POLITECNICO DI TORINO

MASTER's Degree in Data Science Engineering

MASTER'S Thesis

A Nearest Neighbors Gaussian Process Approach to Modeling Customer Electricity Consumption Profiles

Supervisors

Prof. Enrico BIBBONA

Prof. Gianluca MASTRANTONIO

Candidate

Ilaria ZERBINI

April 2025

Summary

This thesis addresses the modeling of customer energy consumption profiles within an energetic community by leveraging data analysis techniques and Nearest Neighbors Gaussian Processes (NNGP). A comprehensive dataset of energy consumption was initially analyzed to extract insights and identify inherent patterns, including short-term correlations, 24-hour cyclic behaviors, and weekly periodicities.

A model was developed that incorporates these multi-scale repetitive patterns and whose formulation is designed to describe the temporal dynamics of the consumption data, by accounting for both immediate sample dependencies and longer cyclic trends.

To estimate the model parameters, an NNGP framework was employed, offering a computationally efficient approach for simulation and inference. The performance of the proposed model was validated by comparing its outputs with synthetic data specifically generated to mimic the observed patterns, as well as against the real dataset.

The results show that the NNGP-based model effectively captures the essential features of customer energy consumption.

Acknowledgements

I would like to express my heartfelt gratitude to my family and friends for their unwavering support and encouragement throughout this journey.
Their patience, understanding, and belief in me have been invaluable.
I am also deeply thankful to everyone who has supported me along the way, whether through guidance, collaboration, or simply by offering words of motivation.
Your contributions have been truly meaningful and greatly appreciated.

Table of Contents

Chapter 1

Introduction

In order to facilitate the comprehension of this work, it is convenient to provide a brief introduction to the fundamental concepts that will be covered throughout the latter.
In particular, the two key concepts on which all this study revolves around are probability and time series analysis.
For this reason, the following introductory section is divided into two parts, each focusing on one of these topics.

1.1 Probability Background

The material presented in this section follows [1].

1.1.1 Set Theory

Definition 1 *The set S of all the possible outcomes of a particular experiment si called* ***sample space*** *for the experiment.*

As an example, consider the toss of a coin as experiment, in this case the sample space consists of only two outcomes:

$$S = \{Heads, Tails\}$$

There are further classification of sample space, but for the work presented in this thesis it is enough to highlight that it can be either **countable**, if its elements can be put in a 1-1 correspondence with a subset of the integers

numbers, or else **uncountable**.

Definition 2 *An **event** is any collection of possible outcome of an experiment. Therefore any subset of S, including itself, it called event.*

Now, let $A \subseteq S$ be a certain event. A is said to occur if the outcome of the experiment is in the latter.
Moreover, it is also possible to establish an ordering between events.
The following two relationships hold and allow to define it:

Definition 3 *Let $B, A \subseteq S$ be two events.*
The following two relationships hold:

- ***containment**: $A \subset B \iff x \in A \implies x \in B$*
- ***equality**: $A = B \iff A \subset B$ and $B \subset A$*

Moreover, for any two events both intersection and union set properties, still hold:

Definition 4 *Let $B, A \subseteq S$ be two events.*
*The **intersection** of two events, denoted $A \cap B$, is the set of outcomes that belong to both A and B:*

$$A \cap B = \{x \mid x \in A \text{ and } x \in B\}. \tag{1.1}$$

*Meanwhile, the **union** of two events, denoted $A \cup B$, is the set of outcomes that belong to at least one of the events:*

$$A \cup B = \{x \mid x \in A \text{ or } x \in B\}. \tag{1.2}$$

*Finally, the **complement** of A, written A^c, is the set of all the outcomes that are not in A:*

$$A^c = \{x \mid x \notin A\}$$

Last definition in this section is **disjoint events**:

Definition 5 *Let $B, A \subseteq S$ be two events.*
A and B are said to be disjoint, or mutually exclusive, events if

$$A \cap B = \emptyset$$

Moreover, let $A_i \subseteq S$, with $i = 1, \ldots, N$, be a series of events. Such events are said to be pairwaise disjoint if

$$A_i \cap A_j = \emptyset, \ \forall \ i \neq j$$

1.1.2 Definition of Probability

When an experiment is performed, its realization is one of the possible outcomes in the sample space.
If such experiment is then repeated a number of times, it could happen that some of the outcomes will show up again. The frequency of occurrence of each outcomes, represents its **probability**.
Therefore, probability is a measure that quantifies the likelihood of a given outcome when it is not yet known whether the event will happen or not. This likelihood is represented by a number between 0 and 1.

Although, to give a more accurate and mathematical definition, it is necessary to introduce the concept of **sigma algebra**, even if just briefly since it will not be focal for this work:

Definition 6 *A collection of subsets of the sample space S is called sigma algebra, denoted by $\mathcal{B}$, if it satisfies the following three properties:*

- $\emptyset \in \mathcal{B}$
- *If $A \in \mathcal{B}$ then $A^c \in \mathcal{B}$*
- *If $A_1, A_2, ... \in \mathcal{B}$, then $\cup_{i=1}^{\infty} A_i \in \mathcal{B}$*

Definition 7 *Given a sample space S and an associated sigma algebra $\mathcal{B}$, P is a probability function defined on $\mathcal{B}$ if it satifies:*

- $P(A) \geq 0$, $\forall A \in \mathcal{B}$

- $P(S) = 1$
- *If* $A_1, A_2, \cdots \in \mathcal{B}$ *are pairwise disjoint* $\implies$ $P(\cup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} P(A_i)$

These three properties are usually referred to as the **Axiom of probability**. It must be noted that the probability function isn't unique, as multiple different ones can exist for each sample space S.

1.1.3 Conditional Probability

Conditional probability allows us to update our beliefs about the occurrence of an event, given that another event has already occurred, playing a fundamental role in statistical inference and decision-making.

Definition 8 *Let A and B be events in a given sample space S, and assume that $P(B) > 0$.*
Then, the conditional probability of A given B is denoted by $P(A|B)$ and is defined as

$$P(A|B) = \frac{P(A \cap B)}{P(B)}, \tag{1.3}$$

In particular, when computing $P(A|B)$ the sample space S is "updated", and effectively becomes equal to B, so that $P(B|B) = 1$. Therefore, if $A \cap B = \emptyset$, then it follows that $P(A|B) = P(B|A) = 0$.

An important result, called Bayes' Rule, allows a rewriting of conditional probability:

Definition 9 *Let A and B be events in a given sample space S, and assume that $P(B) > 0$. Then the following holds:*

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{1.4}$$

The oroginal formulation extends this result to a generic partition of the sample space S.
Let $A_1, A_2, \ldots$ be a partition of the sample space S, and let B be any set. Then, for each $i = 1,2, \ldots$ it holds:

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{\sum_{j=1}^{\infty} P(B|A_j)P(A_j)} \tag{1.5}$$

As it can occur that two events result to be linked in some way, it can also happen the opposite, so that the particular occurrence of a certain event B doesn't impact at all another event A belonging to the same sample space. Considering the latter situation in terms of conditional probability, these two events are such that $P(A|B) = P(A)$.
Leveraging Bayes' Rule (1.5) introduce above, in this case it holds:

- $P(B|A) = P(A|B)\frac{P(B)}{P(A)} = P(A)\frac{P(B)}{P(A)} = P(B)$
- $P(A \cap B) = P(A)P(B)$

In the situation described, A and B are called **independent events**:

Definition 10 *Two events, A and B, are statistically independed if*

$$P(A \cap B) = P(A)P(B)$$

1.1.4 Random Variables

In probability theory, it is often useful to represent uncertain outcomes using numerical values rather than dealing directly with complex sample spaces. A **random variable** provides a structured way to quantify randomness by mapping elements of a sample space to real numbers:

Definition 11 *A random variable is a measurable function from a sample space S into the real numbers*

Moreover, to each random variable X it is associated a **cumulative distribution function**:

Definition 12 *The cumulative distribution function (cdf) of a random variable X, denoted by $F_X(x)$ is defined by:*

$$F_X(x) = P_X(X \leq x), \quad \forall x$$

where P_X is a probability function defined on X as:

$$P_X(X = x_i) = P(\{s_j \in S | X(s_j) = x_i\})$$

The cdf can be either continuous or discontinuous and has to satisfy three conditions:

Definition 13 *The function F(x) is a cdf* $\iff$ *the three following properties hold:*

- $\lim_{x\to-\infty} F(x) = 0$ *and* $\lim_{x\to\infty} F(x) = 1$
- *F(x) is a nondecreasing fucntion of x*
- *F(x) is right continuous* $\implies \lim_{x\downarrow x_0} F(x) = F(x_0)$

If the cdf of a given random variable X is continuous, then X is a continuous random variable, else it is a discrete random variable.
A random variable is completely defined by its cdf, stating its probability distribution.

Another important function is associated with a random variable and its cdf: the **probability density function** (pdf).

Definition 14 *The probability density function of a discrete random variable* X *is defined as:*

$$f_X(x) = P(X = x), \ \forall x$$

Meanwhile for continuous random varibles, the pdf is the function that satisfies:

$$F_X(x) = \int_{-\infty}^{x} f_X(t)\, dt \quad \textit{for all } x.$$

Moreover, there are two requirements necessary for a pdf to be defined so, which are direct consequencies of its definition:

Definition 15 *A function* $f_X(x)$ *is a pdf of a random variable* x $\iff$

- $f_X(x) \geq 0$ *for all* x
- $\sum_x f_X(x) = 1$ (*discrete*) *or* $\int_{-\infty}^{\infty} f_X(x)\, dx = 1$ (*continuous*)

The pdf is usually used to specify the probability of X falling within a particular range of values $[a, b]$, more than taking one particular value.
Such probability corresponds to the integral of $f(X)$ over the desired range, therefore it's the area underneath the curve described by $f(X)$:

$$P(a \leq X \leq b) = \int_a^b f(x)\, dx.$$

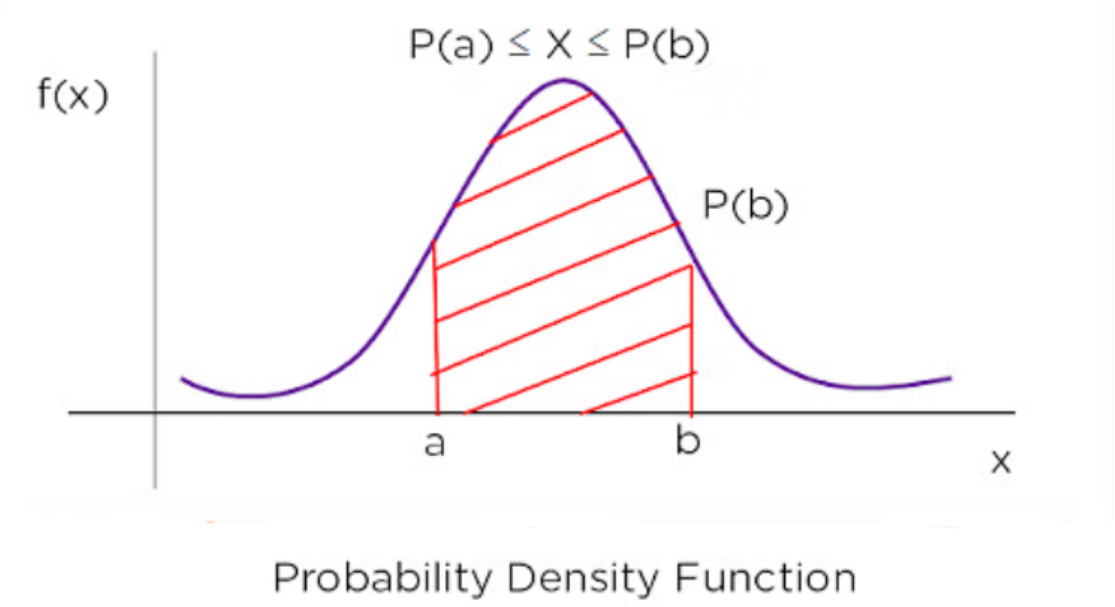


Figure 1.1: A probability density function illustrating the probability of a random variable X lying between the value a and b. The shaded region under the curve corresponds to $P(a \leq X \leq b)$ [2]

Since the probability density function of the Normal, or Gaussian, distribution is particularly relevant to this thesis, its pdf is also reported:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right),$$

where μ and σ^2 are taken to be the mean and variance, which will be briefly described in the following section.

