A STATISTICAL MODEL FOR THE INFLUENCE OF TEMPERATURE ON BIKE DEMAND IN BIKE-SHARING SYSTEMS

by

Tobias Tietze

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 2019

ABSTRACT

A STATISTICAL MODEL FOR THE INFLUENCE OF TEMPERATURE ON BIKE DEMAND IN BIKE-SHARING SYSTEMS

by

Tobias Tietze

The University of Wisconsin-Milwaukee, 2019
Under the Supervision of Professor Daniel Gervini

Efficient fleet management is essential for bike-sharing systems. Thus, it is important to understand the impact of environmental factors on bike demand. This thesis proposes a method to analyze the influence of temperature on bike demand. Hourly temperature data are approximated by smoothed curves and modeled by functional principal components. Bike check-out times, which can be seen as realizations of a doubly stochastic process, are modeled using multiplicative component models on the underlying intensity functions. The respective component scores are then related via a multivariate regression model. An analysis of data from the Divvy system of the City of Chicago is presented as an example of application.

TABLE OF CONTENTS

LIST OF FIGURES

1 Introduction

Sharing systems are becoming more and more common in many different parts of everyday life. In particular, many cities try to reduce traffic and pollution by the installation of bike-sharing systems. Their hope is that many people, both residents and tourists, use the bikes for spontaneous trips as well as regular routes and therefore do something good for the environment and reduce the risk of traffic jams. The basic requirement for bike sharing systems to be accepted and used by a lot of people is that the users have no downsides through using the bikes compared to the usage of cars. For the main part, this concerns the availability of the bikes and the bike stations. It is important that there are bike stations near to the intended starting and end points of the trip, that a bike is available to be checked out and that docks are vacant to check the bikes in afterwards. In order to plan the availability and to re-allocate bikes between stations, the operating company needs to understand the spatiotemporal patterns of bike demand.

There is already a lot of literature about different aspects of bike sharing system (e.g. Borgnat et al., 2011). One important question is how the bike demand can be modeled for a specific bike station. As the demand is different every day, the bike checkout times can be seen as random events of a temporal point process. Gervini (2017) introduces a multiplicative model that estimates the intensity function of such processes and gives the mathematical foundation to analyze the variability of the demand over various days.

Based on this approach, this thesis models the bike demand and examines the temperature as possible influential factor. For this purpose, the temperature is seen as functional data and approximated by smooth functions. Then, a principal component analysis is conducted on the obtained functions which represent different days. The procedures for the analyses applied on the functional data are for example given by Ramsay and Silverman (2005). After the variation in the bike demand and in the temperature is analyzed, the relationship between the results is investigated. The goal is to find out if the overall temperature as well as the fluctuation in the temperature influence the bike demand in terms of the volume and the time of checkouts.

The analysis is conducted on data from the bike-sharing system Divvy of the City of

Chicago. Specifically, bike checkout times at two bike stations with different characteristics are examined between April 1 and November 30 of 2016. The months December to March are left out as bike demand considerably decreases during the winter. Since temperature does not significantly change inside a city, the same temperature data is used for both bike stations.

The thesis is structured as follows. Chapter 2 explains the idea of functional data and how to find an approximating, smooth function that captures important patterns in the functional data. In addition, the principal component analysis is explained for functional data. In the next chapter, the multiplicative component model for replicated temporal point processes is introduced and followed by the estimation of the parameters. Afterwards, the mathematical foundation for a regression analysis of component scores is given in chapter 4. In chapter 5, the approaches of the previous chapters are used to analyze the influence of temperature on the bike sharing system. In particular, the smoothing mechanism and the principal component analysis are used for the temperature data and the bike demand data is analyzed by the use of the multiplicative component model. The relationship between these results is then examined in regression models. The thesis concludes in chapter 6 and provides an outlook on further possible analyses.

2 Functional data analysis

This chapter deals about the smoothing and the analysis of discrete, temporal data points. For this purpose, the data is not seen as a sequence of individual observations but rather as observed extracts of a latent data function. Let the discretely observed data be given by the m pairs (t_j, y_j) and the corresponding latent function denoted by x where y_j is the value of the function x at time t_j, probably blurred by measurement error. Therefore, the following equation is obtained

$$y_j = x(t_j) + \epsilon_j \quad for \quad j = 1, ..., m$$

which can be written in vector notation as

$$y = x(\boldsymbol{t}) + \epsilon$$

where $y = (y_1, ..., y_m)'$, $\boldsymbol{t} = (t_1, ..., t_m)'$ and $\epsilon = (\epsilon_1, ..., \epsilon_m)'$.
In most applications, not just a single function x is of interest but a collection of functional data. Therefore, the observations of function x_i are given by the m_i pairs (t_{ij}, y_{ij}) with $j = 1, ..., m_i$.

2.1 Smoothing functional data

The goal of this subsection is to estimate the latent function of functional data. As the estimation of the function x_i normally takes place independently of the estimations of other functions x_k $(k \neq i)$, just a single function x with observations (t_j, y_j) $(j = 1, ..., m)$ is considered for simplicity in this section. The underlying function x should be smooth and should approximate the value y_j well for all observations $j = 1, ..., m$. A function is declared to be smooth if it possesses one or more derivatives so that adjacent data points are unlikely to be very different. That also means that a smoothed function just reveals the general behavior of the data instead of every single noise terms.

A common approach to find x is to represent the function by a linear combination of basis functions.

Let $\beta = (\beta_1, ..., \beta_K)'$ be a set of basis functions. These functions are chosen so that they are independent of each other and that the linear combination of a sufficiently large number K of them can approximate any function arbitrarily well. The function x can then be expressed by

$$x(t_j) = \sum_{k=1}^{K} c_k \beta_k(t_j)$$

which can be written in vector notation as

$$x(\boldsymbol{t}) = \sum_{k=1}^{K} c_k \beta_k(\boldsymbol{t}) = Bc$$

where $c = (c_1, ..., c_K)'$ is the vector of length K that contains the coefficients and B is the $m \times K$ matrix that contains the values $\beta_k(t_j)$.

Popular choices of basis function systems are the Fourier series system

$$1, \sin(\omega t), \cos(\omega t), \sin(2\omega t), \cos(2\omega t), \sin(3\omega t), \cos(3\omega t), ...$$

the truncated polynomial basis

$$1, t, t^2, t^3, ..., t^d, (t - \tau_1)_+^d, ..., (t - \tau_L)_+^d$$

and the B-spline basis which is the most common choice for non-periodic functional data and therefore used in this thesis.

Both the truncated polynomial basis and the B-spline basis are splines of degree d. A spline of order $d+1$ is defined by a division of the interval over which the function has to be approximated into $L+1$ subintervals which are separated by L knots. Over each of the $L+1$ subintervals, the spline function is a polynomial of degree d. At every knot, the

spline has $d-1$ continuous derivatives. If r knots are coincident, only $d-r$ derivatives are continuous at this so-called breakpoint. Therefore, a spline has a great flexibility and can approximate a given function very well.

Let $\tau = (\tau_1, ..., \tau_L)$ be the non-decreasing knot sequence of length L. As the truncated polynomial basis consists of $d+1$ monomials and L piecewise polynomial functions which are all linearly independent, the dimension of this basis is $L+d+1$. Although this basis is simple to construct, the functions grow rapidly without a bound and are nonzero for most values of t. This produces numerical problems, especially for large values of t.

The B-spline basis is an equivalent basis to the truncated polynomial basis of the same dimension $L+d+1$ and solves this problem. As the functions are chosen to have minimal support, the B-splines are roughly orthogonal and the computation of x is very efficient. In addition, at any given time point, the sum of the B-splines is normalized to 1 and the single functions do therefore not grow rapidly. Because of that, the B-spline basis is numerically more stable. The technical details about a recursive procedure to find the B-spline functions can be looked up in de Boor (2001).

After the basis functions are chosen, recall the model that has to be fitted:

$$y = x(\boldsymbol{t}) + \epsilon \tag{1}$$

with

$$x(\boldsymbol{t}) = \sum_{k=1}^{K} c_k \beta_k(\boldsymbol{t}) = Bc \tag{2}$$

Here the residuals ϵ_j are assumed to be independently and identically distributed with mean 0 and constant variance σ^2. Therefore, the best linear unbiased estimator is given by the ordinary least squares estimator and can be obtained by finding the vector of

coefficients c that minimizes the regression least squares criterion

$$SMSSE(y|c) = \sum_{j=1}^{m} \left(y_j - \sum_{k=1}^{K} c_k \beta_k(t_j) \right)^2 = (y - Bc)'(y - Bc) = ||y - Bc||^2. \quad (3)$$

Setting the first derivative with respect to c of the criterion equal to 0 yields the following equation

$$2BB'c - 2B'y = 0.$$

Solving this for c then gives the optimal estimate $\hat{c}$ of the coefficients

$$\hat{c} = (B'B)^{-1}B'y$$

and the optimal estimate $\hat{y}$ of the fitted values

$$\hat{y} = \hat{x}(\mathbf{t}) = B\hat{c} = B(B'B)^{-1}B'y.$$

This fit approximates the values y the best but might therefore be not very smooth. On the other hand, a greatly smooth function will deviate a lot from the actual values.

Instead of defining the smoothness of $\hat{y}$ beforehand by the number K of basis functions, a often used approach is to optimize a fitting criterion that defines how smoothly the data should be approximated. That means a roughness penalty term with smoothing parameter is added to the criterion (3). As a curve is rough if it has rapid local variation, the added term needs to penalize high absolute values in the second derivative. Thus, the roughness penalty

$$PEN_2(x) = \int (D^2 x(t))^2 dt$$

with the second derivative $D^2x(t)$ of $x(t)$ is introduced and can be generalized to

$$PEN_r(x) = \int (D^r x(t))^2 dt.$$

This generalization is needed if the $(r-2)^{nd}$ derivative of a function should also be approximated as a smooth curve. $PEN_2(x)$ can be rewritten as

$$PEN_2(x) = \int (D^2 x(t))^2 dt = \int (D^2 c'\beta(t))^2 dt = c'(\int (D^2\beta(t))^2 dt)c =: c'R_2c.$$

The penalized residual sum of squares that has to be minimized is then given by

$$PENSSE(y|c) = (y - Bc)'(y - Bc) + \lambda c'R_2c \tag{4}$$

with the smoothing parameter λ. A high value of λ means that the curve will be very smooth and a low value of λ means that the curve is less smooth.
Setting the first derivative with respect to c of the new criterion (4) equal to 0 yields the following equation

$$2BB'c - 2B'y - 2\lambda R_2 c = 0.$$

Solving this for c then gives the optimal estimate $\hat{c}$ of the coefficients

$$\hat{c} = (B'B + \lambda R_2)^{-1}B'y$$

and of the vector $\hat{y}$ of the fitted values

$$\hat{y} = \hat{x}(\mathbf{t}) = B\hat{c} = B(B'B + \lambda R_2)^{-1}B'y.$$

In order to obtain not just the fitted values for the time points $t_1, ..., t_m$ but also for time points in between, the basis functions can be evaluated at a finer time grid $t^* = (t_1^*, ..., t_{m^*}^*)$ with $t_1 \leq t_1^*$ and $t_{m^*}^* \leq t_m$. Let B^* denote the m^* by K matrix that contains the values

$\beta_k(t_j^*)$. Then the new vector of fitted values

$$\hat{y}^* = B^*\hat{c} = B^*(B^{*\prime}B^* + \lambda R_r)^{-1}B^{*\prime}y$$

discretizes the smoothed function x.

2.2 Principal component analysis

Now consider a collection of smoothed functions $(x_1, ..., x_n)$. The idea of the principal component analysis (PCA) is to transform the set of n correlated functions into a smaller set of p orthogonal functions that account for as much of the variability in the data as possible. These new functions are called principal components and highlight the types of variation that mainly occur between the functions x_i.

The following procedure for the principal component analysis is given by Ramsay and Silverman (2005). First, subtract the mean function values of each x_i so that their cross-sectional means $n^{-1}\sum_{i=1}^n x_i(t)$ are 0. Then, consider the following quantities

$$f_i = \int \xi(t)x_i(t)dt$$

where ξ is a weight function and x_i is one of the smoothed and centered functions.

In the first PCA step, choose the weight function $\xi_1(t)$ that maximizes

$$n^{-1}\sum_{i=1}^n f_{1i}^2 = n^{-1}\sum_{i=1}^n \left(\int \xi_1(t)x_i(t)dt\right)^2$$

under the constraint

$$\int \xi_1(t)^2 dt = 1.$$

$\xi_1(t)$ is called the first principal component and f_{1i} is called the first principal component score for the i^{th} observation. The motivation is that the function $\xi_1(t)$ maximizes the mean squares by weighting the areas of highest variation.

