] <- eff$u2
          logf <- eff$logf
          OF <- mean(logf) - sm1*t(c0)%*%J2%*%c0
                - sm2*sum(diag(t(C[,1:ic])%*%J2%*%C[,1:ic]))
          k <- k+1
        }
        if (OF<=OF0) {
          cat('No further improvement in obj. func. is possible\n')
          C[,ic] <- c00
          u <- u00
          u2 <- u200
        }
        # Update s2
        s2 <- colMeans(u2)
        # Stopping criterion
        err <- sqrt(sum((C[,ic]-c00)^2))/sqrt(sum(c00^2))
        cat('Iteration: ',iter, ', Pen. loglik: ',OF,', Error: ',err,"\n")
      }
    }
  }

  #### Output ---------
  dimnames(c0) <- NULL
  mu <- B0%*%c0
  phi <- NULL
  if (p>=1){
    phi <- B0%*%C
  }
  return(list(c0=c0,C=C,s2=s2,u=u,logf=logf,tg=tg,mu=mu,phi=phi))
}

########## AUXILIARY FUNCTIONS -------------------

complogf0 <- function(sumB,c0,dt,B0,m){
  # Computes log-densities for mean-only model
  logf <- -sum(exp(B0%*%c0))*dt + sumB%*%c0 - lfactorial(m)
  return(logf)
}

derivc0 <- function(sumB,c0,dt,B0){
  # Derivatives of loglik/n with respect to c0
  q <- dim(B0)[2]
  Hc0 <- -t(B0)%*%((exp(B0%*%c0)%*%rep(1,q))*B0)*dt
  gc0 <- -t(B0)%*%exp(B0%*%c0)*dt + colMeans(sumB)
  return(list(gc=gc0,Hc=Hc0))
}

compeff <- function(sumB,c0,C,s2,dt,B0,m){
  # Computes random effects and log-pdf using Laplace approximation
```

```
  if (is.vector(C)){
    C <- matrix(C,length(C),1)
  }
  eps <- 1e-16
  n <- dim(sumB)[1]
  p <- length(s2);
  Phi <- B0%*%C
  logf <- rep(0,n)
  u <- matrix(0,n,p)
  u2 <- matrix(0,n,p)
  for (i in 1:n){
    # Compute log(f(x))
    uL <- u[i,]
    D_gi <- matrix(0,1,p)
    H_gi <- diag(p)
    steps <- 0
    while ((steps<5)&(rcond(H_gi)>eps)){
        steps <- steps + 1
        H_gi_old <- H_gi
        uL <- uL - D_gi%*%solve(H_gi)
        lmbi <- exp(B0%*%c0+B0%*%C%*%t(uL))
        D_gi <- -t(lmbi)%*%Phi*dt + sumB[i,]%*%C - uL/s2
        H_gi <- -t(Phi)%*%((lmbi%*%rep(1,p))*Phi)*dt - diag(1/s2,nrow=length(s2))
    }
    if (rcond(H_gi)<=eps){
        H_gi <- H_gi_old
    }
    gi <- -sum(lmbi)*dt + sumB[i,]%*%(c0+C%*%t(uL)) - lfactorial(m[i])
          -sum(uL^2/(2*s2)) - .5*sum(log(2*pi*s2))
    logf[i] <- gi + (p/2)*log(2*pi) - .5*logdet(-H_gi)
    S <- solve(-H_gi)
    u[i,] <- uL
    u2[i,] <- diag(S) + uL^2
  }
  return(list(u=u,u2=u2,logf=logf))
}

derivc <- function(sumB,c0,C,dt,B0,u,u2){
  # Derivatives of loglik/n with respect to C[,end]
  # Uses ad-hoc approx of second derivatives and plug-in scores for integrals
  if (is.vector(C)){
    C <- matrix(C,length(C),1)
    u <- matrix(u,length(u),1)
    u2 <- matrix(u2,length(u2),1)
  }
  n <- dim(u)[1]
  p <- dim(u)[2]
  q <- length(c0)
  ng <- dim(B0)[1]
  lmb <- exp((B0%*%c0)%*%rep(1,n) + B0%*%C%*%t(u))
  ulmb <- (rep(1,ng)%*%t(u[,p]))*lmb
  u2lmb = (rep(1,ng)%*%t(u2[,p]))*lmb
  gc <- -t(B0)%*%rowMeans(ulmb)*dt + t(sumB)%*%u[,p]/n
  Hc <- -t(B0)%*%((matrix(rowMeans(u2lmb),ng,1)%*%rep(1,q))*B0)*dt
```