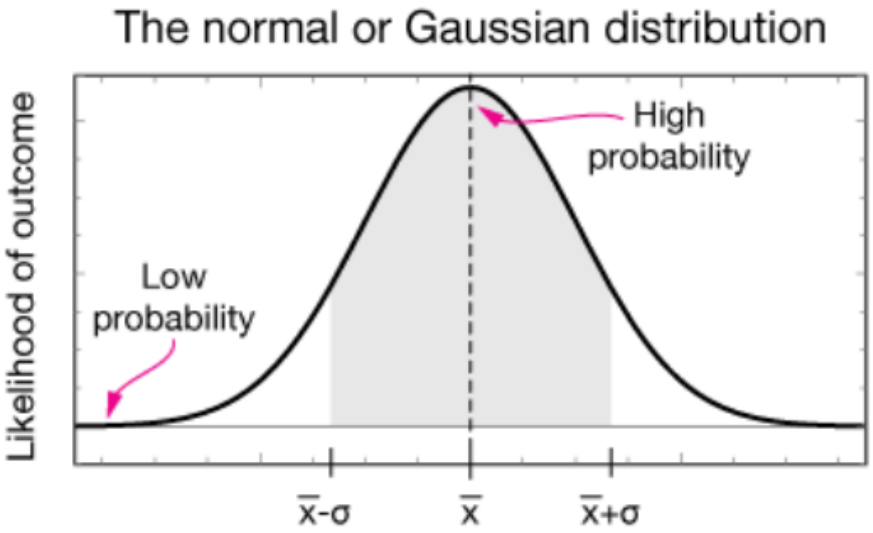


Figure 1.2: The Gaussian curve shows that the most-likely value is the mean, $\bar{x}$. Two other values have been defined, $\bar{x} \pm \sigma$, where σ is the standard deviation and σ^2 is called the "variance "[3]

1.1.5 Mean and Variance

Two important tools used to describe and characterize the PDFs are **mean** and **variance**.
The first one, also known as average, is the long-run arithmetic average value of a random variable having that distribution.
Given a random variable X, then its mean is also known as the expected value of X, denoted $E(X)$.
For a continuous distribution, the mean is defined as:

$$E(X) = \int_{-\infty}^{\infty} x f(x)\, dx$$

where $f(x)$ is the probability density function.
The variance instead, is the expected value of the squared deviation from

the mean of a random variable. This means that is a measure of dispersion, which measures how far the set of possible outcomes is spread out from their average value.

$$\text{Var}(X) = E[(X - \mu)^2]$$

A disadvantage of the variance for practical applications is that its unit of measure differ from the random variable's one. For this reason, usually instead of the variance it is used the **standard deviation**, which is obtained by doing the square root of the variance.
Another disadvantage is that the variance is not finite for many distributions.

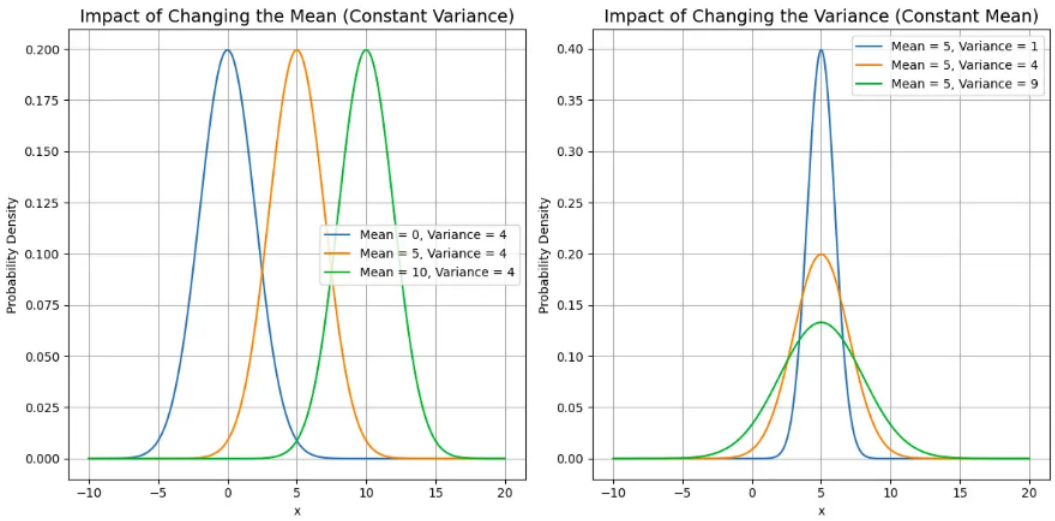


Figure 1.3: Impact of changing the mean and variance of a normal distribution. The left plot shows how shifting the mean while keeping the variance constant affects the location of the distribution. The right plot demonstrates how increasing the variance while keeping the mean constant results in a wider, more spread-out distribution[4]

1.1.6 The Likelihood Function

Another key concept in statistics is the **likelihood function**.
The likelihood function measures how well a statistical model explains observed samples by calculating the probability of seeing that outcome setting a different value for a parameter in the model.

Let $X = (X_1, X_2, ..., X_n)$ be a vector of random variables with a joint probability density function $f(X; \theta)$, where $\theta \in \Theta$ denotes a parameter or a vector of parameters of f.
If the value of the vector θ is fixed, another way to express the pdf is:

$$X \mapsto f(X \mid \theta),$$

Following the same notation, but fixing the vector of observed data $x = x_1, .., x_n$ instead, the likelihood function is thus obtained:

Definition 16 *Let $f(x|\theta)$ denote the joint pdf of the samples $X = (X_1, X_2, ..., X_n)$| Then, given that $X = x$ is observed, the function of θ defined by:*

$$\mathcal{L}(\theta \mid x) = f(x|\theta)$$

is called ***likelihood function****.*

To clarify this concept, let X be a discrete random vector, and x the vector of realizations, then

$$\mathcal{L}(\theta \mid x) = P_\theta(X = x).$$

Now, comparing the likelihood function at two distinct parameter points, assume that this holds

$$P_{\theta_1}(\mathbf{X} = \mathbf{x}) = \mathcal{L}(\theta_1 \mid \mathbf{x}) > \mathcal{L}(\theta_2 \mid \mathbf{x}) = P_{\theta_2}(\mathbf{X} = \mathbf{x}). \tag{1.6}$$

Then this means that the observed sample x is more likely to have occurred with $\theta = \theta_1$ than $\theta = \theta_2$. In terms of likelihood, this means that the value $\theta = \theta_1$ is a more plausible true value for θ than θ_2 is.

The big difference between the pdf and the likelihood function is which variable is considered "fixed" and which is varying instead. In the pdf, $f(x|\theta)$, x is varying, meanwhile for the likelihood $\mathcal{L}(\theta \mid x)$, x stays fixed.

When having to manipulate the likelihood, usually it's more convenient to apply the logarithm on the function. In this case it is commonly called log-likelihood function, is defined by

$$\ell(\theta) = \log L(\theta).$$

Therefore, by definition, the following equality holds:

$$\arg\max_\theta L(\theta \mid \bar{x}) = \arg\max_\theta f(\bar{x} \mid \theta) = \arg\max_\theta f(x_1, ...x_n|\theta). \tag{1.7}$$

This means that maximizing the probability density function at x_j corresponds to maximizing the likelihood of the specific observation x_j. Moreover, since the likelihood function summarizes all the evidence provided by the data, choosing the value of θ that maximizes the likelihood makes it the most "plausible" explanation of the observed data.

Again, since the normal distribution will be widely used during this work, it's convenient to introduce the likelihood of an independent sample for the normal distribution.
If $X_1, X_2, \ldots, X_n$ are independent and normally distributed with mean μ and variance σ^2, their probability density function is given by

$$f(x_i; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2}\right).$$

The likelihood function for the entire sample is obtained as

$$L(\mu, \sigma^2) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i - \mu)^2}{2\sigma^2}\right).$$

Taking the logarithm, the log-likelihood function is derived as

$$\ell(\mu, \sigma^2) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i - \mu)^2.$$

1.2 Bayesian Probability Notions

Bayesian probability has been developed as an alternative to classical statistical approaches. The adoption of Bayesian methods has been applied across multiple scientific domains in recent decades [5].
A fundamental distinction between the two formulations can be found in how probability is interpreted.
Probability has been defined through the frequency of repeated events, while

in the Bayesian framework, probability has been conceived as a measure of belief. This belief can be modified when additional data is collected.
The practical value of this approach has been demonstrated in scenarios where uncertainties must be measured and updated continuously when new evidence is found out.
An example of the difference between the two formulation is the likelihood function. In traditional statistics the likelihood revoles around these three key facts:

- The likelihood is seen as a function of the parameters given fixed data
- It's used to find point estimates
- Parameters are treated as fixed, unknown constants

Meanwhile in the Bayesian setting:

- The likelihood is combined with the prior distribution using Bayes' theorem (which will be introduced soon)
- It helps update prior beliefs about parameters to get posterior distributions
- Parameters are treated as random variables with probability distributions

In the Bayesian setting, parameters, usually gathered under the symbol θ, are treated as random variables, with their variability encoded in a prior distribution $\pi(\theta)$.
This prior distribution represents the experimenter's subjective belief about the parameter and can incorporate results from previous studies, expert knowledge, or be specified as a non-informative distribution when little prior knowledge exists.
All Bayesian inference revolves around the possibility of using the Bayes rule (1.5) to find out the posterior distribution of a certain parameter θ.
This can be done using the following:

$$\pi(\theta|X) = \frac{f(X|\theta)\pi(\theta)}{m(X)} \tag{1.8}$$

where $\pi(\theta|X)$ is defined as the posterior distribution, $f(X|\theta)$ as the likelihood, $\pi(\theta)$ as the prior probability, and $m(X)$ as the normalizing factor, calculated as:

$$m(X) = \int f(X|\theta)\pi(\theta)d\theta \tag{1.9}$$

In this context, the conditional probability conveys the relationship between data and model parameters.
The likelihood function $f(X|\theta)$ instead, represents the conditional probability of observing the data X given specific parameter values θ.
Similarly, the posterior distribution $\pi(\theta|x)$ represents the conditional probability of the parameters given the observed data.
The posterior distribution $\pi(\theta|X)$ explains both the variability of the parameter and its location.
While it is possible to summarize this distribution into a single value through expectation or other measures, the full posterior distribution provides a deeper understanding of the parameter's uncertainty.

A particularly useful concept in Bayesian analysis is that of conjugate families [6]. These are pairs of prior and likelihood distributions where the posterior distribution belongs to the same family as the prior, simplifying the computations for the Bayesian inference.
In such cases, updating the parameter estimates reduces to updating the parameters of the prior distribution with information from the sample.

Even though Bayesian inference is particularly suitable for some kind of problem, the calculation of the posterior distribution often involves intractable integrals, particularly in the computation of the normalizing constant $m(X)$. This computational complexity is an important drawback of this approach, more so for large-scale problems.

1.2.1 Multivariate Normal

In order to introduce Gaussian processes, which form the core of this work, it is first necessary to present the **Multivariate Normal** distribution and its key properties [7].
The multivariate normal distribution generalizes the one-dimensional normal distribution to higher dimensions. In fact, a random vector is said to be k-variate normally distributed if every linear combination of its k components follows a univariate normal distribution.

Therefore, a k-dimensional random vector $\mathbf{X} = (X_1, \ldots, X_k)^T$ follows a multivariate normal distribution if, given the vector $\mu \in \mathbf{R}^k$ and the symmetric positive definite matrix Σ, then

$$P(\mathbf{X}) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right) \tag{1.10}$$

and it is denoted as

$$\mathbf{X} \sim \mathcal{N}_k(\boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{1.11}$$

In the previous expression, the k-dimensional vector $\boldsymbol{\mu} \in \mathbf{R}^k$ represents the mean of $\mathbf{X}$:

$$\boldsymbol{\mu} = \mathbb{E}[\mathbf{X}] = (\mathbb{E}[X_1], \mathbb{E}[X_2], \ldots, \mathbb{E}[X_k])^T, \tag{1.12}$$

while Σ represents its covariance matrix:

$$\Sigma_{i,j} = \mathbb{E}[(X_i - \mu_i)(X_j - \mu_j)] = \mathrm{Cov}[X_i, X_j] \tag{1.13}$$

where, in both cases, $1 \leq i \leq k$ and $1 \leq j \leq k$.