In the k^{th} PCA step, choose the weight function $\xi_k(t)$ that maximizes

$$n^{-1}\sum_{i=1}^{n} f_{ki}^2 = n^{-1}\sum_{i=1}^{n}\left(\int \xi_k(t)x_i(t)dt\right)^2$$

under the constraints

$$\int \xi_k(t)^2 dt = 1$$

and

$$\int \xi_l(t)\xi_k(t)dt = 0 \quad \forall 1 \leq l < k.$$

The orthogonality assumptions ensures that each principal component indicates a new type of variation compared to the previous ones. The principal component ξ_1 reflects the most variability between the functions x_j. The other principal components then contribute in descending order.

The values of the PC scores f_{ki} reveal which observations induce a lot of the k^{th} type of variation and which observations barely differ from the mean function. If the value of the PC score f_{ki} is extremely high or low, x_i strongly deviates from the mean function in the way described by the principal component ξ_k. If f_{ki} is around 0, x_i is similar to the mean function in this regard.

There are different computational methods for the functional principal component analysis. One approach is to discretize the observed functions x_i on a fine time grid of m equally spaced time points $t_1, ..., t_m$ that span the interval on which the function is defined. Let $\mathbf{X}$ be the $n \times m$ data matrix that consists of the values of the n functions $x_1, ..., x_n$ evaluated at the m time points $t_1, ..., t_m$. From here on, a multivariate principal component analysis of $\mathbf{X}$ is conducted. According to Ramsay and Silverman (2005), finding the princpipal

components is equivalent to finding the eigenvectors of the eigenequation

$$\mathbf{V}\mathbf{u} = \lambda\mathbf{u} \tag{5}$$

where, after centering $\mathbf{X}$, $\mathbf{V} = n^{-1}\mathbf{X}'\mathbf{X}$ is the sample variance-covariance matrix, $\mathbf{u}$ an eigenvector and λ the corresponding eigenvalue.

Once the vector principal components are found, the next step is to transform them back to the functional principal components. The functional version of the eigenequation (5) is given by

$$\int v(s,t)\xi(t)dt = \rho\xi(s) \tag{6}$$

where

$$v(s,t) = n^{-1}\sum_{i=1}^{n} x_i(s)x_i(t)$$

is the covariance function.
The left side of (6) can be written as

$$V\xi = \int v(\cdot,t)\xi(t)dt$$

with the integral transform V of the principal component weight function ξ. The eigenequation can then be expressed as

$$V\xi = \rho\xi. \tag{7}$$

The sample variance-covariance matrix $\mathbf{V}$ discretizes the integral transform V on the fine time grid and therefore has the elements $v(t_j, t_k)$. In the same way, the vector $\boldsymbol{\xi}$ of length m with elements $\xi(t_j)$ discretizes any given function ξ. Let T be the length of the interval

and $w = \frac{T}{m}$. Then (7) can be approximated for any t_j by

$$V\xi(t_j) = \int v(t_j, t)\xi(t)dt \approx w \sum_{k=1}^{m} v(t_j, t_k)\xi(t_k).$$

Thus, the functional eigenequation $V\xi = \rho\xi$ can be approximated in the discrete form

$$w\mathbf{V}\boldsymbol{\xi} = \rho\boldsymbol{\xi}.$$

The solutions of this equation correlate with the solutions of the multivariate eigenequation (5). It holds for the eigenvalues that

$$\rho = w\lambda.$$

To get the eigenfunction ξ, the discrete approximation $w||\boldsymbol{\xi}||^2 = 1$ of the functional normalization constraint $\int \xi(t)^2 dt = 1$ is used. Given a normalized eigenvector $\mathbf{u}$ of $\mathbf{V}$, the vector $\boldsymbol{\xi}$ is computed by

$$\boldsymbol{\xi} = w^{-1/2}\mathbf{u}.$$

The eigenfunction ξ can then be obtained by interpolating the values $\xi(t_j)$ of the vector $\boldsymbol{\xi}$. Since a fine time grid is used, the choice of interpolation method does not have a big effect on the results.

This procedure has finally provided the functional eigenvalues and the eigenfunctions. The k^{th} functional eigenvalue ρ_k reveals how much variability is explained by the k^{th} functional principal component. The share of the k^{th} functional principal component in the total variability is given by $\frac{\rho_k}{\sum_{l=1}^{n-1} \rho_l}$. Therefore, the p orthogonal components that account for as much variability in the data as possibly are the eigenfunctions that correspond to the p largest eigenvalues. The number p of principal components that are used for further interpretation and analysis can be chosen based on how much variability is sufficient to be explained.

3 Replicated temporal point processes

This section deals with point processes in $\mathcal{F}$, that means random countable sets in the set $\mathcal{F}$. A temporal point process is here defined as a point process in $\mathbb{R}$ and is needed to model an event that takes place randomly over time. In particular, it is random how often and when the event takes place. Although temporal point processes are used in chapter 5, this chapter uses general point processes in $\mathcal{F}$.

A point process X in $\mathcal{F}$ is locally finite if $\#(X \cap B) < \infty$ with probability 1 for any bounded subset $B \subseteq \mathcal{F}$. Define the count function $N(B) = \#(X \cap B)$ for any bounded subset $B \subseteq \mathcal{F}$. This count function characterizes the point process X and is therefore equivalent to X in this case. In addition, the intersection of X with the subset B is defined as $X_B = X \cap B$.
A function $\lambda : \mathcal{F} \to [0, \infty)$ is locally integrable if $\int_B \lambda(t)dt < \infty$ for any bounded subset $B \subseteq \mathcal{F}$. Given such a locally integrable function $\lambda : F \to [0, \infty)$, a point process X is a Poisson process with intensity function λ if

i) $N(B)$ is Poisson distributed with parameter $\int_B \lambda(t)dt$

ii) conditionally on $N(B) = m$, the m points in X_B are independent and identically distributed with density function $\tilde{\lambda} = \frac{\lambda}{\int_B \lambda(t)dt}$.

X is then denoted by $X \sim \mathcal{P}(\lambda)$.

3.1 Multiplicative component model

This subsection provides an additive model for log-intensity functions of Poisson processes introduced by Gervini (2017). The motivation is to characterize replicated Poisson processes in a way to reveal the variation between these processes. The results can then be interpreted in the same way as for a principal component analysis in chapter 2.

Let $X \sim \mathcal{P}(\lambda)$ and B a fixed bounded region of $\mathcal{F}$. The density function of X_B at

$x_B = \{t_1, ..., t_m\}$ is given by

$$\begin{aligned} f(x_B) &= f(\{t_1, ..., t_m\}) \\ &= f(N(B) = m) f(\{t_1, ..., t_m\} | N(B) = m) \\ &= \mathrm{e}^{-\int_B \lambda(t)dt} \frac{(\int_B \lambda(t)dt)^m}{m!} \prod_{i=1}^{m} \tilde{\lambda}(t_i) \\ &= \mathrm{e}^{-\int_B \lambda(t)dt} \frac{(\int_B \lambda(t)dt)^m}{m!} \frac{\prod_{i=1}^{m} \lambda(t_i)}{\prod_{i=1}^{m} \int_B \lambda(t)dt} \\ &= \mathrm{e}^{-\int_B \lambda(t)dt} \frac{1}{m!} \prod_{i=1}^{m} \lambda(t_i) \end{aligned} \tag{8}$$

The density function is used to find the probability that the realizations of X_B are in a specific locally finite subset of $\mathcal{F}$. Let $\mathcal{N} = \{A \subseteq \mathcal{F} | \#(A \cap B) < \infty$ for any bounded subset B $\subseteq \mathcal{F}\}$ be the family of locally finite subsets of $\mathcal{F}$. Then it holds for $F \subseteq \mathcal{N}$

$$\begin{aligned} P(X_B \in F) &= \sum_{m=0}^{\infty} \int_B ... \int_B \mathbb{1}(\{t_1, ..., t_m\} \in F) f(\{t_1, ..., t_m\}) dt_1 ... dt_m \\ &= \sum_{m=0}^{\infty} \mathrm{e}^{-\int_B \lambda(t)dt} \frac{1}{m!} \int_B ... \int_B \mathbb{1}(\{t_1, ..., t_m\} \in F) \prod_{i=1}^{m} \lambda(t_i) dt_1 ... dt_m \end{aligned}$$

and in general for any function $h : \mathcal{N} \to [0, \infty)$

$$\mathbb{E}(h(X_B)) = \sum_{m=0}^{\infty} \int_B ... \int_B h(\{t_1, ..., t_m\} f(\{t_1, ..., t_m\}) dt_1 ... dt_m.$$

Consider now n replicated Poisson processes $X_1, ..., X_n$. The corresponding intensity functions are assumed to be different because a single intensity function can barely model different replications with subject-specific patterns well. In addition, the intensity functions are treated as latent random effects since they are not observable in practice. Such replicated processes are then called doubly stochastic processes or Cox processes (Møller and Waagepetersen, 2004).

Let Λ be a random function which takes values on the space $\mathcal{G}$ of non-negative locally integrable functions on $\mathcal{F}$. A doubly stochastic process is given by a pair (X, Λ) with

$X|\Lambda = \lambda \sim \mathcal{P}(\lambda)$ where X is observable but Λ is not. Therefore, the distribution of X is characterized by the latent intensity process Λ. The n replicated Poisson processes are given by the independent and identically distributed realizations $(X_1, \Lambda_1), ..., (X_n, \Lambda_n)$ of (X, Λ).

Consider now the logarithm $\log \Lambda(t)$ of the latent intensity function. The model for this process is based on the Karhunen-Loève theorem which allows to represent the centered process as a linear combination of orthogonal functions

$$\log \Lambda(t) - \mu(t) = \sum_{k=1}^{\infty} U_k \phi_k(t)$$

with the mean function $\mu(t) \in L^2(B)$ of $\log \Lambda(t)$, ϕ_k's an orthonormal basis on $L^2(B)$, and uncorrelated random variables U_k's with mean 0 and variance σ_k^2. Rearranging this equation gives

$$\log \Lambda(t) = \mu(t) + \sum_{k=1}^{\infty} U_k \phi_k(t).$$

The stochastic process $\log \Lambda(t)$ is therefore given by a mean function and additional components which explain different types of variation in the process. Hence, this representation is closely related to the principal component analysis introduced in subsection 3.2 and can be interpreted similarly.

Since just the first few components are of interest in practice, the model is truncated to a number p of components. In addition, the coefficients U_k are assumed to be independent and normally distributed random variables in order to derive maximum likelihood estimators in the next subsections. The adjusted model is then given by

$$\log \Lambda(t) = \mu(t) + \sum_{k=1}^{p} U_k \phi_k(t) \tag{9}$$

where $\mu \in L^2(B)$, $\phi_1, ..., \phi_p$ are orthonormal functions in $L^2(B)$ and U_k's are independent $\mathbb{N}(0, \sigma_k^2)$ random variables.

This additive model for $\log \Lambda(t)$ can be translated into a multiplicative model for $\Lambda(t)$:

$$\Lambda(t) = \lambda_0(t) \prod_{k=1}^{p} \xi_k(t)^{U_k} \tag{10}$$

where $\lambda_0 = \mathrm{e}^{\mu}$ is the baseline intensity function and the $\xi_k = \mathrm{e}^{\phi_k}$ are multiplicative components.

The approach to estimate the mean function and the components in (9) is to model μ and $\phi_1, ..., \phi_p$ as linear combinations of basis functions $\beta_1, ..., \beta_q$. These basis function can, for example, again be B-splines which were introduced in subsection 2.1. Let $\boldsymbol{\beta} = (\beta_1, ..., \beta_q)'$ be the vector of the basis functions, $\boldsymbol{c}_0$ the vector of the weights corresponding to the mean function and $\boldsymbol{c}_k$ the vector of the weights corresponding to the k^{th} component. Then it follows with $\mu(t) = \boldsymbol{c}_0'\boldsymbol{\beta}(t)$ and $\phi_k(t) = \boldsymbol{c}_k'\boldsymbol{\beta}(t)$ that

$$\begin{aligned}
\log \Lambda(t) &= \mu(t) + \sum_{k=1}^{p} U_k \phi_k(t) \\
&= \boldsymbol{c}_0'\boldsymbol{\beta}(t) + \sum_{k=1}^{p} U_k \boldsymbol{c}_k'\boldsymbol{\beta}(t) \\
&= \boldsymbol{c}_0'\boldsymbol{\beta}(t) + \boldsymbol{U}'\boldsymbol{C}'\boldsymbol{\beta}(t) \\
&= (\boldsymbol{c}_0 + \boldsymbol{C}\boldsymbol{U})'\boldsymbol{\beta}(t)
\end{aligned}$$

where $\boldsymbol{C} = (\boldsymbol{c}_1, ..., \boldsymbol{c}_p)$ and $\boldsymbol{U} = (U_1, ..., U_p)'$. Then, the coefficient vectors $\boldsymbol{c}_0$, $\boldsymbol{c}_k$'s and the variances σ_k^2's have to be estimated in order to get an estimation for the mean and component functions and therefore for the distribution of the Poisson process X.