```
  return(list(gc=gc,Hc=Hc))
}

logdet <- function(A){
  # log(det(A)) for symmetric non-negative definite A
  R <- chol(A)
  return(2*sum(log(diag(R))))
}
```

"simul.R" function:

```
    gen1 <- function(n,a) {
  #
  # Generates one-component model with AR(1) scores
  # The AR coefficient is "a", the sample size is "n"
  #

  # Functional parameters and component scores
  tg <- seq(0,1,length.out=500);
  mu <- 5*sin(pi*tg)  # Int of exp(mu(t,0)) is 54.3
  phi <- sqrt(2)*sin(2*pi*tg)
  sigma <- .2
  w <- rep(0,n)
  e <- rnorm(n)
  w[1] <- e[1]
  for (i in 2:n) {
    w[i] <- a*w[i-1] + e[i]
  }

  # Intensity functions
  lmb <- exp(rep(1,n)%*%t(mu)+sigma*w%*%t(phi))
  Ilmb <- rowSums(lmb)*(tg[2]-tg[1])
  Mlmb <- apply(lmb,1,max)

  # Data
  m <- rpois(n=n,lambda=Ilmb)
  x <- vector("list",n)
  for (i in 1:n) {
    f_lmb <- approxfun(tg,lmb[i,])
    x[[i]] <- rep(0,m[i])
    if (m[i]>0) {
      # Generates data by acceptance/rejection
      k <- 0
      while (k < m[i]){
        tt <- runif(1)
        y <- Mlmb[i]*runif(1)
        if (y <= f_lmb(tt)) {
          k <- k+1
          x[[i]][k] <- tt
        }
      }
    }
  }
```

```
  # Output
  return(list(x=x,m=m,w=w,sigma=sigma,a=a,tg=tg,mu=mu,phi=phi,lmb=lmb))
}

#### ------------------------------------------------------------

gen2 <- function(n,b) {
  #
  # Generates one-component model with MA(1) scores
  # The MA coefficient is "b", the sample size is "n"
  #

  # Functional parameters and component scores
  tg <- seq(0,1,length.out=500);
  mu <- 5*sin(pi*tg)  # Int of exp(mu(t,0)) is 54.3
  phi <- sqrt(2)*sin(2*pi*tg)
  sigma <- .2
  w <- rep(0,n)
  e <- rnorm(n)
  w[1] <- e[1]
  for (i in 2:n) {
    w[i] <- b*e[i-1] + e[i]
  }

  # Intensity functions
  lmb <- exp(rep(1,n)%*%t(mu)+sigma*w%*%t(phi))
  Ilmb <- rowSums(lmb)*(tg[2]-tg[1])
  Mlmb <- apply(lmb,1,max)

  # Data
  m <- rpois(n=n,lambda=Ilmb)
  x <- vector("list",n)
  for (i in 1:n) {
    f_lmb <- approxfun(tg,lmb[i,])
    x[[i]] <- rep(0,m[i])
    if (m[i]>0) {
      # Generates data by acceptance/rejection
      k <- 0
      while (k < m[i]){
        tt <- runif(1)
        y <- Mlmb[i]*runif(1)
        if (y <= f_lmb(tt)) {
          k <- k+1
          x[[i]][k] <- tt
        }
      }
    }
  }

  # Output
  return(list(x=x,m=m,w=w,sigma=sigma,b=b,tg=tg,mu=mu,phi=phi,lmb=lmb))
}
```