Two fundamental properties of multivariate normal distributions are **marginalization** and **conditioning**:

Marginalization

Given a Gaussian partitioned random vector $\mathbf{X} = [\mathbf{X_A}, \mathbf{X_B}]^T$, the marginal distributions are obtained by integrating out the other variables:

$$P(\mathbf{X_A}) = \int P(X_A, X_B) dX_B \tag{1.14}$$

$$P(\mathbf{X_B}) = \int_{X_A} P(X_A, X_B; \mu, \Sigma), dX_A \tag{1.15}$$

More importantly, this operation preserves normality:

$$\mathbf{X_A} \sim \mathcal{N}(\boldsymbol{\mu_A}, \boldsymbol{\Sigma_{AA}}) \tag{1.16}$$

$$\mathbf{X_B} \sim \mathcal{N}(\boldsymbol{\mu_B}, \boldsymbol{\Sigma_{BB}}) \tag{1.17}$$

where

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_A \\ \boldsymbol{\mu}_B \end{bmatrix}, \quad \boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{\Sigma}_{AA} & \boldsymbol{\Sigma}_{AB} \\ \boldsymbol{\Sigma}_{BA} & \boldsymbol{\Sigma}_{BB} \end{bmatrix} \tag{1.18}$$

The conditional distribution of $\mathbf{X_A}$ given $\mathbf{X_B}$ remains normal:

$$\mathbf{X_A}|\mathbf{X_B} \sim \mathcal{N}(\boldsymbol{\mu_{A|B}}, \boldsymbol{\Sigma_{A|B}}) \tag{1.19}$$

where

$$\boldsymbol{\mu_{A|B}} = \boldsymbol{\mu_A} + \boldsymbol{\Sigma_{AB}}\boldsymbol{\Sigma_{BB}^{-1}}(\mathbf{X_B} - \boldsymbol{\mu_B}) \tag{1.20}$$

$$\boldsymbol{\Sigma_{A|B}} = \boldsymbol{\Sigma_{AA}} - \boldsymbol{\Sigma_{AB}}\boldsymbol{\Sigma_{BB}^{-1}}\boldsymbol{\Sigma_{BA}} \tag{1.21}$$

Although this property is widely applied, as models must be updated continuously when new data becomes available, conditioning presents significant computational challenges.
In particular, the formula requires the inversion of $\boldsymbol{\Sigma_{BB}}$, which has a computational complexity of $\mathcal{O}(n^3)$ for an $n \times n$ matrix. In large-scale problems, matrix inversion becomes computationally prohibitive in terms of both time and memory requirements, making approximations the only feasible solution. These considerations are particularly relevant in applications such as Gaussian processes, where conditioning operations are performed repeatedly during model fitting and prediction.
The approach presented in this work leverages a specific type of approximation known as the **Nearest Neighbor Gaussian Process**, which will be introduced in the third chapter.

1.2.2 Gaussian Processes

As stated above, the multivariate normal distribution is used to model **finite** collections of real-valued variables. Its direct extension is the **Gaussian process**, which is used to model **infinite-sized** collections of real-valued variables[8].
The distinctive property of Gaussian processes is that they are not merely considered distributions over random vectors but rather distributions over random functions.
More generally, Gaussian processes belong to the class of **stochastic processes**, which are collections of random functions, $\{f(x) : x \in X\}$, indexed by elements from some set X, known as the index set. Often, the indices $x \in X$ represent time points, so that the variables $f(x)$ describe the temporal evolution of a quantitative phenomenon.

Gaussian processes are such that any finite subcollection of random variables follows a multivariate Gaussian distribution.
In other words, a collection of random variables $\{f(x) : x \in X\}$ is said to be drawn from a Gaussian process with mean function $\mu(\cdot)$ and covariance function $K(\cdot, \cdot)$, also known as Kernel, if and only if, for any finite set of elements $x_1, \ldots, x_m \in X$, the joint distribution of the corresponding set of random variables $\{f(x_1), \ldots, f(x_m)\}$ follows a multivariate normal distribution.

$$\begin{bmatrix} f(x_1) \\ \vdots \\ f(x_n) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu(x_1) \\ \vdots \\ \mu(x_n) \end{bmatrix}, \begin{bmatrix} K(x_1, x_1) & \cdots & K(x_1, x_n) \\ \vdots & \ddots & \vdots \\ K(x_n, x_1) & \cdots & K(x_n, x_n) \end{bmatrix} \right) \tag{1.22}$$

and it is indicated as

$$f \sim \mathcal{GP}(\mu(\cdot), K(\cdot, \cdot)) \tag{1.23}$$

Moreover, given $x_1, x_2 \in X$, the mean and covariance functions are defined as:

$$\mu(x_1) = \mathbb{E}[f(x_1)] \tag{1.24}$$

$$K(x_1, x_2) = \mathbb{E}[(f(x_1) - \mu(x_1))(f(x_2) - \mu(x_2))] \tag{1.25}$$

Regarding the mean function, there are no strict constraints, except that it must be real-valued. It is common practice to assume that it is identically zero, as uncertainty about the mean function can be accounted for by introducing an additional term in the covariance function.

Meanwhile, given a certain covariance function $K(\cdot, \cdot)$, for any finite set of elements $x_1, \ldots, x_m \in X$, the resulting covariance matrix

$$\Sigma = \begin{bmatrix} K(x_1, x_1) & \ldots & K(x_1, x_m) \\ \vdots & \ddots & \vdots \\ K(x_m, x_1) & \ldots & K(x_m, x_m) \end{bmatrix} \tag{1.26}$$

must be positive semidefinite.

This implies that the matrix $\Sigma \in \mathbb{R}^{m \times m}$ must satisfy the following condition:

$$x^\top \Sigma x \geq 0 \quad \text{for all vectors } x \in \mathbb{R}^m.$$

This formulation leads to several key properties that serve as verification criteria:

- **Symmetry:** The matrix Σ must satisfy $\Sigma = \Sigma^\top$.
- **Non-negative Eigenvalues:** All eigenvalues of Σ must be non-negative.
- **Quadratic Form:** For any vector x, the quadratic form $x^\top \Sigma x$ must always be non-negative.

Exponential Kernel

In order clarify the relationship between the covariance matrix Σ and the kernel $K(\cdot, \cdot)$, it is convenient to show example.
Consider the following Gaussian process:

$$f(\cdot) = \mathcal{GP}(0, K(\cdot, \cdot))$$

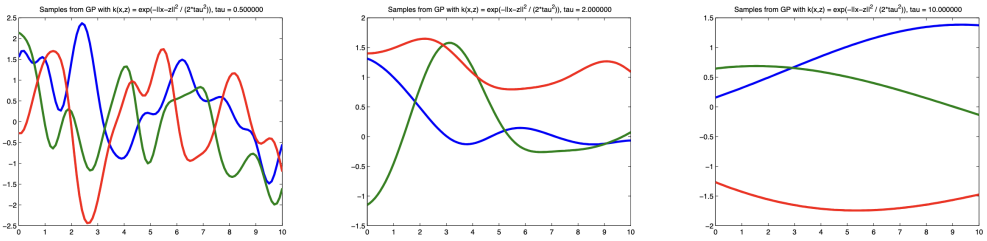
where $f : \mathbb{R} \to \mathbb{R}$. As for the choice of the covariance function, it is set to the **squared exponential kernel**, which is defined as:

$$K_{SE}(x, x') = exp\Big(-\frac{1}{2\tau^2}||x - x'||^2\Big), \quad \tau > 0$$

As the GP chosen has mean zero, the expectation is that the drawn functions will tend to be distributed around zero.
Then, given this kernel choice, for any pair of elements $x, x' \in \mathbb{R}$:

- $f(x)$ and $f(x')$ will tend to have high covariance when x and x' are similar in the input space (i.e., $\|x - x'\| \approx 0$), so that $K(x, x') = \exp\left(-\frac{1}{2\tau^2}\|x - x'\|^2\right) \approx 1$).
- $f(x)$ and $f(x')$ will tend to have low covariance when x and x' are dissimilar (i.e., $\|x - x'\| \gg 0$), so that $K(x, x') = \exp\left(-\frac{1}{2\tau^2}\|x - x'\|^2\right) \approx 0$).

This behavior can be summed up as functions drawn from a zero-mean Gaussian process prior with the squared exponential kernel will tend to be "locally smooth" with high probability. This translates to the fact that closer function values are highly correlated, and this correlation decrease as a function of distance in the input space (see Fig.1.4).



(a) (b) (c)

Figure 1.4: Samples from a zero-mean Gaussian process prior with $K_{SE}(\cdot, \cdot)$ covariance function, using (a) $\tau = 0.5$, (b) $\tau = 2$, and (c) $\tau = 10$. Note that as the bandwidth parameter τ increases, then points which are farther away will have higher correlations than before, and hence the sampled functions tend to be smoother overall [7].

Gaussian Processes for Regression Tasks

One of the most significant applications of Gaussian processes is their use in regression tasks [9, 10]. Regression involves finding a best-fitting function to describe an underlying phenomenon, given only a set of observed samples.

Consider the dataset $\mathcal{S} = \{(x_i, y_i)\}_{i=1}^n$, where (x_i, y_i) are independent and identically distributed (i.i.d.) samples.
In the Gaussian process regression model, the observations are assumed to follow:

$$y_i = f(x_i) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma_n^2) \tag{1.27}$$

where a zero-mean Gaussian process prior is placed on the function $f(\cdot)$:

$$f(\cdot) \sim \mathcal{GP}(0, \Sigma(\cdot, \cdot)) \tag{1.28}$$

for a valid covariance function $\Sigma(\cdot, \cdot)$.

Now, let $T = \{(x_*^{(i)}, y_*^{(i)})\}_{i=1}^{m_*}$ be the test set, consisting of i.i.d. variables drawn from the same unknown distribution as $\mathcal{S}$.

To ease the notation, fix the following

$$X = \begin{bmatrix} - & (x^{(1)})^\intercal & - \\ - & (x^{(2)})^\intercal & - \\ & \vdots & \\ - & (x^{(m)})^\intercal & - \end{bmatrix} \in \mathbb{R}^{m \times n}, \quad \tilde{f} = \begin{bmatrix} f(x^{(1)}) \\ f(x^{(2)}) \\ \vdots \\ f(x^{(m)}) \end{bmatrix} \tag{1.29}$$

$$\tilde{\epsilon} = \begin{bmatrix} \epsilon^{(1)} \\ \epsilon^{(2)} \\ \vdots \\ \epsilon^{(m)} \end{bmatrix}, \quad \tilde{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(m)} \end{bmatrix} \in \mathbb{R}^m, \tag{1.30}$$

$$X_* = \begin{bmatrix} - & (x_*^{(1)})^\intercal & - \\ - & (x_*^{(2)})^\intercal & - \\ & \vdots & \\ - & (x_*^{(m_*)})^\intercal & - \end{bmatrix} \in \mathbb{R}^{m_* \times n}, \quad \tilde{f}_* = \begin{bmatrix} f(x_*^{(1)}) \\ f(x_*^{(2)}) \\ \vdots \\ f(x_*^{(m_*)}) \end{bmatrix} \tag{1.31}$$

$$\tilde{\epsilon}_* = \begin{bmatrix} \epsilon_*^{(1)} \\ \epsilon_*^{(2)} \\ \vdots \\ \epsilon_*^{(m_*)} \end{bmatrix}, \quad \tilde{y}_* = \begin{bmatrix} y_*^{(1)} \\ y_*^{(2)} \\ \vdots \\ y_*^{(m_*)} \end{bmatrix} \in \mathbb{R}^{m_*}. \tag{1.32}$$