3.2 Estimation

Let the single vector $\boldsymbol{\theta}$ be the collection of $\boldsymbol{c}_0$, $\boldsymbol{c}_k$'s and σ_k^2's. In order to be able to compute the probability function $P(X_B \in F)$ in (38), consider the marginal density function of

X_B at $x_B = \{t_1, ..., t_m\}$ using the parameter vector $\boldsymbol{\theta}$:

$$\begin{aligned} f(x_B; \boldsymbol{\theta}) &= \int f(x_B, \boldsymbol{u}) d\boldsymbol{u} \\ &= \int f(x_B|\boldsymbol{u}) f(\boldsymbol{u}) d\boldsymbol{u}. \end{aligned}$$

Conditioned on a realization $\boldsymbol{u} = (u_1, ..., u_p)^T$ of the random vector $\boldsymbol{U}$, model (9) provides a realization $\log \lambda_{\boldsymbol{u}}$ of the process $\log \Lambda$ which then gives together with (8) a non-random log-density function of X_B:

$$\begin{aligned} \log f(x_B|\boldsymbol{u}) &= -\int_B \lambda_{\boldsymbol{u}}(t) dt + \sum_{i=1}^{m} \log \lambda_{\boldsymbol{u}}(t_i) - \log m! \\ &= -\int_B \mathrm{e}^{(\boldsymbol{c}_0 + \boldsymbol{C}\boldsymbol{u})' \boldsymbol{\beta}(t)} dt + (\boldsymbol{c}_0 + \boldsymbol{C}\boldsymbol{u})' \sum_{i=1}^{m} \boldsymbol{\beta}(t_i) - \log m! \end{aligned}$$

which translates to a formula for the non-random density function of X_B conditioned on $\boldsymbol{u}$:

$$\begin{aligned} f(x_B|\boldsymbol{u}) &= \exp\left(-\int_B \mathrm{e}^{(\boldsymbol{c}_0 + \boldsymbol{C}\boldsymbol{u})' \boldsymbol{\beta}(t)} dt + (\boldsymbol{c}_0 + \boldsymbol{C}\boldsymbol{u})' \sum_{i=1}^{m} \boldsymbol{\beta}(t_i) - \log m! \right) \\ &= \frac{1}{m!} \exp\left(-\int_B \mathrm{e}^{(\boldsymbol{c}_0 + \boldsymbol{C}\boldsymbol{u})' \boldsymbol{\beta}(t)} dt + (\boldsymbol{c}_0 + \boldsymbol{C}\boldsymbol{u})' \sum_{i=1}^{m} \boldsymbol{\beta}(t_i) \right) \end{aligned}$$

On the other hand, the density function of the random vector $\boldsymbol{U}$ consisting of the independent $\mathbb{N}(0, \sigma_k^2)$-distributed random variables U_k is given by the product of the individual density functions:

$$\begin{aligned} f(\boldsymbol{u}) &= \prod_{k=1}^{p} \frac{1}{\sqrt{2\pi\sigma_k^2}} \mathrm{e}^{-\frac{u_k^2}{2\sigma_k^2}} \\ &= \frac{1}{(2\pi)^{\frac{p}{2}} \prod_{k=1}^{p} \sigma_k} \mathrm{e}^{-\sum_{k=1}^{p} \frac{u_k^2}{2\sigma_k^2}} \end{aligned}$$

which yields the following formula for the logarithm of the density function of $\boldsymbol{U}$:

$$\log f(\boldsymbol{u}) = \sum_{k=1}^{p} \left(-\frac{1}{2} \log 2\pi\sigma_k^2 - \frac{u_k^2}{2\sigma_k^2} \right).$$

Although $f(x_B|\boldsymbol{u})$ and $f(\boldsymbol{u})$ are given by concrete formulas, there is no closed form solution for $f(x_B;\boldsymbol{\theta})$. However, $f(x_B;\boldsymbol{\theta})$ can be approximated by Laplace's method using $\log f(x_B|\boldsymbol{u})$ and $\log f(\boldsymbol{u})$ which is explained in the technical supplement of Gervini (2017). The model parameters are estimated by a maximum likelihood approach with suitable roughness penalties that provide for a smooth mean function μ and smooth components $\phi_1, ..., \phi_p$.

Given n independent realizations $x_1, ..., x_n$ of X_B, the penalized maximum likelihood estimator $\hat{\boldsymbol{\theta}}$ is given by

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \rho_n(\boldsymbol{\theta})$$

with

$$\rho_n(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n} \log f(x_i;\boldsymbol{\theta}) - \nu_1 P(\mu) - \nu_2 \sum_{k=1}^{p} P(\phi_k) \tag{11}$$

where ν_1 and ν_2 are smoothing parameters and $P(g) = \int_B (D^2 g(t))^2 dt$ is a penalty function for the mean and the components. This optimization problem can be solved by using the Newton-Raphson algorithm that updates the parameters by iteration. In this connection, it is beneficial to use the representations $P(\mu) = \boldsymbol{c}_0^T \boldsymbol{\Omega} \boldsymbol{c}_0$ and $P(\phi_k) = \boldsymbol{c}_k^T \boldsymbol{\Omega} \boldsymbol{c}_k$ for a matrix $\boldsymbol{\Omega}$ which is also derived in the technical supplement of Gervini (2017). In addition, the Newton-Raphson algorithm can be adjusted with a cyclic border condition so that $\Lambda(a) = \Lambda(b)$ holds on the interval $B = [a, b]$.

Finally, the procedure provides parameter estimates $\hat{\boldsymbol{c}}_{\boldsymbol{0}}$, $\hat{\boldsymbol{c}}_{\boldsymbol{k}}$'s and $\hat{\sigma}_k^2$'s which then give an estimated baseline intensity function

$$\hat{\lambda}_0(t) = \mathrm{e}^{\hat{\boldsymbol{c}}_{\boldsymbol{0}}' \boldsymbol{\beta}(t)}$$

and estimated multiplicative components

$$\hat{\xi}_k(t) = \mathrm{e}^{\hat{\boldsymbol{c}}_{\boldsymbol{k}}{}'\boldsymbol{\beta}(t)}.$$

In addition, an estimate $\boldsymbol{u}$ of $\boldsymbol{U}$ is found for the realization x_B of X_B during the Laplace approximation of $f(x_B;\boldsymbol{\theta})$. Therefore, repeating this step n times provides estimates $\boldsymbol{u}_1, ..., \boldsymbol{u}_n$ for all realizations $x_1, ..., x_n$ of X_B.

The k^{th} element of $\boldsymbol{u}_i$ then describes by how much realization x_i contributes to the variation described by the k^{th} component U_k. The component U_k explains $\frac{\sigma_k^2}{\sum_{k=1}^{p}\sigma_k^2} \cdot 100\%$ of the total variability.

4 Regression analysis of component scores

Chapter 2 and chapter 3 introduced procedures to analyze different types of data. Whereas the principal component approach analyzes the variation between smoothed functional data, the multiplicative component model reveals the variation between data from point processes. For both, several components with corresponding component scores were obtained.

In practice, a range of data is often analyzed with respect to different questions of interest, e.g. all days in 2018 are analyzed with respect to the temporal distribution of crimes as well as the hourly number of policemen on duty for the city of Chicago. If the variation in each data set is examined by one of the introduced approaches, it might now be of interest to investigate the relationship between the component scores. The motivation is to find out if a type of variation in a first data set is somehow correlated with a type of variation in a second data set.

Let $f_l = (f_{l1}, ..., f_{ln})'$ be the vector consisting of the principal component scores of n observations for the l^{th} principal component. On the other hand, for the multiplicative component model, let $u_k = (u_{k1}, ..., u_{kn})'$ be the vector consisting of the component scores of n observations for the k^{th} component. A standard approach to relate the corresponding component scores is a linear regression model. The straight line regression equation is given by

$$u_{ki} = \beta_0 + \beta_1 f_{li} + \epsilon_i \quad \text{for} \quad i = 1, \ldots, n \tag{12}$$

where β_0 is the intercept, β_1 is the slope and ϵ_i is the error term. The random error terms are assumed to be independent with mean 0 and constant variance.

(12) can be extended arbitrarily by adding power functions of f_{li} to the regression. The quadratic regression equation is for example given by

$$u_{ki} = \beta_0 + \beta_1 f_{li} + \beta_2 f_{li}^2 + \epsilon_i \quad \text{for} \quad i = 1, \ldots, n.$$

It is beneficial to first make a scatterplot of f_l and u_k. Plotting the corresponding f_{li} and u_{ki} against each other for all n observations uncovers any obvious pattern between these components. Based on this indication, an appropriate regression model can be chosen. After fitting a regression model, the coefficient of determination R^2 then provides a measure how well the observed u_{ki}'s are explained by the examined relationship. This measure is given by

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} = 1 - \frac{\sum_{i=1}^{n}(u_{ki} - \hat{u}_{ki})^2}{\sum_{i=1}^{n}(u_{ki} - \bar{u}_k)^2}$$

where $\hat{u}_{ki}$ is the fitted value of u_{ki} and $\bar{u}_k = \frac{1}{n}\sum_{i=1}^{n} u_{ki}$ is the average component score for the k^{th} component.

It holds $R^2 \approx 1$ if the fitted component scores almost match the observed component scores which means that the model almost perfectly replicates the observations. On the other hand, it holds $R^2 \approx 0$ if the fitted component scores barely match the observed component scores. That means, the model can not explain the observations.

In addition to the regression models, the correlation coefficient

$$\rho_{kl} = \frac{\sum_{i=1}^{n}(f_{li} - \bar{f}_l)(u_{ki} - \bar{u}_k)}{\sqrt{\sum_{i=1}^{n}(f_{li} - \bar{f}_l)^2}\sqrt{\sum_{i=1}^{n}(u_{ki} - \bar{u}_k)^2}}$$

can be computed in order to check for a linear relationship between the components. Here, $\bar{f}_l = \frac{1}{n}\sum_{i=1}^{n} f_{li}$ is the average component score for the l^{th} principal component and $\bar{u}_k = \frac{1}{n}\sum_{i=1}^{n} u_{ki}$ is again the average component score for the k^{th} component in the multiplicative model. If $|\rho_{kl}| \approx 1$ holds, the components are highly linearly correlated.

In chapter 5, functional data as well as data from point processes is analyzed. Afterwards, the relationship of their component scores is examined under the use of regression models and the introduced measures. However, the analyses explained in this chapter could also be applied on component scores that all come either from principal component analyses of functional data or from multiplicative models for point processes.

5 Application on bike sharing system

This chapter examines if the temperature had influence on the demand for the bike sharing system Divvy in Chicago between April and November 2016. For this purpose, the temperature in Chicago is examined to find the average temperature over the eight months and different types of variations which show how single days deviate from the average temperature. On the other hand, the trip starting times of two bike stations are also analyzed in order to find the temporal distribution of the bike demand and the major types of variation. Afterwards, the component scores of both analyses are used to investigate if there is a relationship between the bike demand and the temperature.

5.1 Temperature analysis

In this subsection, data of the temperature in Chicago is used which is publicly available on the data science website Kaggle (https://www.kaggle.com/). For each of the 244 days between 04/01/2016 and 11/30/2016, the hourly temperature data is available. Therefore, the functional data approach from chapter 2 is used to find 244 smooth functions representing the data and to analyze their variation.

As a first step, the data for the time frame of interest is extracted and converted from Kelvin to degree Fahrenheit. Then, the functional data is smoothed with a roughness penalty under the use of a cubic B-spline basis and ten equally spaced knots in the interval (0,24). The optimal smoothing parameter is $\lambda = 12.59$ and can be obtained by generalized cross-validation. The principal component analysis is then conducted with $p = 5$ components. As the first two of them already account for 97.83% of the variability, just these two components are considered in the following.
To illustrate and interpret the components ξ_1 and ξ_2, it is beneficial to plot the mean function of the temperature and to show how the first respectively the second type of variation affect the mean function. Therefore, a positive multiple of the components is separately added and subtracted to the mean function in figure 1 in order to point out

the effect of the corresponding type of variation. Here, the multiple 3 is sufficient to make the effect clear.