Since the marginalization property of Gaussian processes holds, any function $f(\cdot)$ drawn from a GP prior with covariance function $k(\cdot, \cdot)$ must satisfy that the marginal distribution over any finite subset of X follows a joint multivariate normal distribution.
This property applies in particular when concatenating the training and test sets:

$$\begin{bmatrix} \vec{f} \\ \vec{f}_* \end{bmatrix} \Big| X, X_* \sim \mathcal{N}\left(\vec{0}, \begin{bmatrix} \Sigma(X,X) & \Sigma(X,X_*) \\ \Sigma(X_*,X) & \Sigma(X_*,X_*) \end{bmatrix}\right), \tag{1.33}$$

where

$$\Sigma(X,X) = \Sigma(x^{(i)}, x^{(j)}), \quad \Sigma(X_*,X_*) = \Sigma(x_*^{(i)}, x_*^{(j)})$$

$$\Sigma(X,X_*) = \Sigma(x^{(i)}, x_*^{(j)}), \quad \Sigma(X_*,X) = \Sigma(x_*^{(i)}, x^{(j)}).$$

Moreover, since it is assumed that the noise terms in both the training and test sets are i.i.d., it follows that:

$$\begin{bmatrix} \vec{\varepsilon} \\ \vec{\varepsilon_*} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \sigma^2 I & \mathbf{0} \\ \mathbf{0}^T & \sigma^2 I \end{bmatrix}\right).$$

Now, recalling the standard regression model:

$$y_i = f(x_i) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}(0, \sigma_n^2), \tag{1.34}$$

and given the Gaussian process prior, it follows that the sum of normal random variables remains normally distributed. As a result, the model formulation extends to:

$$\begin{bmatrix} \vec{y} \\ \vec{y_*} \end{bmatrix} \Big| X, X_* = \begin{bmatrix} \vec{f} \\ \vec{f_*} \end{bmatrix} + \begin{bmatrix} \vec{\varepsilon} \\ \vec{\varepsilon_*} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \Sigma(X,X) + \sigma^2 I & \Sigma(X,X_*) \\ \Sigma(X_*,X) & \Sigma(X_*,X_*) + \sigma^2 I \end{bmatrix}\right).$$

The posterior predictive distribution is then obtained by conditioning the joint Gaussian distribution on the observations:

$$\vec{y_*} | \vec{y}, X, X^* \sim \mathcal{N}(\mu_*, \Sigma_*), \tag{1.35}$$

where:

$$\mu_* = \Sigma(X_*, X)\Big[\Sigma(X,X) + \sigma^2 I\Big]^{-1} \vec{y} \tag{1.36}$$

$$\Sigma_* = \Sigma(X_*, X_*) + \sigma^2 I - \Sigma(X_*, X)\Big[\Sigma(X,X) + \sigma^2 I\Big]^{-1} \Sigma(X, X_*). \tag{1.37}$$

While this formulation is particularly convenient, it suffers from the same computational limitations as the multivariate normal distribution. Specifically, although the posterior is well-defined, inverting a potentially large covariance matrix introduces significant computational challenges in many practical applications.

1.3 Time Series

Time series analysis is recognized as a fundamental tool for understanding and forecasting the behavior of measurable phenomena over time[11].
In practice, observations are collected at discrete time points to capture the evolution of the underlying process [12]. These observations may represent a single characteristic (univariate time series) or multiple correlated characteristics (multivariate time series).

Let t_0 denote the initial time point and T the final time point of the recordings. The time series S is then defined as the ordered sequence of observations:

$$S = \{x_{ij} : i = 1, \ldots, p;\ j = t_0, \ldots, T\},$$

where x_{ij} represents the observation of the i-th feature at time j and p denotes the number of recorded features.
The overall duration of the series is given by $T - t_0$.

Time series data may exhibit a variety of patterns including trends, seasonalities, and cyclic behaviors. These patterns are analyzed to uncover the underlying dynamics, and such analysis is crucial for the development of accurate predictive models[13].

Trends

A trend represents a long-term, persistent increase or decrease in the data over an extended period. Trends are typically linked to systematic changes such as technological progress, demographic shifts, or economic growth.
They are modeled separately to allow for the isolation of short-term variations from the long-term movement of the data.

Seasonal Patterns

Seasonal patterns refer to regular, periodic fluctuations that occur within fixed intervals, such as daily, monthly, or yearly cycles.
These patterns are generally induced by recurring events, for instance, weather conditions, holidays.
Methods such as seasonal decomposition are employed to extract and model the seasonal component from the overall time series.

Cyclic Behaviors

Cyclic behaviors involve fluctuations that occur over periods longer than a season.
Unlike seasonal patterns, cyclic variations do not have a fixed periodicity and are often influenced by economic or business cycles. These cycles can vary

in duration and amplitude, and their detection generally requires advanced techniques that accommodate non-stationary dynamics.

The presence of outliers and irregular fluctuations is also checked during model formulation. Figure 5.5 illustrates a typical time series, highlighting its key components and variations over time [14].

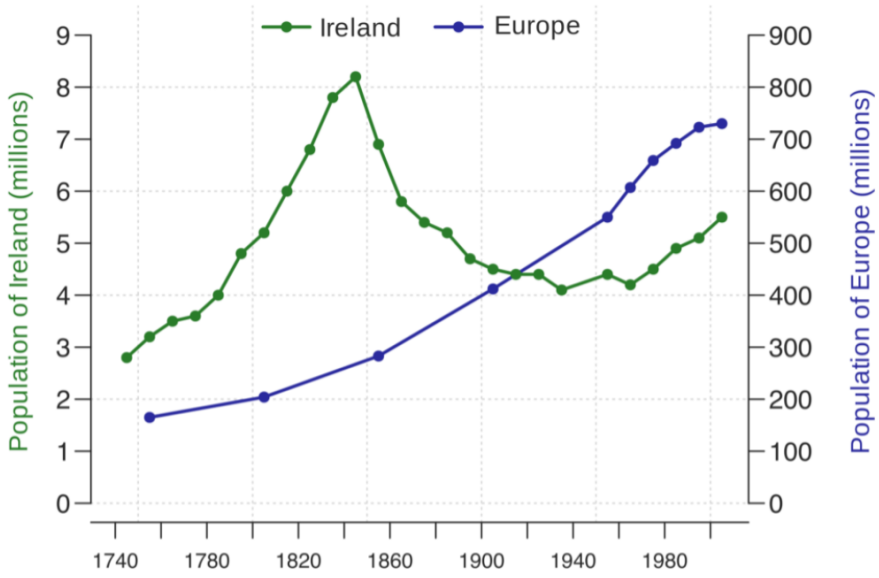


Figure 1.5: Time series reporting the population of Ireland and Europe between 1750 to 2005[15]

1.3.1 The lag plot

A **lag plot** is a graphical representation used in statistics and probability to analyze the dependence between successive values of a time series.
Given a discrete-time stochastic process $\{X_t\}$, the lag-k plot is defined as the set of points

$$\{(X_t, X_{t-k}) \mid t = k+1, \ldots, n\}.$$

This representation allows to visualize trends and seasonal dependencies in the data.

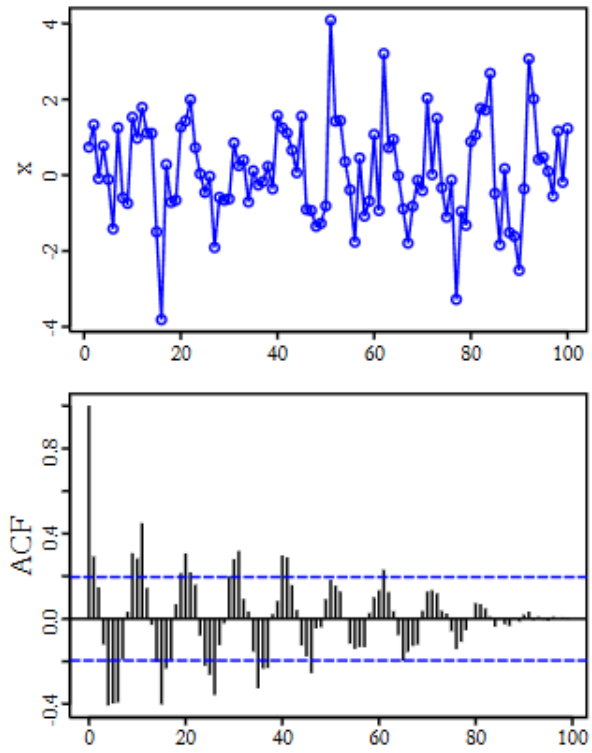


Figure 1.6: A plot showing 100 random numbers with a "hidden" sine function, and an autocorrelation plot of the series on the bottom[16]

If a structured pattern is observed in the plot, statistical dependence between X_t and X_{t-k} exist.
In contrast, if the points are randomly scattered, the observations are taken to be weakly correlated or uncorrelated.
The presence of structure in lag plots is related to the **autocorrelation function** (ACF), which indicates the correlation between X_t and its lagged values, and for this reason the two are interchangeable.
The exact formula for the autocorrelation for a given time series $\{X_t\}$ is

$$\rho(k) = \frac{\mathbb{E}[(X_t - \mu)(X_{t+k} - \mu)]}{\mathbb{E}[(X_t - \mu)^2]} \tag{1.38}$$

while its "empiric" version is

$$\hat{\rho}(k) = \frac{\sum_{t=1}^{n-k}(x_t - \bar{x})(x_{t+k} - \bar{x})}{\sum_{t=1}^{n}(x_t - \bar{x})^2} \tag{1.39}$$

where $\bar{x}$ represents the mean of the time series.

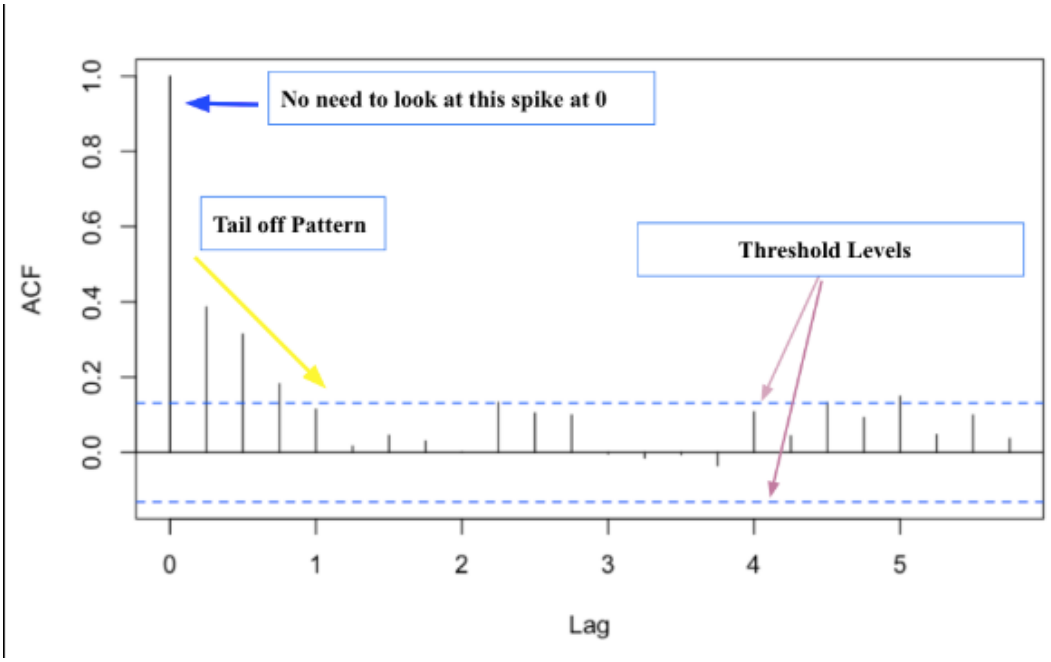


Figure 1.7: Autocorrelation function (ACF) plot illustrating key trend patterns. The spike at lag 0 represents the trivial correlation of the series with itself and can be ignored. The "tail-off pattern" indicates a gradual decay in correlation, which could shown seasonality in the data. The "threshold levels" (dashed lines) mark the significance boundaries, helping to identify whether the autocorrelation at different lags are statistically significant[17]

1.4 Rstudio and Stan

To execute the data analysis, cleaning and visualization R studio[18] was used.
This choice has been led by the fact that R is rich of statistical based libraries, and has a direct integration with STAN. The latter has been used to develop the model itself, and it's a Software for Bayesian Data Analysis.
It enables highly complex statistical modeling using Bayesian inference, allowing for more accurate and interpretable results in complex data scenarios. Since at the start of this work it wasn't clear how complex the model needed would turn out to be Stan's flexibility for a wide range of applications, from simple linear regression to multi-level models and time-series analysis was

ideal.

Chapter 2

Dataset Analysis

2.1 Introduction

Energy is considered to be a fundamental resource that is used to drive modern societies by powering homes, industries, and economies. As the global demand for energy is observed to continue rising, its efficient use and management are gettign more and more important [19].
One key area for future improvements is the monitoring and forecasting of energy consumption.
By accurately monitoring and forecasting energy consumption, resource allocation can be optimized to ensure a more sustainable and responsible use.
Obtaining clean, high-frequency energy consumption data for households is a challenging task, and the process is further complicated by the presence of outliers in the available data.
Current methods for forecasting energy consumption have been based on the application of statistical techniques to datasets that are collected from smart meters. These datasets are considered to be expensive and time-consuming to obtain and are affected by issues such as sparsity and irregularities.
In addition, residential energy consumption data is even more problematic, since in recent years personal privacy concerns have risen [20].
One of the possible solutions to address these problems consists in using already available data to train new generative model. This would allow the use of generative, synthetic, data as basis for the data analysis and forecasting, avoiding the problems linked to gathering real data.
Although, in order to create a high-quality model, it is necessary to analyze

existing data to gather domain knowledge about common user behaviors and factors that may affect energy consumption [21].
The aim of this work is to identify the most important consumption patterns from a small energy consumption dataset, to create a statistical model that is used to describe these patterns in a simple manner, and to use this model to generate new data. The quality of the new data is then assessed in comparison with the original dataset.
The analysis is based on an in-depth exploration of the dataset, with both short-term and long-term trends being extracted and key features, such as daily cycles, being highlighted.
These patterns are used to inform the construction of a comprehensive model that is able to generate data that are observed to closely resemble the original dataset.
The model is validated through a comparison with synthetic data generated from the original dataset, both visually and in terms of its underlying trends.

2.2 Energy Consumption Dataset

The dataset used in this work originates from a small energy community located in the city of Pinerolo, Italy.
The data collection starts on January 1^{st}, 2022, and covers one full year of energy consumption for all community members.
Although the dataset is not extensive, it is well diversified, including users of different types.
The dataset is divided into two distinct collections: **Measurements** and **Attributes**.
While the user base is shared between them, the grouping and focus of the data differ. However, for both collections, the data granularity is one hour, resulting in a multivariate time series of length 365×24 for each user.
In the following subsections, the two collections are described in detail.

Measurements Dataset

The Measurements dataset is structured as follows:

- *Dimensions*: The dataset consists of 115 columns and 8760 rows.
- *Data*: The first column contains the timestamp, starting from January 1^{st}, 2022, at $00:00:00$. The remaining columns represent 114 distinct

users, where each entry corresponds to the hourly energy consumption of a user, expressed in kWh.

As shown in Fig.2.1, users are identified through a code, which is introduced in the Attributes dataset.

timestamp	bta1_1	bta1_10	bta1_2	bta1_3	bta1_4	bta1_5	bta1_6	bta1_7	bta1_8	bta1_9	bta1_c1
2022-01-01 00:00:00	0.016	0.054	0.008	0.033	0.007	0.020	0.009	0	0.030	0.090	0.148
2022-01-01 01:00:00	0.008	0.123	0.008	0.026	0.007	0.008	0.010	0	0.029	0.074	0.150
2022-01-01 02:00:00	0.014	0.052	0.007	0.028	0.007	0.011	0.009	0	0.030	0.048	0.151
2022-01-01 03:00:00	0.015	0.115	0.008	0.026	0.006	0.009	0.009	0	0.030	0.042	0.151
2022-01-01 04:00:00	0.016	0.062	0.007	0.025	0.006	0.008	0.010	0	0.030	0.043	0.150
2022-01-01 05:00:00	0.008	0.079	0.008	0.026	0.006	0.008	0.009	0	0.029	0.042	0.150

Figure 2.1: First rows of the Measurements dataset.

The dataset does not contain any missing values. However, numerous instances are present where the recorded energy consumption is equal to zero.

	Column	Zero_Count
bta2_c3	bta2_c3	8443
bta3_8	bta3_8	6044
bta1_7	bta1_7	5351
bta2_8	bta2_8	1581
bta2_c7	bta2_c7	1494
dom2_4	dom2_4	1099
bta3_c3	bta3_c3	1089
bta3_c1	bta3_c1	519
bta5_7	bta5_7	445
bta5_10	bta5_10	402
bta3_10	bta3_10	227
dom3_7	dom3_7	151
bta6_c7	bta6_c7	103
bta4_c2	bta4_c2	47
dom4_7	dom4_7	24
bta3_2	bta3_2	19
bta6l_4	bta6l_4	18

Figure 2.2: Distribution of zero consumption values.

These cases pose a challenge in identifying user consumption patterns. A consumption value of zero indicates that the user has completely turned off their power, making the data point "meaningless."

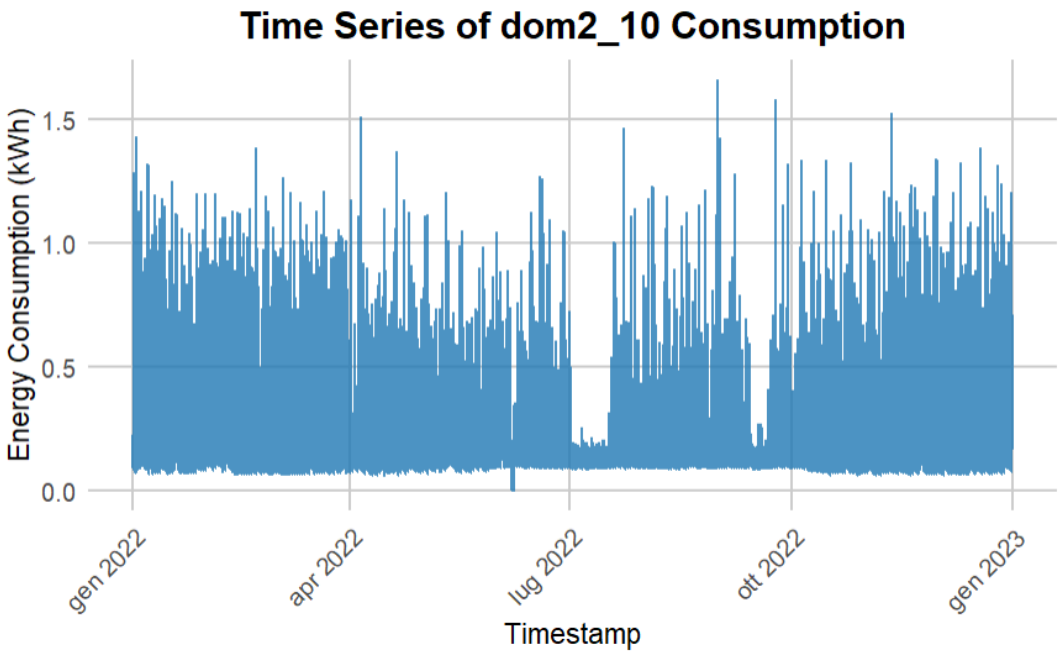


Figure 2.3: Time series of energy consumption for the user *dom*2_10 throughout 2022.

For this reason, users with an excessive number of zero entries were excluded from the analysis.
Specifically, as shown in Fig.2.2, all users with 1000 or more zero entries were discarded.
For users with $100 < \#zeros < 1000$, the consumption time series was analyzed and the following procedure was applied:

- If the zero values were consecutive, indicating that the power was turned off for a few days, a small value of 0.0001 was added to these entries. This adjustment was necessary for the application of a log transformation to the time series.

- If the zero values were scattered, possibly due to a blackout or measurement error, each zero entry was replaced with the value recorded at the same hour on the preceding day.

Attributes Dataset

The Attributes dataset exhibits a structure that differs significantly from that of the Measurements dataset.
The dataset comprises 114 rows (one for each user) and 14 columns, which are listed and briefly described below:

- **user**: Represented by the same code as in the Measurements dataset;
- **type**: Derived from the *user* code, indicating whether the user belongs to the "bta" or "dom" category. The "dom" category includes domestic users (households), while the "bta" category (which stands for "bassa tensione e altri usi" in italian) comprises entities that are not considered private apartments (e.g., offices, sports centers, and buildings in general);
- **class**: The power level, identified by the number following the type;
- **category**: A general categorization of the kind of user;
- **activity_type**: The specific activity carried out in the building;
- **voltage_level**: Indicates the supply voltage level, which may be either $220V$ or $380V$. Typically, household contracts are set to $220V$, whereas some businesses (such as bakeries and restaurants) use $380V$. Higher voltage generally implies higher consumption (see Fig.2.5);
- **yearly_energy**: Total energy consumed during 2022, measured in Kw;
- **p_mean**, **p_std**, **p_min**, **p_q1**, **p_median**, **p_q3**, **p_max**: Various statistical measures regarding the energy consumption, including mean, median, and quantiles;

user	type	class	category	activity_type	voltage_level	yearly_energy	p_mean	p_std	p_min	p_q1	p_median	p_q3	p_max
bta1_1	bta	bta1	office	estate_agency	220	114.2040	0.0130370	0.0076695	0.007	0.008	0.009	0.016	0.087
bta1_10	bta	bta1	cultural_and_sports	association	220	1335.8180	0.1524906	0.1834451	0.000	0.043	0.078	0.204	1.682
bta1_2	bta	bta1	residential	multifamily_building	220	76.8830	0.0087766	0.0018802	0.000	0.007	0.008	0.010	0.022
bta1_3	bta	bta1	office	business	380	175.6740	0.0200541	0.0133207	0.008	0.009	0.022	0.026	0.348
bta1_4	bta	bta1	residential	multifamily_building	220	89.9365	0.0102667	0.0195912	0.003	0.006	0.007	0.008	0.641
bta1_5	bta	bta1	NA	NA	220	86.7060	0.0098979	0.0063457	0.007	0.008	0.008	0.009	0.112

Figure 2.4: First rows of Attributes dataset.

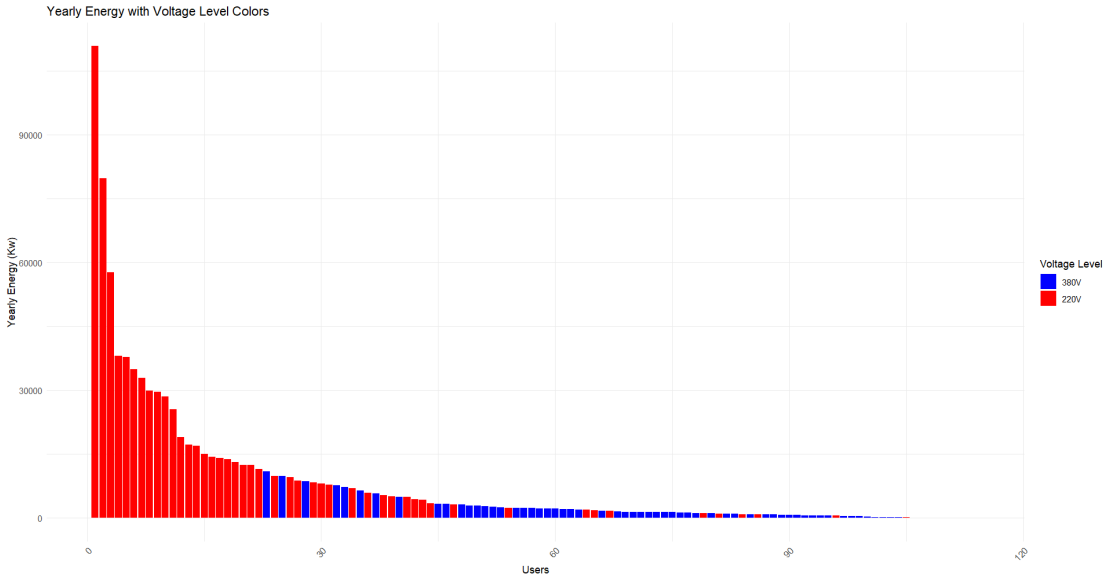


Figure 2.5: Histogram of the yearly energy consumption for the whole Attributes dataset. The two colors indicates which voltage level the user has chosen in the electricity contract. It seems that higher voltage is correlated with higher consumption.