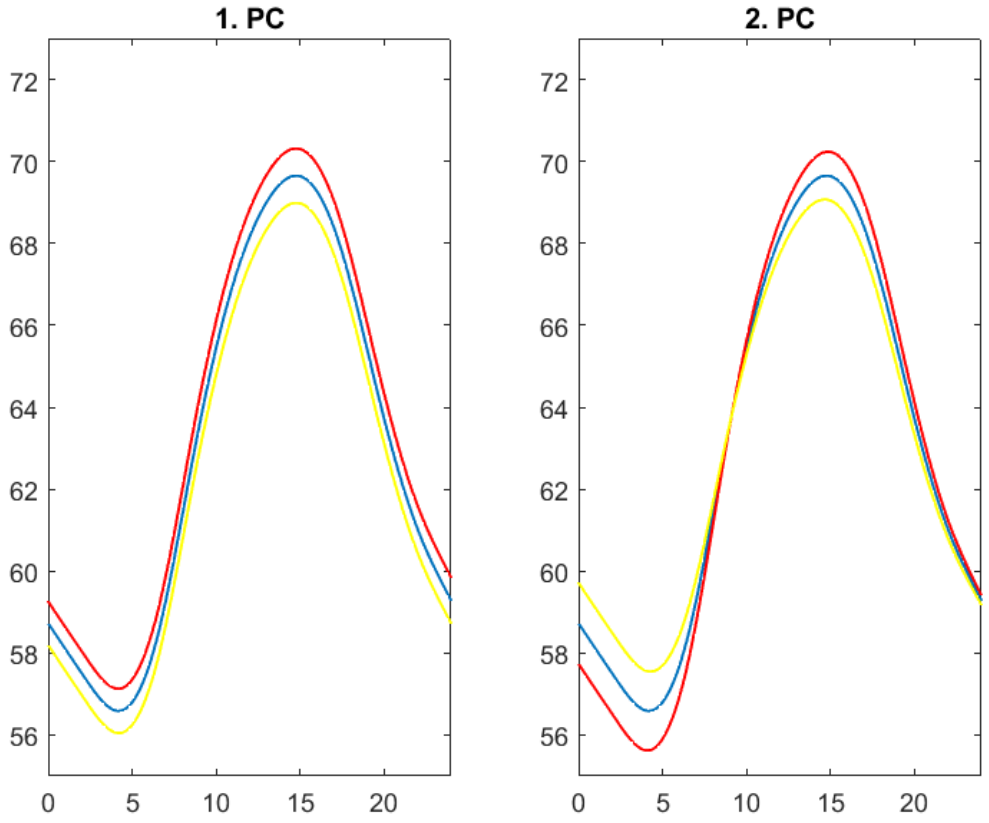

Figure 1: Effect of the first two principal components. (1.PC) Mean function (blue) and mean plus a positive (red) and negative (yellow) multiple of 1. PC. (2.PC) Mean function (blue) and mean plus a positive (red) and negative (yellow) multiple of 2. PC.

The mean temperature function shows approximately that, on average, the minimum daily temperature is 57 degree Fahrenheit at around 4am and the maximum daily temperature is 70 degree Fahrenheit at around 3pm. The first principal component is a size component which represents higher temperature over the entire day. On the other hand, the second principal component is a contrast component which represents a higher maximum temperature but a lower minimum temperature. This component therefore shows a higher amplitude between the daily maximum and minimum temperature. A negative multiple of the components shows in both cases the opposite effect. Most of the variation

between the days is due to the first component that accounts for 92.49% of the total variation. The second component just accounts for another 5.34% and is therefore less significant than the first one.

It is now of interest how each day differs from the mean function in terms of the effects demonstrated by the principal components. This question can be answered by looking at the principal component scores which are given in figure 2.

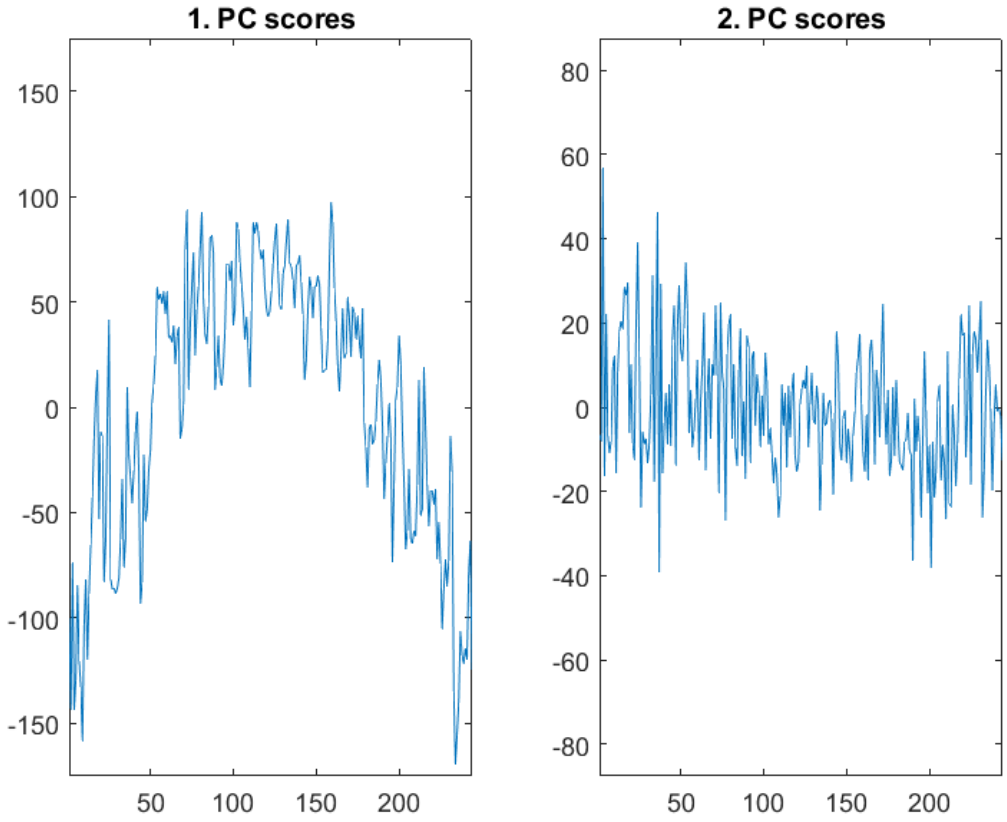

Figure 2: Daily component scores of first two principal components.

Days with a positive component score deviate from the mean function in the way described by adding the principal component to the mean function (see red function in figure 1) whereas days with a negative score show the reverse effect demonstrated by subtracting the principal component from the mean function (see yellow function in figure 1). In addition, the higher the absolute value of a score is, the stronger deviates the day from the mean function in the described way.

Considering the scores for the first PC, one can see the clear pattern that summer days

have a positive score and days in spring and fall have negative scores. This means that summer days tend to have a higher temperature over the entire day than days in spring and fall do. This insight is what one would naturally expect.
The scores of the second principal component show no obvious pattern except a spread around 0. This spread is a little bit wider in spring and in fall than in summer. This means that the difference between the daily maximum and minimum temperature fluctuates more in spring and in fall.

5.2 Bike demand analysis

In this subsection, the demand for the bike sharing system Divvy in Chicago is analyzed. The necessary open source data can be found on the Chicago Data portal website (https://data.cityofchicago.org). This data set gives for each of the 244 days between 04/01/2016 and 11/30/2016 the exact bike checkout times. That means the data for different days differs in terms of how often bikes were checked out and also at what time. Hence, each day is assumed to be a realization of a Poisson process with a corresponding unknown intensity function where the bike check outs are the events that take place. Therefore, the multiplicative component model from chapter 3 is used in order to model the random function Λ which characterizes the underlying Poisson process and explains the varying bike demand over the eight months.

The analysis is conducted for two different bike stations individually. Station 35 is located at the Navy Pier where a lot of people are on the move and 85314 bikes were check out between April and November 2016. On the other hand, station 166 at Ashland Avenue & Wrightwood Avenue is located in a more calm neighbourhood with just 4304 bike demands during the same time.
The model uses again a cubic B-spline basis and ten equally spaced knots in the interval (0,24) for both bike stations. First, consider station 35. The optimal smoothing parameters $\nu_1 = 0.0316$ and $\nu_2 = 0.1$ are obtained by cross-validation. Although five principal components are computed, just the first two are considered here since they already ac-

count for 95.59% of the variability.

First, based on the plotted baseline intensity function Λ, the effects of the two multiplicative components ξ_1 and ξ_2 are shown. This is done by multiplying and dividing the baseline function by one of the components which is taken to the power of a constant that points out the type of variation. Here again, the constant 3 is sufficient to highlight the effect. Therefore, the products $\lambda_0(t)\xi_k(t)^3$ and $\lambda_0(t)\xi_k(t)^{-3}$ are plotted for both components $k = 1, 2$ in figure 3.

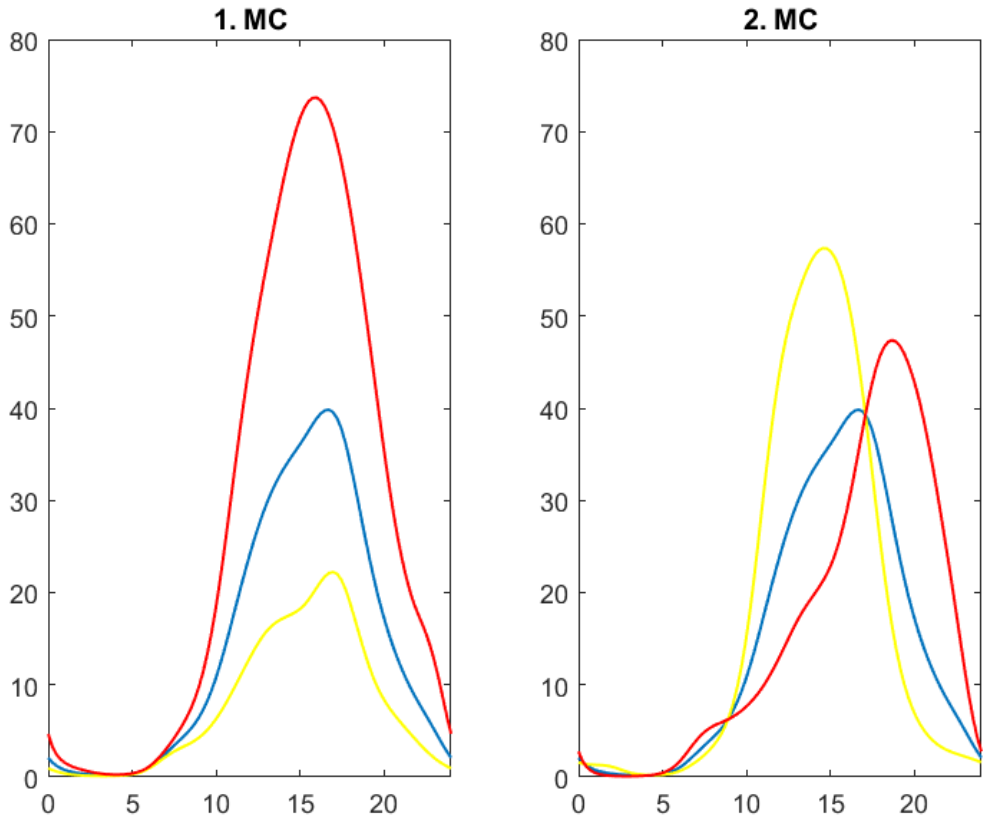

Figure 3: Station 35: Effect of the first two components. (1.MC) Baseline intensity (blue) and baseline times a positive (red) and negative (yellow) power of 1. MC. (2.MC) Baseline (blue) and baseline times a positive (red) and negative (yellow) power of 2. MC.

The baseline intensity function shows the highest demand approximately betweeen 4pm and 5pm and barely a bike check out in the early morning hours. The integral of the baseline intensity function would give the average bike demand over the 244 days. The

first multiplicative component is a size component which represents higher demand over the entire day. On the contrary, the second multiplicative component is a shift component for which the highest daily demand takes place a little bit later than usually. In addition, the demand is slightly more concentrated around the time of highest demand. A negative power of the components shows in both cases the opposite effect except of the concentration around the new time of highest demand in component 2. There, the demand is also higher than usual. Most of the variation between the days is due to the first component that accounts for 89.83% of the total variation. The second component just accounts for another 5.76% and is therefore less significant than the first one.
To find out which days show these deviations from the usual demand contribution, the component scores are plotted in figure 4.