The Attributes dataset is considerably cleaner than the Measurements dataset, containing only two NAN values, one in the column "category" and one in "activity_type". As neither of these will be used in the analysis, no cleaning has been performed.
Below, a grouped count for each "category" type is visualized:

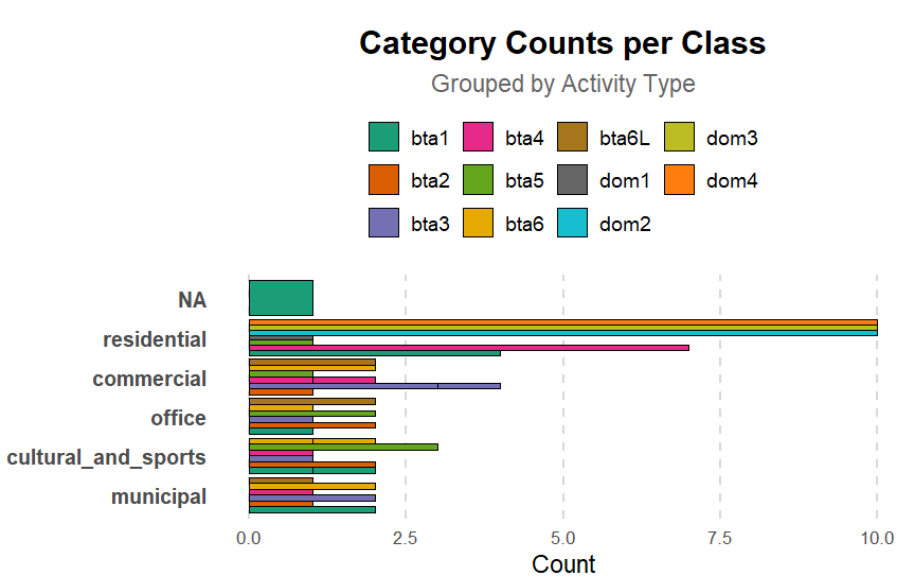


Figure 2.6: Histogram showing a grouped count of each category for each class in Attributes dataset.

Basic data exploration has been applied to the dataset to gain further insight and to identify hidden patterns.
Initially, the total energy consumption was visualized with respect to "activity_type" and "category":

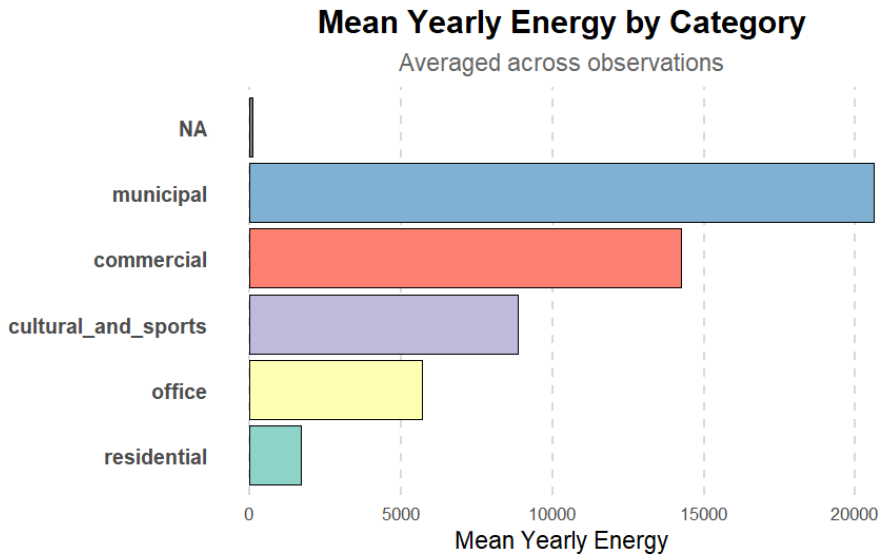


Figure 2.7: Histogram showing the mean yearly energy consumption for each category in the Attributes dataset.

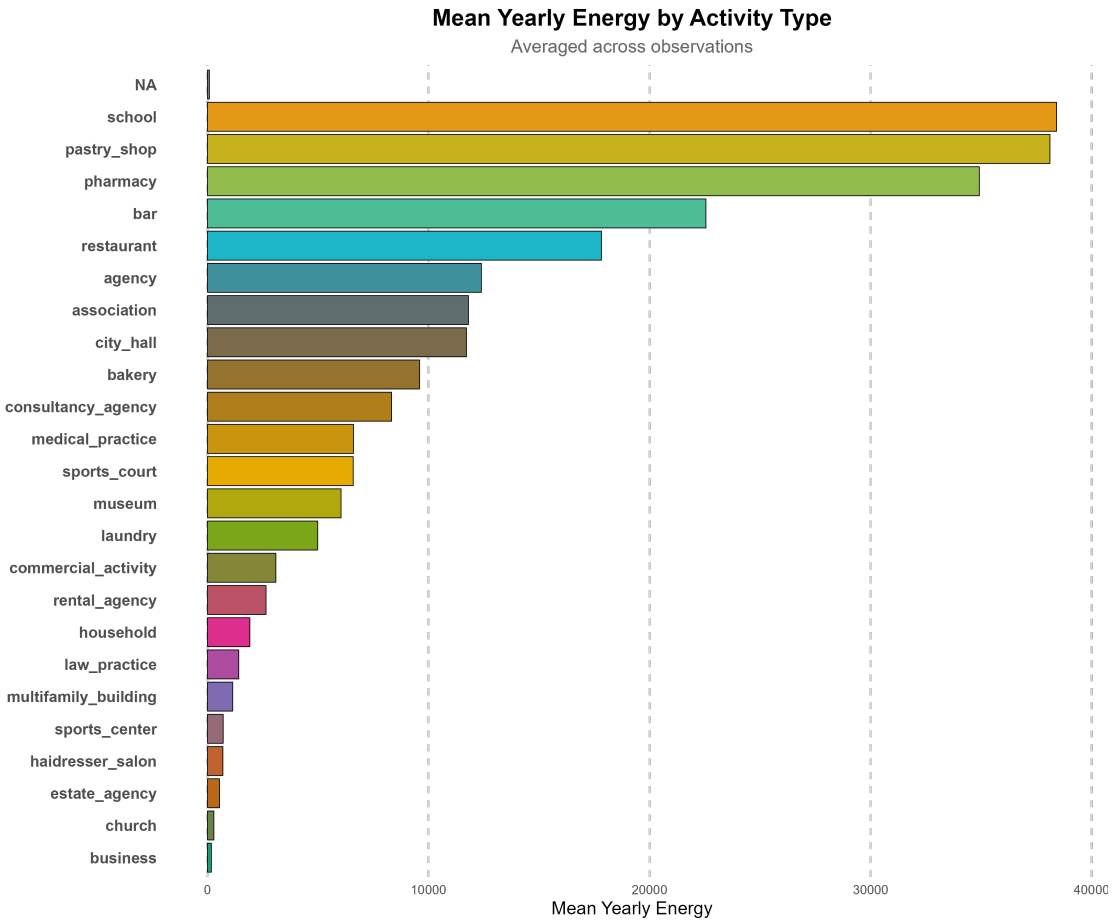


Figure 2.8: Histogram showing the mean yearly energy consumption for each activity type in the Attributes dataset.

It is evident that consumption behavior differs significantly across categories. However, even within a single category, it is difficult to box the data (e.g., Fig.2.9).

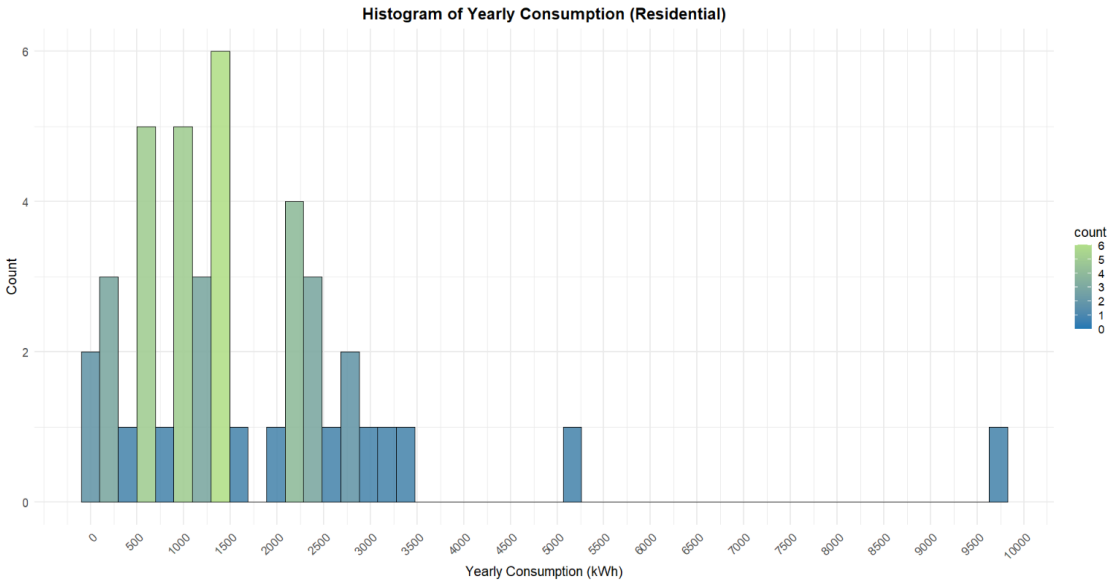


Figure 2.9: Histogram of yearly energy consumption for Residential users

Consequently, the study has been focused on households, as it is presumed that common patterns may be more readily identified among domestic users despite differences in individual habits.

2.2.1 Finding Hidden Patterns in the Dataset

After the dataset was analyzed to identify the available information and its potential uses, the next step involved searching for specific patterns in the energy consumption time series.
As previously noted, the Attributes dataset was initially used to assess the value of leveraging various categories and activity types.
Consequently, the analysis now focuses solely on the residential (household) category.
Prior to the detailed analysis, a box plot of the distribution of yearly energy consumption for the household category was generated:

Figure 2.10 clearly shows that some users are outliers, exhibiting energy consumption values that are substantially higher than those of the remainder of the dataset.
By excluding samples with a total consumption greater than 7000 kW, the distribution appears much cleaner:

After this, the time series plot of the energy consumption for the user

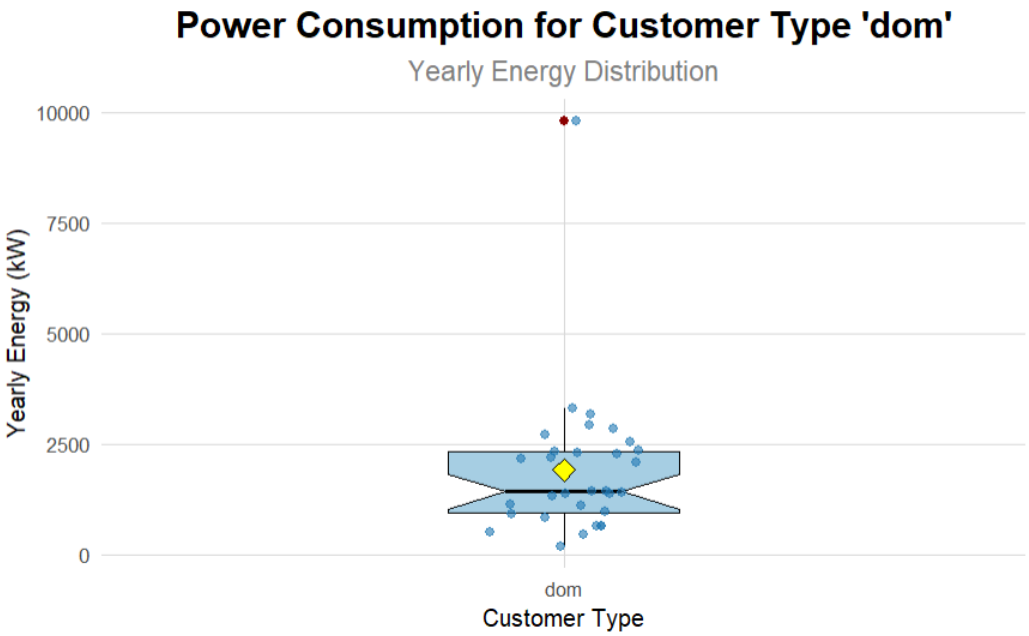


Figure 2.10: Boxplot showing the distribution of the energy consumption for the residential customer of type "dom". In the plot two outliers are clearly identified.