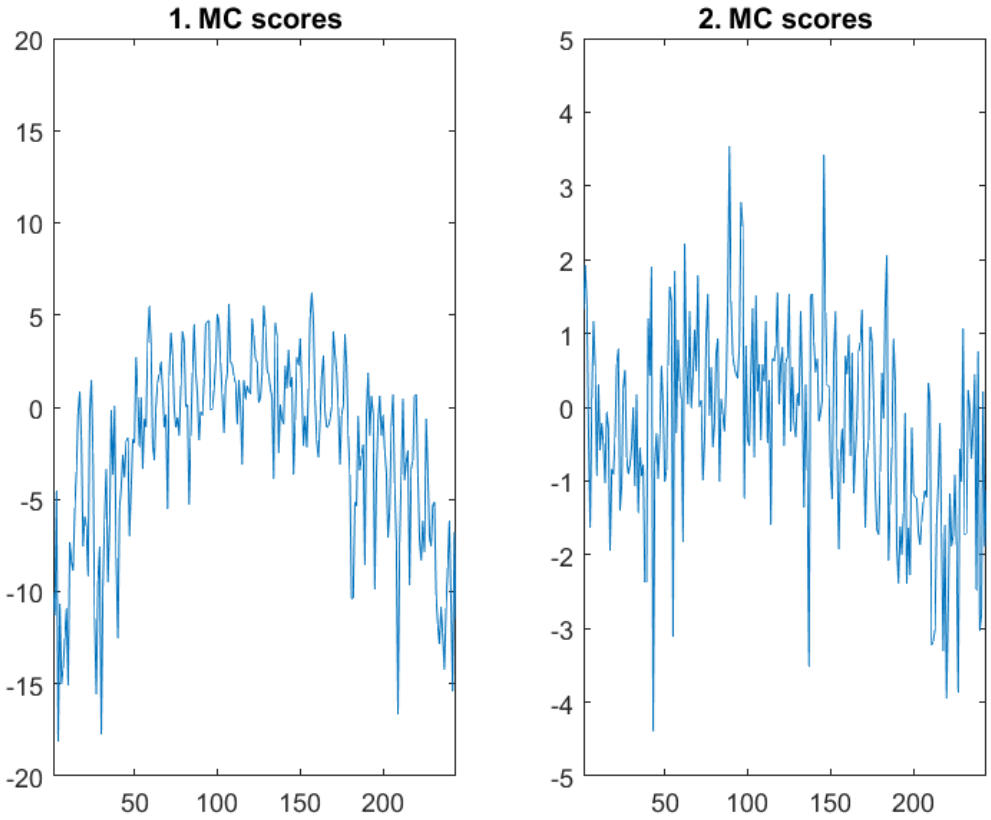

Figure 4: Station 35: Daily component scores of first two multiplicative components.

Considering the scores for the first PC, one can see the clear pattern that summer days have a positive score and days in spring and fall have highly negative scores. This means

that the demand tends to be generally higher over the entire day during the summer than during spring and fall.

The scores of the second principal component show a similar but clearly weaker pattern. The scores of summer days are higher again positive but the negative scores of the spring and fall time are just a little bit higher in absolute values. This relationship was considerably extremer for the first component. Nevertheless, the second component might suggest that people use the bikes a little bit later during the summer days and a little bit earlier during the spring and fall.

Now, the same analysis is conducted for station 166. A cross-validation provides the optimal smoothing parameters $\nu_1 = 0.001$ and $\nu_2 = 0.1$.

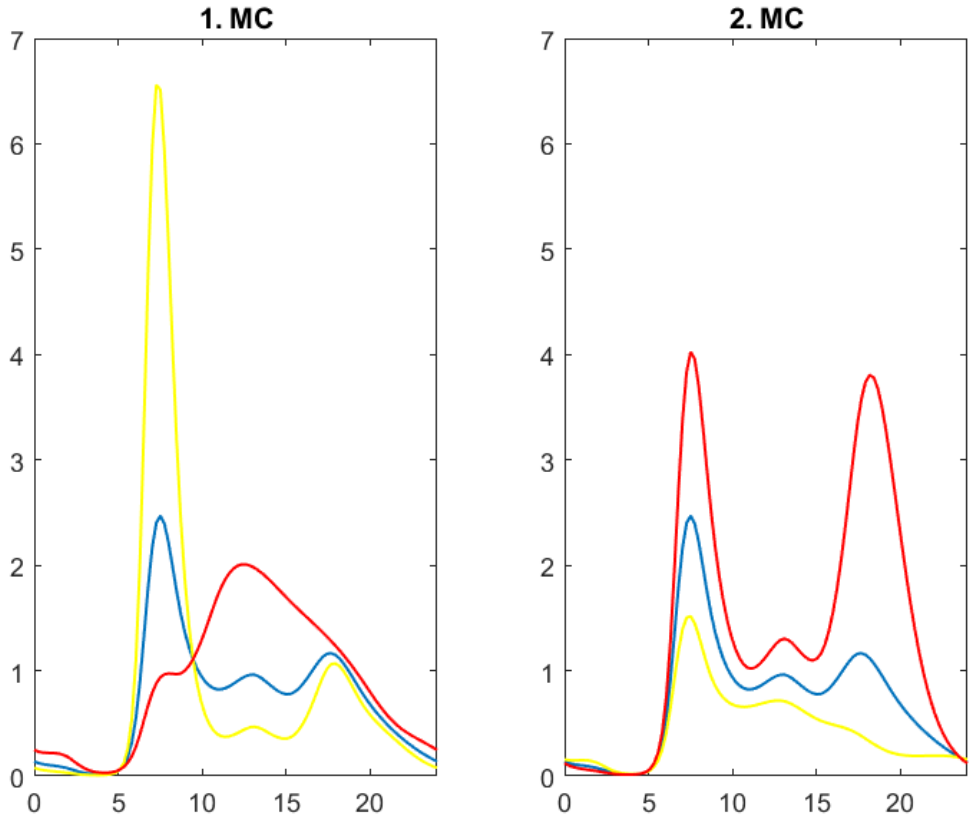

Figure 5: Station 166: Effect of the first two components. (1.MC) Baseline intensity (blue) and baseline times a positive (red) and negative (yellow) power of 1. MC. (2.MC) Baseline (blue) and baseline times a positive (red) and negative (yellow) power of 2. MC.

Again, consider the first two multiplicative components which account for 83.66% of the total variability. Figure 5 shows the baseline intensity function and the effects of the two components.

The baseline intensity function shows the highest demand at a short-time peak around at 8am. A reason for that could be that some people who live in that neighbour use the bike sharing system to get to work in the morning. The first component is a shape component as a positive power of the multiplier implies that most bikes are demanded around noon whereas a negative multiplier strongly concentrates most of the demand at 8am. On the other hand, the second component is a typical size component and represents more bike check outs over the entire day and two instead of one peak in the demand. This additional peak at around 6pm could be due to people who take the bike to get home after work. Here, the more important first component accounts just for 56.64% of the variability and the size component for at least 27.02% which is a very high amount for a second component.

Plotting the component scores for station 166 again provides insights which days contribute to which type of variation. Figure 6 shows the component scores for both types of variation.

The scores for the shape component are spread relatively even with no obvious pattern. That means there is no clear relationship between the time in the year and the shape of the demand during the day.

In contrast, the scores of the size component show the same pattern as seen for the size component of station 35. Days in the summer have a positive score whereas days in spring and fall have negative scores. This suggests again that the demand is generally higher in the summer.

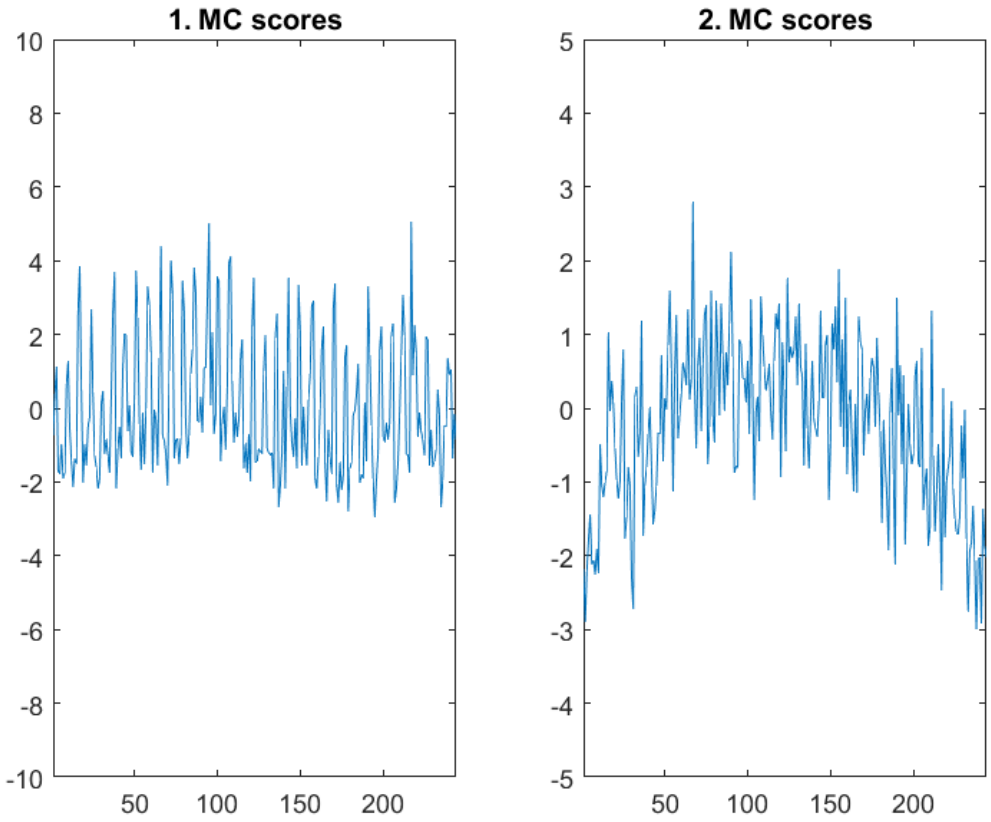

Figure 6: Station 166: Daily component scores of first two multiplicative components.

5.3 Regression analysis

In this subsection, the results from 5.1 and 5.2 are used to examine if the temperature had an influence on the bike demand between April and November 2016. Therefore, a regression approach is applied on the component scores of the temperature data and the component scores of the bike demand data. Here, to investigate the relationship between the k^{th} multiplicative component of the bike demand and the l^{th} principal component of the temperature, 244 pairs (f_{li}, u_{ki}) are used where f_{li} is the predictor value and u_{ki} is the response value.

First, consider again the data for station 35. The component scores obtained in the previous analysis are now plotted against each other in a scatterplot in figure 7.

The upper left scatterplot shows the scores of the first bike component plotted against the scores of the first temperature component. It is clearly observable that negative temperature component scores correspond to negative bike component scores and that

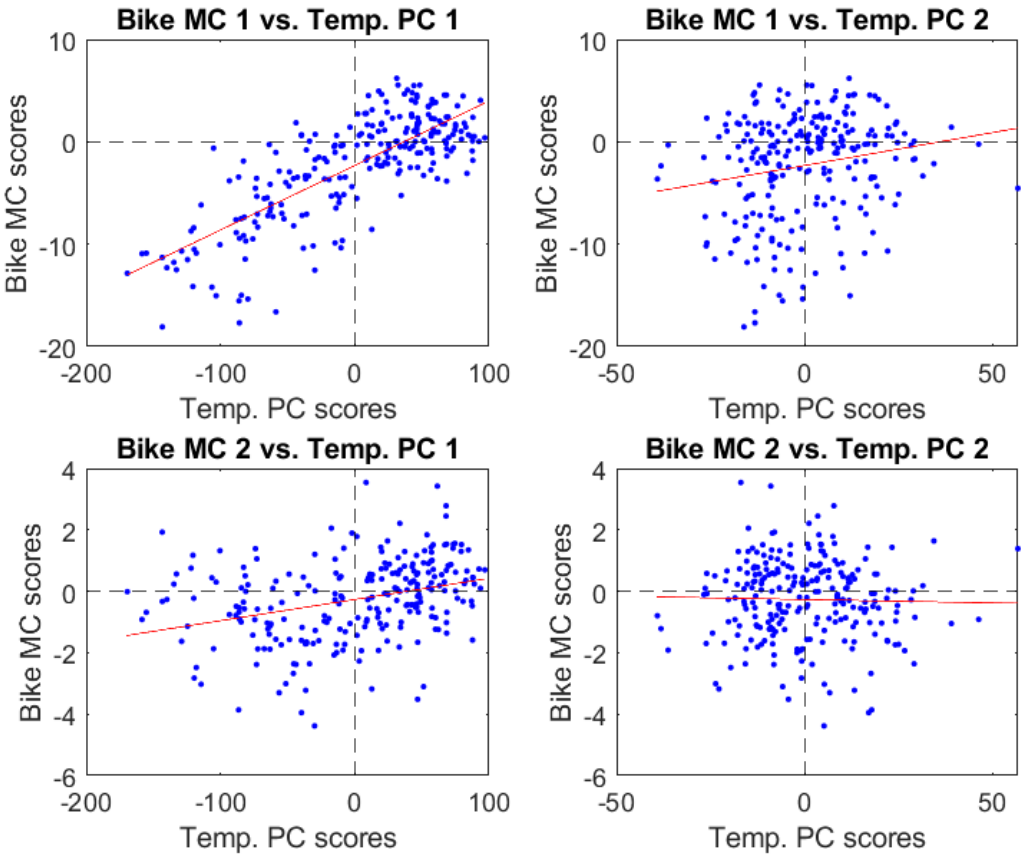

Figure 7: Station 35: Scatterplot of the component scores for the first bike MC and the first temperature PC (upper left plot). Scatterplot of the component scores for the first bike MC and the second temperature PC (upper right plot). Scatterplot of the component scores for the second bike MC and the first temperature PC (lower left plot). Scatterplot of the component scores for the second bike MC and the second temperature PC (lower right plot). All plots are supplemented by a fitted simple linear regression line.

there is an increasing linear relationship between them. Fitting a simple linear regression equation yields the following estimated relationship:

$$u_{1i} = -2.315 + 0.06306 f_{1i} \quad \text{for} \quad i = 1, \ldots, 244.$$

The corresponding coefficient of determination $R^2 = 0.5972$ and the correlation coefficient $\rho = 0.7728$ confirm the first guess of a linear relationship. Recall that both components are size components and higher scores mean in this context that there is a higher temperature respectively bike demand over the entire day. Therefore, the analysis suggests that a higher overall temperature yields to a higher bike demand over the entire day.

The upper right and lower right scatterplots illustrate the scores of the first respectively the second bike component plotted against the scores of the first temperature component. These plots show completely random scatters with no patterns. Therefore, there is no need to fit any regression in these cases. As the second temperature component was a contrast component that changes the amplitude between daily maximum and minimum temperature, it follows that the amplitude of the temperature fluctuation over a day does not influence bike demand.

The lower left scatterplot that relates the second bike component to the first temperature component could reveal a slight linear relationship with much variation. Fitting a simple linear regression equation yields the following estimated relationship:

$$u_{2i} = -0.2671 + 0.0069 f_{1i} \quad \text{for} \quad i = 1, \ldots, 244.$$

The corresponding coefficient of determination is $R^2 = 0.1088$ and the correlation coefficient is $\rho = 0.3298$. Due to these very low values, it is unlikely that there is actually a linear relationship. In addition, there is no other obvious pattern in the scatterplot that might need to be examined. Therefore, it seems that a time shift in the demand is not due to higher temperature. This gets even clearer by splitting the data set into workdays and weekend days. If a shift in the bike demand depends on the overall temperature, this relationship also needs to arise when the same analysis is conducted by fitting new principal component and point process models on the workdays or the weekend days separately. The simple linear regressions between the two components yield for the workdays $R^2 = 0.028$ and $\rho = -0.1674$ and for the weekend data $R^2 = 0.0021$ and $\rho = 0.0455$. These results reinforce that there is no relationship between a demand shift for the bikes and a higher temperature.

Next, consider the data for station 166. The analysis is again based on the scatterplots of the component scores obtained by the different models. They are given in figure 8.

Both upper scatterplots as well as the lower right scatterplot show random scatters where

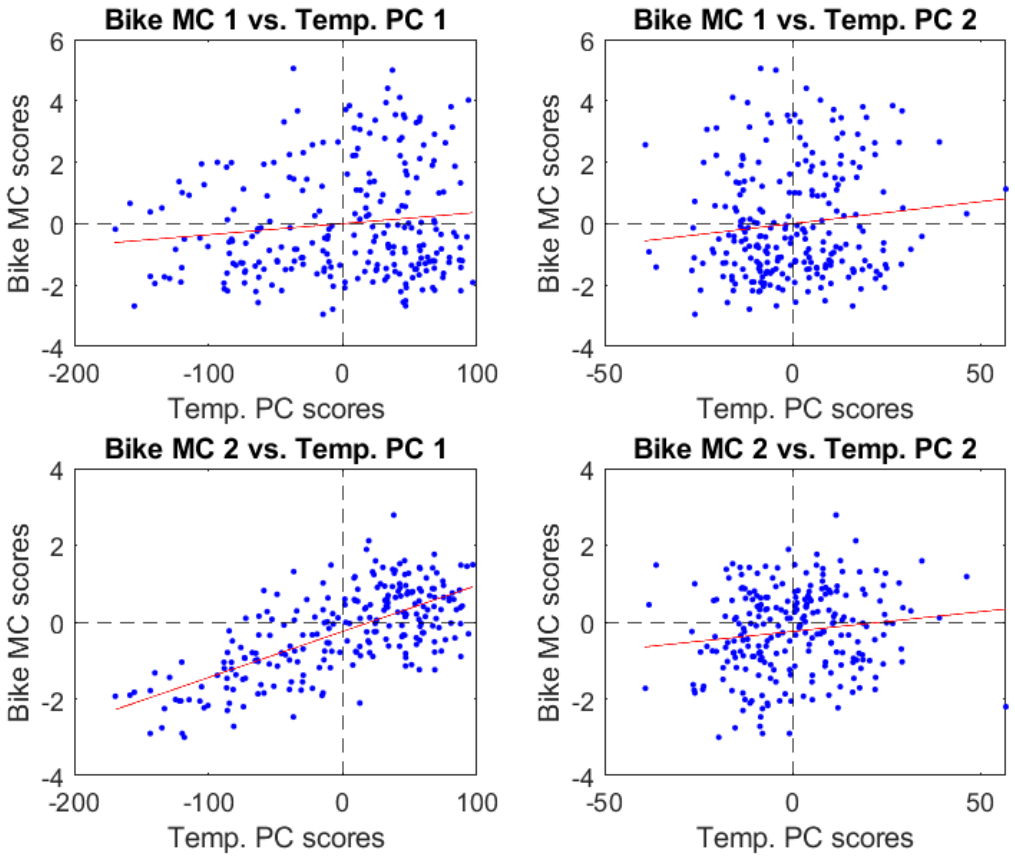

Figure 8: Station 166: Scatterplot of the component scores for the first bike MC and the first temperature PC (upper left plot). Scatterplot of the component scores for the first bike MC and the second temperature PC (upper right plot). Scatterplot of the component scores for the second bike MC and the first temperature PC (lower left plot). Scatterplot of the component scores for the second bike MC and the second temperature PC (lower right plot). All plots are supplemented by a fitted simple linear regression line.

no pattern is identifiable. Hence, no further analyses are conducted on these three component combinations. With the temporal PC 2 being a contrast component, station 166 gives the same conclusion as station 35 that there is no relationship between the bike demand and the amplitude between the daily maximum and minimum temperature. On the other hand, the upper left scatterplot illustrates the relationship between the shape of the bike demand and a higher level of temperature over the entire day. The data shows barely any connection. The fitted model

$$u_{1i} = -0.0493 + 0.0036 f_{1i} \quad \text{for} \quad i = 1, \ldots, 244$$

provides the statistical measures $R^2 = 0.0136$ and $\rho = 0.1165$ which confirm that there is most likely no relationship.

Now consider the only scatterplot with a obvious pattern. The lower left plot shows a linearly increasing relationship between the overall bike demand and the general temperature. Fitting a simple linear model gives

$$u_{2i} = -0.2503 + 0.0122 f_{1i} \quad \text{for} \quad i = 1, \ldots, 244.$$

Even if the corresponding coefficient of determination $R^2 = 0.4648$ and the correlation coefficient $\rho = 0.6817$ are smaller than for the corresponding components of station 35 ($R^2 = 0.5972$ and $\rho = 0.7728$), they confirm that a higher demand over the entire day is positively correlated with generally higher temperature.

6 Conclusions and outlook

The goal of this thesis was to examine the influence of temperature on the bike demand for the bike-sharing system Divvy in Chicago between April 1 and November 30 2016. For this purpose, functional temperature data for the city of Chicago was smoothed with a roughness penalty and the obtained curves were subject to a functional principal component analysis which revealed the two major types of variation. The most variability is due to a higher temperature level over the entire day in summer and generally colder days during spring and fall. The second principal component shows differently high amplitudes between the daily maximum and minimum temperatures where the amplitude oscillates over the entire time between April and November 2016.

On the other hand, the daily bike checkout times were assumed to be realizations of a doubly stochastic temporal point process and modeled by a multiplicative component model. Although two bike stations with presumably different user profiles were investigated, the estimation of the parameters provided for both stations two similar components that described the significant changes in the demand. In each case, the size component represents days with generally higher or lower demand which occur in summer respectively in spring and fall. The shift component shows for station 35 that the highest demand in summer takes place a little bit later during the day than usual whereas the shape of demand for station 166 changes frequently over the entire eight months.

The component scores of both analyses were then examined in scatterplots, simple linear regression models and statistical measures in order to find out what influence the temperature variation has on the bike demand. It turned out that the scores of the size components are highly positively correlated. That means, the overall demand is higher on days with a generally high temperature (i.e. in the summer). Moreover, the results showed no significant relationship between the other components. Thus, it seems that the period of high demand during a day is not influenced by any characteristic of the temperature.

However, as the time shift in demand showed minimal evidence of a correlation with the overall temperature level, it might be possible that the bike demand is influenced by a

factor which shows a comparable pattern in the component scores as the size component of the temperature. Therefore, further analysis could focus for example on the time of sunrise and sunset as the summer can again be distinguished from spring and fall in this regard. In addition, precipitation could also be taken into account to refine the prediction of bike usage.

7 References

[1] Borgnat, P., Robardet, C., Rouquier, J., Abry, P., Flandrin, P., and Fleury, E. (2011). *Shared bicycles in a city: A signal processing and data analysis perspective*. Advances in Complex Systems **14** 1-24.

[2] de Boor, C. (2001). *A Practical Guide to Splines*. Springer, New York, revised edition.

[3] Gervini, D. (2017). *Multiplicative component models for replicated point processes*. ArXiv 1705.09693.

[4] Møller, J., and Waagepetersen, R.P. (2004). *Statistical Inference and Simulation for Spatial Point Processes*. Chapman and Hall/CRC, Boca Raton.

[5] Ramsay, J. O. and Silverman, B. W. (2005). *Functional Data Analysis*. Springer, New-York, second edition.

Appendix MATLAB

The first MATLAB file *ThesisAnalysis.m* contains the code for the temperature analysis and the bike demand analysis. The second file *RegressionAnalysis.m* conducts the regression analysis between the component scores of the previous analyses. The file *TemperatureGCV.m* uses generalized cross-validation to find the optimal smoothing parameter for the temperature data. The last four files, *bspl.m*, *fpc.m*, *cv_mcatpp_cyc.m* and *mcatpp_cyc.m*, are functions that were all written and provided by Professor Daniel Gervini. *bspl.m* computes the B-spline basis function and their derivatives and evaluates them on a given time grid. *fpc.m* conducts the functional principal component analysis. *cv_mcatpp_cyc.m* finds the optimal smoothing parameters for the multiplicative component model. *mcatpp_cyc.m* ultimately estimates the multiplicative component model.