“dom2_10” is presented again with a different focus, to evaluate whether reducing the amount of data improves clarity and reveals a pattern:

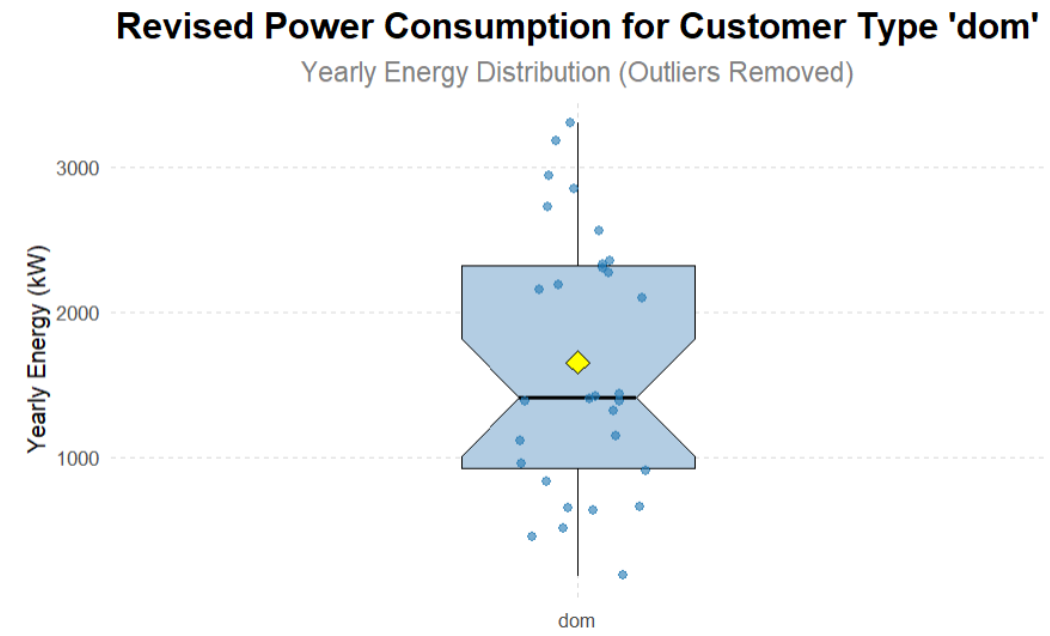


Figure 2.11: Boxplot showing the distribution of the energy consumption for the residential customer of type "dom", taking out the outlier identified in the previous step. After the cleaning the boxplot seems shows more clearly information about the distribution of the users.

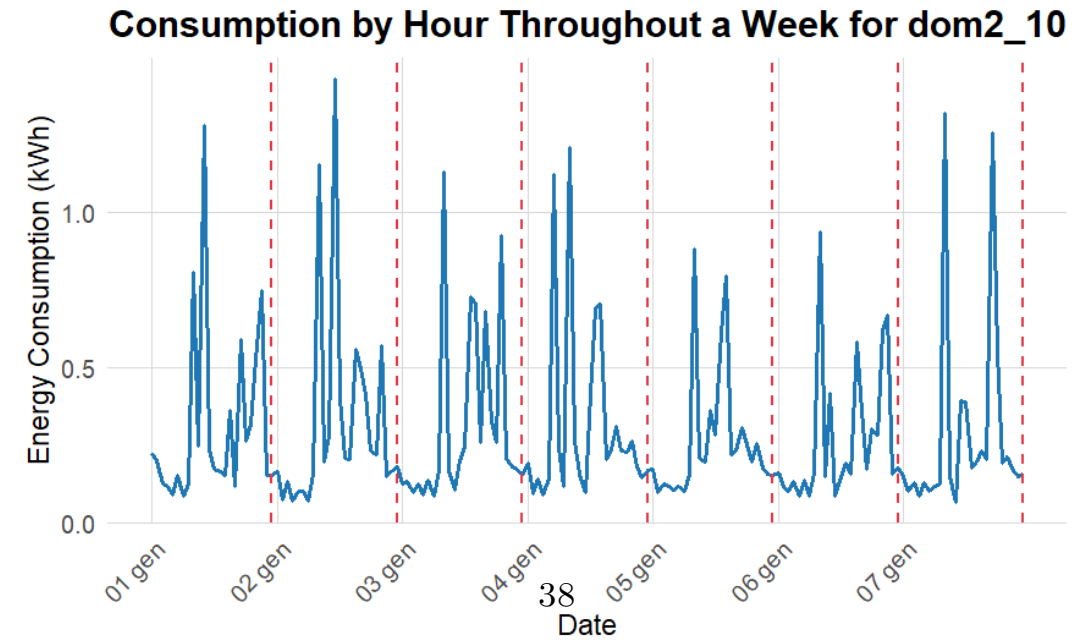


Figure 2.12: Classic line plot of the energy consumption time series for the user *dom*2_10. In particular the data about the first week of January is shown. The vertical lines signals the start of a new day.

Figure 2.12, which displays the energy consumption for the first week of January, reveals a day-to-day pattern indicated by discontinuous vertical lines.
To investigate this pattern further and understand its nature, a heatmap of the consumption was generated:

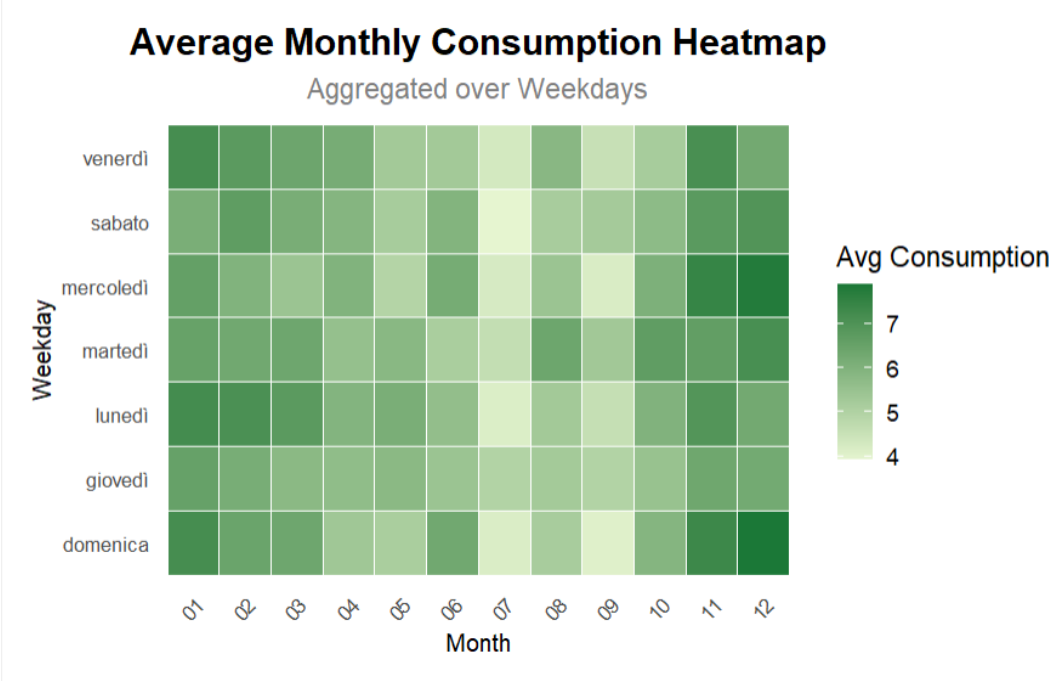


Figure 2.13: Heatmap of the energy consumption time series for the user *dom*2_10. To create the plot, data manipulation has been applied, creating the adding the weekday as new feature for the data. In the plot no clear tren is shown except for a seasonal one.

The heatmap represents the **average monthly energy consumption** aggregated by **weekdays**.
The x-axis corresponds to the **months**, while the y-axis corresponds to the **days of the week** (in Italian). The color intensity indicates the **level of energy consumption**, with darker green shades signifying **higher consumption** and lighter shades indicating **lower consumption**.
Some main patterns are easily identified:

- During the summer season, particularly in **July and August**, lighter colors are observed, indicating **lower consumption** as expected due to

the absence of heater use during warm months;

- Certain weekdays, such as **Friday and Sunday**, exhibit darker shades in some months, suggesting **higher energy usage**;

- Conversely, during the winter season, particularly in **November and December**, energy consumption increases, which is expected due to heater usage and festivities during which families tend to cook more.

Unfortunately, as the dataset covers only one year, focusing on seasonal patterns is not advisable due to insufficient data.
In addition, no clear pattern for the weekdays was observed; therefore, this possibility was not further investigated.
The next step involved a deeper investigation of the pattern highlighted in Fig. 2.12.
To validate the presence of a day-to-day pattern in the data, a lag plot was employed.
Lag plots display the time series against lagged versions of itself, which helps visualize autocorrelation even when standard auto-correlations vanish.
Thus, if a pattern is visible in the lag plot, complex dependencies in the time series can be identified.
Figure 2.14 displays an initial lag plot, also known as an ACF (autocorrelation function) plot:

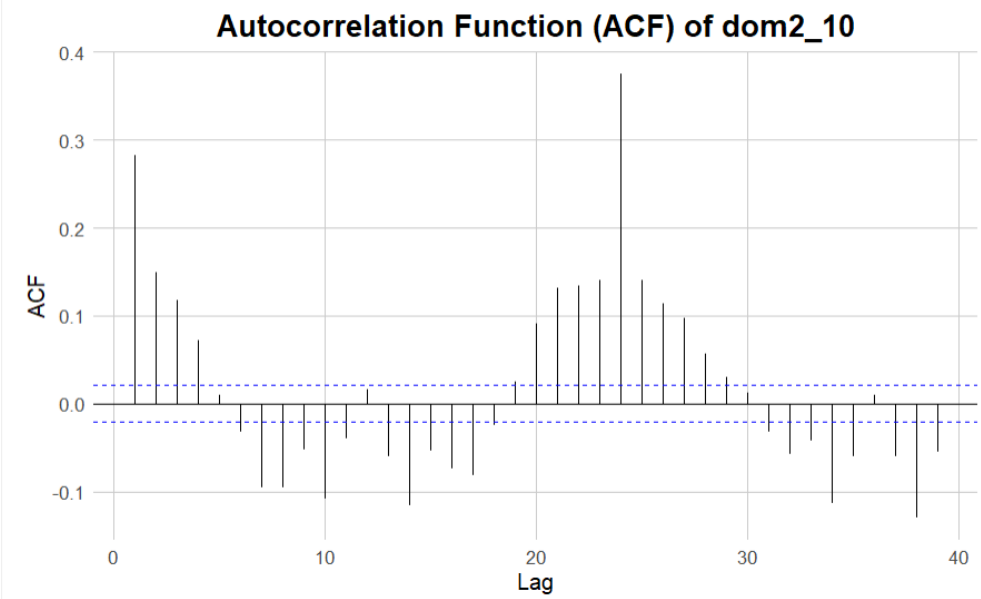


Figure 2.14: Autocorrelation function plot of the energy consumption time series for the user *dom*2_10. In this case no other specification has been passed to the function, therefore the *max_lag* value is 40

A correlation is observed with a lag between 20 and 30 samples, with a higher peak at the 25^{th} sample. To further identify recurring behaviors, the same lag plot was generated for 170 samples (see Fig. 2.16) and for 500 samples (see Fig. 2.15):