TemperatureAndBikeAnalysis.m

```
close all
clc

%%%%%%%Temperature PCA

%Consider the data for the temperature
data=xlsread('TemperatureChicago2016.xls');
x_temp=data';
%Original time grid
t =linspace(0,24,24)';
%Define knots
knots=linspace(min(t),max(t),10+2);
%Finer time grid
t2=linspace(min(t),max(t),200);
%Initialize vector for fitted values
```

```
xhat_temp=zeros(length(t2),length(x_temp));
%Evaluate basis function on time grid using the predefined function bspl
%(by Prof. Gervini)
Phi=bspl(t,4,knots,0);
%Evaluate basis function on a finer time grid
Phi2=bspl(t2,4,knots,0);
%%Evaluate second derivative of basis function on time grid for the
%roughness penalty
Phi_secder = bspl(t,4,knots,2);
%Optimal parameter for roughness penalty (obtained by GCV)
lmd_smo = 12.59;
%Roughness penalty
R = (Phi_secder'*Phi_secder)/length(t);

%Estimate values of x
for k=1:length(x_temp)
    %Estimate c. Therefore use the original time grid
    c = (Phi'*Phi+lmd_smo*R)^(-1)*Phi'*x_temp(:,k);
    %Estimate xhat with c (based on original time grid) and Phi2 (based on
    %finer time grid)
    xhat_temp(:,k) = Phi2*c;
end

%Compute principal components using the predefined function bspl
%(by Prof. Gervini)
[pc_temp,lmb_temp,s_temp] = fpc(t2,xhat_temp',5);

%Compute cumulative share of variability
impact_temp=cumsum(lmb_temp)/sum(lmb_temp);
```

```
%Plot results

%mean functions
mu_temp=mean(xhat_temp,2);

%PCs
figure(1)
for k=1:2
        subplot(1,2,k)
        plot(t2,pc_temp(k,:)')
        axis([0 24 -0.5 0.5])
        title([num2str(k),'. PC'])
end

%Mu+/-PC
figure(2)
for k=1:2
        subplot(1,2,k)
        plot(t2,mu_temp,t2,mu_temp+3*pc_temp(k,:)', 'red',
                t2,mu_temp-3*pc_temp(k,:)', 'yellow','linewidth',1)
        axis([0 24 55 73])
        title([num2str(k),'. PC'])
end

%PC Scores
figure(3)
t3=linspace(1, 244, 244);
for k=1:2
```

```
        subplot(1,2,k)
        plot(t3,s_temp(:,k)')
        axis([1 244 -190/k 190/k])
        title([num2str(k),'. PC scores'])
end

%%%%%%%Bike PCA

%Consider the data for the bike demand
data=load('station_35.txt');

%Split the data in vectors of checkout times for every day
x_bike=cell(1,244);
dates=data(:,1);
times=data(:,2);
for k = 1:244
    x_bike{1,k}=times(dates==k);
end

%Define a basis structure for the multiplicative component model
%range of data, order of B-splines, number of knots for B-spline
basis = struct('rng', [0 24], 'or', 4, 'nk', 10);
%Number of computed components
p=5;

%Find optimal smoothing parameter by cross-validation
%sm1=0.0316; %station 35 (mu)
%sm1=0.0010; %station 166 (components)
%sm2=0.1000; %station 35 (mu)
```

```
%sm2=0.1000; %station 166 (components)
[sm1,sm2,other] = cv_mcatpp_cyc(x_bike,basis,p);

%Apply multiplicative component model
[c0,C,s2,u,logf] = mcatpp_cyc(x_bike,basis,p,sm1,sm2,50);

%Define finer time grid on [0,24]
t3 = linspace(basis.rng(1),basis.rng(2),100);
%Define knots
knots = linspace(basis.rng(1),basis.rng(2),basis.nk+2);

%Evaluate basis function on time grid
B = bspl(t3,basis.or,knots,0);

%Compute cumulative share of variability
lmb_bike = s2;
impact_bike=cumsum(lmb_bike)/sum(lmb_bike);

%Plot results

%Estimated baseline intensity Lambda_0
figure(4)
lmb0 = exp(B*c0);
plot(t3,lmb0,'linewidth',2)

%Multiplicative components
figure(5)
plot(t3,exp(B*C),'linewidth',2)
```

```
%Baseline function multiplied with multiplicative components
figure(6)
subplot(1,2,1)
lmbplus = exp(B*c0+3*B*C(:,1));
lmbmin = exp(B*c0-3*B*C(:,1));
plot(t3,lmb0,t3,lmbplus,'red',t3,lmbmin,'yellow','linewidth',1)
axis([0 24 0 7])
title(['1. MC'])

subplot(1,2,2)
lmb0 = exp(B*c0);
lmbplus = exp(B*c0+3*B*C(:,2));
lmbmin = exp(B*c0-3*B*C(:,2));
plot(t3,lmb0,t3,lmbplus,'red',t3,lmbmin,'yellow','linewidth',1)
axis([0 24 0 7])
title(['2. MC'])

%PC Scores
figure(8)
t4=linspace(1, 244, 244);
for k=1:2
        subplot(1,2,k)
        plot(t4,u(:,k)')
        axis([1 244 -20/k^2 20/k^2])
        title([num2str(k),'. MC scores'])
end

%Isolate PC scores for temperature and bike demand for regressions analysis
tempscore_1 = s_temp(:,1);
```

```
tempscore_2 = s_temp(:,2);


bikescore_1 = u(:,1);
bikescore_2 = u(:,2);
```

RegressionAnalysis.m

```
%First two PCs Bike Demand
%First two PCs Temperature
%Linear Regression
figure(9)
subplot(2,2,1)
[fitpoly11, gof11]=fit(tempscore_1,bikescore_1,'poly1')
plot(fitpoly11,tempscore_1,bikescore_1)
yline(0, 'LineStyle', '--', 'LineWidth', 0.2, 'color', 'black')
xline(0, 'LineStyle', '--', 'LineWidth', 0.2, 'color', 'black')
xlabel('Temp. PC scores')
ylabel('Bike MC scores')
legend('off')
R11 = corrcoef(bikescore_1,tempscore_1);
R11(2, 1)
title('Bike MC 1 vs. Temp. PC 1')


subplot(2,2,2)
[fitpoly12, gof12]=fit(tempscore_2,bikescore_1,'poly1')
plot(fitpoly12,tempscore_2,bikescore_1)
yline(0, 'LineStyle', '--', 'LineWidth', 0.2, 'color', 'black')
xline(0, 'LineStyle', '--', 'LineWidth', 0.2, 'color', 'black')
xlabel('Temp. PC scores')
ylabel('Bike MC scores')
```

```
legend('off')
R12 = corrcoef(bikescore_1,tempscore_2);
R12(2, 1)
title('Bike MC 1 vs. Temp. PC 2')

subplot(2,2,3)
[fitpoly21, gof21]=fit(tempscore_1,bikescore_2,'poly1')
plot(fitpoly21,tempscore_1,bikescore_2)
yline(0, 'LineStyle', '--', 'LineWidth', 0.2, 'color', 'black')
xline(0, 'LineStyle', '--', 'LineWidth', 0.2, 'color', 'black')
xlabel('Temp. PC scores')
ylabel('Bike MC scores')
legend('off')
R21 = corrcoef(bikescore_2,tempscore_1);
R21(2, 1)
title('Bike MC 2 vs. Temp. PC 1')

subplot(2,2,4)
[fitpoly22, gof22]=fit(tempscore_2,bikescore_2,'poly1')
plot(fitpoly22,tempscore_2,bikescore_2)
yline(0, 'LineStyle', '--', 'LineWidth', 0.2, 'color', 'black')
xline(0, 'LineStyle', '--', 'LineWidth', 0.2, 'color', 'black')
xlabel('Temp. PC scores')
ylabel('Bike MC scores')
legend('off')
R22 = corrcoef(bikescore_2,tempscore_2);
R22(2, 1)
title('Bike MC 2 vs. Temp. PC 2')
```

TemperatureGCV.m

```
close all
clc

%Consider the data for the temperature
data=xlsread('TemperatureChicago2016.xls');
x=data';

%Time grid for 24 hours
t =linspace(0,24,24)';
%Initialize vector for fitted values and variances
xhat = zeros(24,length(x));
sigmasqrhat = zeros(244,1);

%Define knots
knots=linspace(min(t),max(t),10+2);

%Evaluate basis function on time grid using the predefined function bspl
%(by Prof Gervini)
Phi_orig = bspl(t,4,knots,0);
%%Evaluate second derivative of basis function on time grid for the
%roughness penalty
Phi_secder = bspl(t,4,knots,2);
%Roughness penalty
R = (Phi_secder'*Phi_secder)/length(t);

%Start loop for the different values of lambda
range=10.^(0:.1:25);
count=1;
```

```
GCV=0;
for lambda=range
   S_lambda=Phi_orig*(Phi_orig'*Phi_orig+lambda*R)^(-1)*Phi_orig';
   xhat=S_lambda*x;
   GCV(count)=24*norm(x-xhat,2)^2/(24-trace(S_lambda))^2;
   count=count+1;
end

%Plot the GCV
plot1 = figure;
loglog(range,GCV)
legend('GCV(\lambda)')

%Find lambda that minimizes the GCV
minimum=find(GCV==min(GCV));
lambda=range(minimum);
sprintf('The Minimum is at lambda=%f',lambda)

%Compute the estimates using the optimal lambda
S_lambda=Phi_orig*(Phi_orig'*Phi_orig+lambda*R)^(-1)*Phi_orig';
xhat=S_lambda*x;

%Plot data against the estimates of the first ten days as example
plot2 = figure;
plot(t,x(:,1:10),'.',t,xhat(:,1:10))
legend('data',sprintf('x(t)',lambda))
```

bspl.m

```
function y = bspl(x,k,t,r)
```

```
%function y = bspl(x,k,t,r)
%
%B-spline basis functions and their derivatives
%
%INPUT:
%   x   (m x 1 or 1 x m)    Input grid.
%   k   (scalar)             Spline order.
%   t   (n x 1 or 1 x n)    Knots, must be a strictly increasing sequence
%                              and must INCLUDE interval endpoints.
%   r (scalar)             Order of derivative.
%
%OUTPUT:
%   y   (m x n+k-2)          Basis function (or derivative) values at X
%


if nargin<4
    error('Not enough input arguments')
end
if size(t,1)>1
    t = t';
end

m = length(x);
n = length(t);
y = zeros(m,n+k-2);

if r==0
```

```
    t = [repmat(t(1),1,k-1), t, repmat(t(n),1,k-1)];
    n = length(t);
    b = zeros(1,k);
    for l = 1:m
        b(1) = 1;
        i = max(find(t<=x(l)));
        if i==n, i = n-k; end
        for j = 1:k-1
            dr(j) = t(i+j)-x(l);
            dl(j) = x(l)-t(i+1-j);
            saved = 0;
            for r = 1:j
                term = b(r)/(dr(r)+dl(j+1-r));
                b(r) = saved + dr(r)*term;
                saved = dl(j+1-r)*term;
            end
            b(j+1) = saved;
        end
        y(l,i-k+1:i) = b;
    end

else

tt = [repmat(t(1),1,k-2), t, repmat(t(n),1,k-2)];
    B = bspl(x,k-1,t,r-1);
msp = ((k-1)./(ones(m,1)*(tt(k:n+2*(k-2))-tt(1:n+k-3)))).*B;
y(:,1) =  - msp(:,1);
y(:,2:n+k-3) = msp(:,1:n+k-4) - msp(:,2:n+k-3);
```

```
y(:,n+k-2) =  msp(:,n+k-3);


end
```

fpc.m

```
function [pc,lmb,s] = fpc(t,x,q)
%[pc,lmb,s] = fpc(t,x,q)
%Functional Principal Components (input data must be smooth, on a fine
%time grid)
%Input
%   t: Time grid (1 x m)
%   x: Curves (n x m)
%   q: Number of PCs to estimate (scalar)
% Output
%   pc:  Principal components (q x m)
%   lmb: Eigenvalues (q x 1)
%   s:   Component scores (n x q)
%


% Input check
[n,m] = size(x);
t = t(:)';
if length(t)~=m
    error('Dimensions of T and X incompatible')
end


% Gram matrix
x = x-repmat(mean(x),[n 1]);     %%%% NOTE: X is centered from now on
ht = [t(2)-t(1),(t(3:m)-t(1:m-2))/2,t(m)-t(m-1)];
```

```
G = (x.*repmat(ht,[n 1]))*x';


% PC computation
[Wstar,Lstar] = svd(G);
lmb = diag(Lstar(1:q,1:q))/n;
W = Wstar(:,1:q)*diag(1./sqrt(n*lmb(1:q)));
pc = W'*x;
s = (x.*repmat(ht,[n 1]))*pc';
```

cv_mcatpp_cyc.m

```
function [optsm1,optsm2,other] = cv_mcatpp_cyc(x,basis,p)
% [optsm1,optsm2,other] = cv_mcatpp_cyc(x,basis,p)
%
% Cross-validation search of smoothing parameters for temporal MCA
%   with cyclic border condition
%  (Five-fold cross-validation is used)
%
% INPUT:
%   x: Observed time points (n x 1 cell).
%      Each x{i} is a vector containing the data from replication i.
%   basis: B-spline basis parameters. Struct with the following fields:
%       rng: Time range (1 x 2 vector).
%       or: Spline order (integer; 4 is cubic splines).
%       nk: Number of knots (integer). Knots will be equally spaced.
%   p: Number of model components (integer>=0).
%
% OUTPUT:
%   optsm1: Optimal smoothing parameter for the mean (scalar>=0)
%   optsm2: Optimal smoothing parameter for components (scalar>=0)
```

```
%            (Returns [] if p=0).
%   other: Additional output. Struct with the following fields:
%       optOF1: Optimal value of the objective function at optsm1.
%       optOF2: Optimal value of the objective function at optsm2.
%                (Returns [] if p=0).
%       smgrid: Grid of smoothing parameters used.
%       OF1grid: Objective function at smgrid for mean-only models.
%       OF2grid: Objective function at smgrid for p-component models.
%                 (Returns [] if p=0).
%
% Programs called: MCATPP_CYC, PRED_MCATPP
%
% Version: June 2018

% Cross-validation
itmax = 10;
smgrid = 10.^(-7:.5:-1)';
Ng = length(smgrid);
% Find optimal sm for mean
disp('Finding optimal sm1')
OF1grid = -Inf(Ng,1);
optOF1 = -Inf;
optsm1 = -Inf;
for i = 1:Ng
    disp(['Cross-validating for grid
                point ' num2str(i) ' of ' num2str(Ng)])
    logf = cv_5f(x,basis,0,smgrid(i),0,itmax);
    OF1grid(i) = mean(logf);
    if OF1grid(i)>optOF1
```

```
        optOF1 = OF1grid(i);
        optsm1 = smgrid(i);
    end
end
% Find optimal sm for components
disp(' ')
disp('Finding optimal sm2')
if p==0
    optOF2 = [];
    optsm2 = [];
    OF2grid = [];
else
    OF2grid = -Inf(Ng,1);
    optOF2 = -Inf;
    for i = 1:Ng
        disp(['Cross-validating for grid
                    point ' num2str(i) ' of ' num2str(Ng)])
        logf = cv_5f(x,basis,p,optsm1,smgrid(i),itmax);
        OF2grid(i) = mean(logf);
        if OF2grid(i)>optOF2
            optOF2 = OF2grid(i);
            optsm2 = smgrid(i);
        end
    end
end
% Output
other = struct('optOF1',optOF1,'optOF2',optOF2,'smgrid',smgrid,...
    'OF1grid',OF1grid,'OF2grid',OF2grid);
end
```

```
%%%% -------- AUXILIARY FUNCTIONS


function logf = cv_5f(x,basis,p,sm1,sm2,itmax)
% Five-fold cross-validation for MCATPP
% Computes cross-validated log-densities
% Programs called: MCATPP_CYC, PRED_MCATPP
n = length(x);
logf = -Inf(n,1);
B = round(n/5);
for i = 1:5
    itest = ((i-1)*B+1):(i*B);
    if i==5
        itest = ((i-1)*B+1):n;
    end
    itrain = setdiff(1:n,itest);
    try
        [T,c0,C,s2] = evalc('mcatpp_cyc(x(itrain),basis,p,sm1,sm2,itmax)');
    catch ME1
        c0 = [];
        C = [];
        s2 = [];
    end
    if ~isempty(c0)
        [T,logf(itest)] = evalc('pred_mcatpp(x(itest),basis,c0,C,s2)');
    end
end
end
```

mcatpp_cyc.m

```
function [c0,C,s2,u,logf] = mcatpp_cyc(x,basis,p,sm1,sm2,itmax)
% [c0,C,s2,u,logf] = mcatpp_cyc(x,basis,p,sm1,sm2,itmax)
%
% Multiplicative Component Analysis for Temporal Point Processes
%   (PCA of log-intensities) with cyclic border condition
%
% INPUT:
%   x: Observed time points (n x 1 cell).
%      Each x{i} is a vector containing the data from replication i.
%   basis: B-spline basis parameters. Struct with the following fields:
%       rng: Time range (1 x 2 vector).
%       or: Spline order (integer; 4 is cubic splines).
%       nk: Number of knots (integer). Knots will be equally spaced.
%   p: Number of model components (integer>=0).
%   sm1: Smoothing parameter for the mean (scalar>=0).
%   sm2: Smoothing parameter for the components (scalar>=0).
%        (All components have norm 1 but the mean does not, so different
%         sm's may be needed to attain the same degree of smoothness).
%   itmax: Maximum number of iterations (integer).
%
% OUTPUT:
%   c0: Mean basis coefficients (q x 1).
%   C: Component basis coefficients (q x p).
%   s2: Component variances (p x 1).
%   u: Individual component scores (n x p).
%   logf: Individual log-densities (n x 1).
%
% External calls: BSPL
```

```
%
% Version: June 2018

% Input check
c0 = [];
C = [];
s2 = [];
u = [];
logf = [];
if ~iscell(x)
    disp('Error: X must be cell array')
    return
else
    [mx,nx] = size(x);
    if (mx>1 && nx>1)
        disp('Error: X must be a one-dimensional cell array')
        return
    end
end
n = length(x);

% Data filtering (elimination of data outside RNG)
m = zeros(n,1);
a = basis.rng(1);
b = basis.rng(2);
for i = 1:n
    x{i}(x{i}<a) = [];
    x{i}(x{i}>b) = [];
    m(i) = length(x{i});
```

```
end
if any(m==0)
    disp('Warning: Some x{i}s have no data within basis range')
end
if any(m>=200)
    disp('Warning: Some x{i}s have more than 200 observations')
    disp('This may cause Inf values in the likelihood function')
    disp('This method is intended for relatively small x{i}s')
    disp('For large x{i}s you can just use kernel smoothing')
end

% Initialization
q = basis.or + basis.nk;
t = linspace(a,b,300);
dt = t(2)-t(1);
knt = linspace(a,b,basis.nk+2);
B0 = bspl(t,basis.or,knt,0);
J0 = (B0'*B0)*dt;
if basis.or>2
    B2 = bspl(t,basis.or,knt,2);
    J2 = (B2'*B2)*dt;
else
    J2 = zeros(q,q);
end
sumB = zeros(n,q);
for i = 1:n
    B_i = bspl(x{i},basis.or,knt,0);
    sumB(i,:) = sum(B_i,1);
end
```

```
Bab0 = bspl([a b],basis.or,knt,0);
Bab1 = bspl([a b],basis.or,knt,1);
Cyc = [Bab0(1,:)-Bab0(2,:); Bab1(1,:)-Bab1(2,:)];
Pcyc = null(Cyc);


% -----> Initial mean-only model
c0 = (Pcyc*Pcyc')*log(mean(m)/(b-a))*ones(q,1);
logf = complogf0(sumB,c0,dt,B0,m);
OF = mean(logf)-sm1*c0'*J2*c0;
disp('---> Computing mean')
disp(['Iteration: 0, Pen. loglik: ' num2str(OF)])
err = 1;
iter = 0;
while err>1e-3 && iter<itmax
    iter = iter + 1;
    c00 = c0;
    OF0 = OF;
    [gc,Hc] = derivc0(sumB,c0,dt,B0);
    gpll = gc-2*sm1*J2*c0;
    Hpll = Hc-2*sm1*J2;
    direction = Pcyc*((Pcyc'*Hpll*Pcyc)\(Pcyc'*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*c0'*J2*c0;
```

```
        k = k+1;
    end
    % Stopping criterion
    if OF<=OF0 || ~all(isfinite(c0))
        disp('No further improvement in obj. func. is possible')
        c0 = c00;
        OF = OF0;
    end
    err = norm(c0-c00)/norm(c00);
    disp(['Iteration: ' num2str(iter) ', Pen. loglik: ' ...
            num2str(OF) ', Error: ' num2str(err)])
end

% -----> Sequential PC estimation
C = zeros(q,p);
s2 = zeros(p,1);
u = zeros(n,p);
u2 = zeros(n,p);
for ic = 1:p
    if ic==1
        P = Pcyc;
    else
        P = null([Cyc; C(:,1:ic-1)'*J0]);
    end
    % Initial estimators
    C(:,ic) = (P*P')*ones(q,1);
    C(:,ic) = C(:,ic)/sqrt(C(:,ic)'*J0*C(:,ic));
    if ic==1
        Ilmb0 = sum(exp(B0*c0))*dt;
```

```
    u(:,ic) = sqrt(b-a)*log(max(m,1)/Ilmb0);
    s2(ic) = var(u(:,ic));
else
    s2(ic) = s2(ic-1)/2;
end
[u(:,1:ic),u2(:,1:ic),logf] = ...
    compeff(sumB,c0,C(:,1:ic),s2(1:ic),dt,B0,m,u(:,1:ic));
OF = mean(logf)-sm1*c0'*J2*c0-sm2*sum(diag(C(:,1:ic)'*J2*C(:,1:ic)));
disp(['---> Computing component ' num2str(ic)])
disp(['Iteration: 0, Pen. loglik: ' num2str(OF)])
err = 1;
iter = 0;
% Iterations
while err>1e-3 && iter<itmax
    iter = iter + 1;
    % Update C
    c00 = C(:,ic);
    u00 = u;
    u200 = u2;
    OF0 = OF;
    [gc,Hc] = derivc(sumB,c0,C(:,1:ic),dt,B0,u(:,1:ic),u2(:,1:ic));
    gpll = gc-2*sm2*J2*C(:,ic);
    Hpll = Hc-2*sm2*J2;
    direction = P*((P'*Hpll*P)\(P'*gpll));
    OF = -Inf;
    k = 0;
    while OF<=OF0 && k<6
        step = 0.7^k;
        C(:,ic) = c00-step*direction;
```

```
            C(:,ic) = C(:,ic)/sqrt(C(:,ic)'*J0*C(:,ic));
            [u(:,1:ic),u2(:,1:ic),logf] = ...
                compeff(sumB,c0,C(:,1:ic),s2(1:ic),dt,B0,m,u00(:,1:ic));
            OF = mean(logf) - sm1*c0'*J2*c0 ...
                -sm2*sum(diag(C(:,1:ic)'*J2*C(:,1:ic)));
            k = k+1;
        end
        if OF<=OF0 || ~all(isfinite(C(:,ic)))
            disp('No further improvement in obj. func. is possible')
            C(:,ic) = c00;
            u = u00;
            u2 = u200;
        end
        % Update s2
        s2 = mean(u2,1)';
        % Stopping criterion
        err = norm(C(:,ic)-c00)/norm(c00);
        disp(['Iteration: ' num2str(iter) ', Pen. loglik: ' ...
            num2str(OF) ', Error: ' num2str(err)])
    end
end
end

%%%%%%------ AUXILIARY FUNCTIONS

function logf = complogf0(sumB,c0,dt,B0,m)
%Computes log-densities for mean-only model
logf = -sum(exp(B0*c0))*dt + sumB*c0 - gammaln(m+1);
end
```

```
function [gc0,Hc0] = derivc0(sumB,c0,dt,B0)
% Derivatives of loglik/n w.r.t c0
q = size(B0,2);
Hc0 = -(B0'*((exp(B0*c0)*ones(1,q)).*B0))*dt;
gc0 = -B0'*exp(B0*c0)*dt + mean(sumB,1)';
end

function [u,u2,logf] = compeff(sumB,c0,C,s2,dt,B0,m,u_ini)
%Computes random effects and log-pdf using Laplace approximation
[n,p] = size(u_ini);
Phi = B0*C;
logf = zeros(n,1);
u = u_ini;
u2 = u_ini.^2;
for i = 1:n
    % Compute log(f(x))
    uL = u_ini(i,:);
    D_gi = zeros(1,p);
    H_gi = eye(p);
    for steps = 1:5
        uL = uL - D_gi/H_gi;
        lmbi = exp(B0*c0+B0*C*uL');
        D_gi = -lmbi'*Phi*dt + sumB(i,:)*C - uL./s2';
        H_gi = -Phi'*((lmbi*ones(1,p)).*Phi)*dt - diag(1./s2);
    end
    gi = -sum(lmbi)*dt + sumB(i,:)*(c0+C*uL') - gammaln(m(i)+1) ...
        -sum(uL.^2./(2*s2')) - .5*sum(log(2*pi*s2));
    logf(i) = gi + (p/2)*log(2*pi) - .5*logdet(-H_gi);
```

```
    S = (-H_gi)\eye(p);
    u(i,:) = uL;
    u2(i,:) = diag(S)' + uL.^2;
end
end

function [gc,Hc] = derivc(sumB,c0,C,dt,B0,u,u2)
% Derivatives of loglik/n w.r.t C(:,end)
% Uses ad-hoc approx of second derivatives and plug-in scores for integrals
[n,p] = size(u);
q = length(c0);
ng = size(B0,1);
lmb = exp(B0*c0*ones(1,n)+B0*C*u');
ulmb = (ones(ng,1)*u(:,p)').*lmb;
u2lmb = (ones(ng,1)*u2(:,p)').*lmb;
gc = (-B0'*mean(ulmb,2))*dt + (sumB'*u(:,p)/n);
Hc = -(B0'*((mean(u2lmb,2)*ones(1,q)).*B0))*dt;
end

function y = logdet(A)
% log(det(A)) for symmetric non-neg A
R = chol(A);
y = 2*sum(log(diag(R)));
end
```