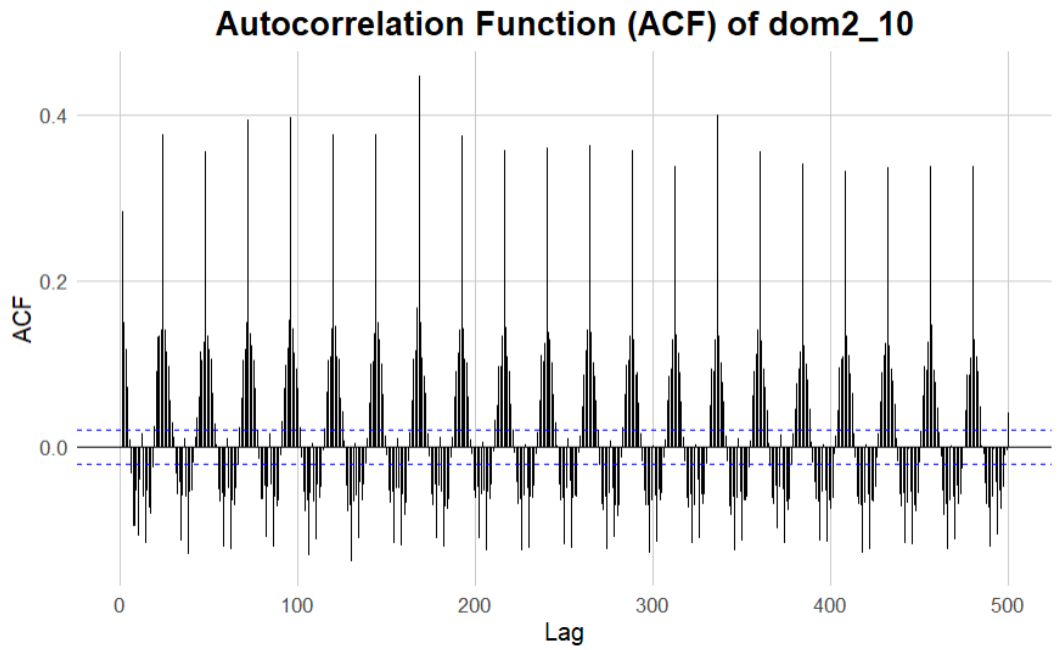


Figure 2.15: Autocorrelation function plot of the energy consumption time series for the user *dom*2_10. In this case the maximum lag has been set to 500, in order to be able to catch weekly pattern. It is clear that a repetitive behavior is present.

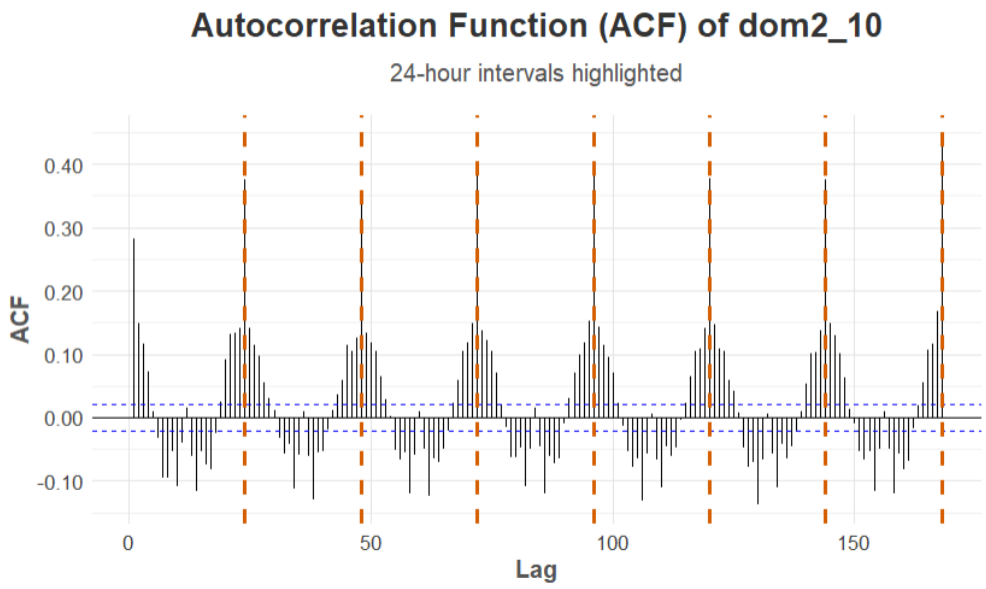


Figure 2.16: Autocorrelation function plot of the energy consumption time series for the user *dom*2_10. In this case the maximum lag has been set to 170, to focus on daily patterns. Moreover, non continuous vertical lines have been plotted each 24 entries. Such lines overlap with the peak value for the correlation.

Figure 2.16 confirms that the daily dependence is a long-term pattern. Such dependence appears to be stable, maintaining the same intensity throughout an entire week. Vertical discontinuous lines have been plotted every 25 samples, overlapping with the peak of each high-correlation group. This observation is consistent with the expectation that each user follows a daily routine.
In Figure 2.15, the same pattern is observed even across several weeks of lag. Furthermore, as shown in Fig. 2.17, three specific groups have been highlighted, corresponding to samples that represent the same reference weekday lagged by one, two, or three weeks.
The peaks of these groups are slightly higher than those of the surrounding groups, suggesting that a weekly pattern is present in addition to the daily pattern, with a stronger dependence at a one-week lag.

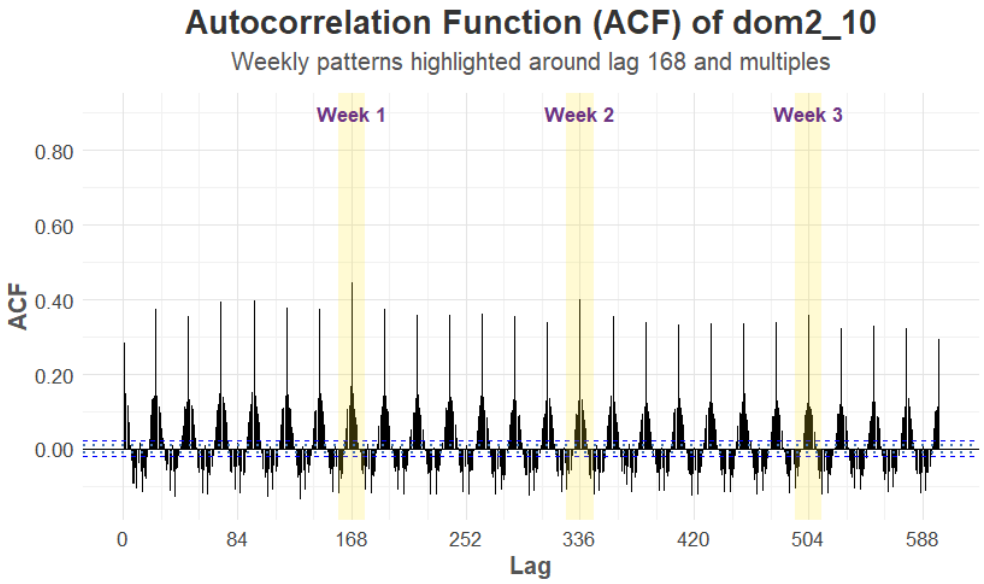


Figure 2.17: Autocorrelation function plot of the energy consumption time series for the user *dom*2_10. In this case the maximum lag has been set to 600, in order to be able to catch weekly pattern. The highlighted sections show that in correspondence of the weekly pattern the peak of the autocorrelation seem to be even higher than in the ther instances.

Chapter 3

Generative NNGP Model

3.1 Nearest Neighbor Gaussian Processes

Nearest Neighbor Gaussian Process (NNGP) based models are defined as a family of highly scalable Gaussian process based models.
The key concepts for the formulation of these models is lays in the Vecchia's approximation [22].
Given three events A, B, and C, their join distribution $P(A, B, C)$, can be expressed as

$$P(A, B, C) = P(A)P(B|A)P(C|A, B) \tag{3.1}$$

With Vecchia's approximation, the latter can be rewritten as

$$P(A, B, C) \approx P(A)P(B|A)P(C|A) \tag{3.2}$$

such approximation becomes more accurate when events B and C are close to be conditionally independent given knowledge of A.
Alternative formulations can be created, but this require knowledge of which events are close to be conditionally independent given others.
This last assumption is then used to construct the NNGP models, extending Vecchia's approximation to a process by assuming conditional independence given information from neighboring locations.
As explained in the previous introductory chapters regarding the Gaussian processes, using the conditioning property is computationally expensive, but required for GP based models.
A possible solution, is in fact applying the Nearest Neighbors approximation [23].

This approach has been designed to address computational constraints while preserving the adaptability of traditional Gaussian Processes, making it an optimal choice when the sample size becomes prohibitive.

Given a certain quantitative phenomenon $y(s)$, where $s \in S$ are usually temporal indices, the general formulation of the NNGP model is written as

$$y(s) = m_\theta(s) + w(s) + \epsilon(s) \tag{3.3}$$

where:

- $m_\theta(s) = x(s)^\top \beta$, indicates a linear dependency with a generic covariate $x(s)$
- $w(s)$ is a latent process
- $\epsilon(s) \sim$ iid $N(0, \tau^2)$ indicates an error factor

Next, it is assumed that $w(s)$ is modeled as a Gaussian process

$$w(s) \sim \mathrm{GP}\Big(0, C_\theta(\cdot, \cdot)\Big),$$

where C_θ is the covariance function that describes how correlation decays with time.
Following the Bayesian formulation, the parameters θ will have a certain prior distributions:

$$\theta \sim p(\theta) \tag{3.4}$$

which reflects prior knowledge about the parameters.
From these two statements it's possible to create two different models: a "bulkier" one, called **latent GP model** and a lighter one called **response GP model**[24].
The latent GP model comes straight from the above formulation:

$$y(s) \sim \mathrm{GP}(m_\theta(s) + w(s), \tau^2 I_n) \tag{3.5}$$

While the response GP model comes from conditioning the observations $y(s)$ over the Gaussian process $w(s)$, therefore removing the need to explicitly model it, obtaining:

$$y(s) \sim \mathrm{GP}(m_\theta(s), C_\theta(\cdot, \cdot) + \tau^2 I_n) \tag{3.6}$$

The actual difference between the two formulations lies in the covariance matrix of $w(s)$.
Since this work focuses on the more parsimonious response NNGP model, the construction of this simplification is explained in more depth. Given the starting process $y(s)$, this can be modeled as:

$$y(s) \sim \text{GP}(m_\theta(s), \{C_\theta(\cdot,\cdot) + \tau^2 I_n\}^*) \tag{3.7}$$

where

$$\{C_\theta(\cdot,\cdot) + \tau^2 I_n\}^* = (I - A)^T D^{-1}(I - A)$$

To construct the matrices A and D used in the latter formulation, let $N(s_i)$ denote the set of at most M closest time indexes precedent to s_i.
For $i > 1$, the i^{th} row of A has nonzero entries at the positions indexed by $N(s_i)$. These nonzero entries are calculated by

$$A(i, N(s_i)) = C_\theta(s_i, N(s_i))\Big(C_\theta(N(s_i), N(s_i)) + \tau^2 I\Big)^{-1}. \tag{3.8}$$

The ith diagonal element of D is defined as

$$D(i,i) = C_\theta(s_i, s_i) + \tau^2 - C_\theta(s_i, N(s_i))\Big(C_\theta(N(s_i), N(s_i)) + \tau^2 I\Big)^{-1} C_\theta(N(s_i), s_i). \tag{3.9}$$

Doing this, even if the dimension of the dataset grows, all the computations regarding the covariance matrix can be simplified, restricting them to only the non-zero neighboring location for each index.
Such entries are interpreted as the weights obtained by predicting $y(s_i)$ based on the values of $y(s)$ at the time indexes in$N(s_i)$. This technique is called **kriging**[25].
The diagonal elements in D instead, represent the variance of $y(s_i)$ conditioned on its neighbors in the "past" $y(N(s_i))$.

3.2 Generic Stan Model Assumption

The objective of this work is taken as demonstrating the potential of the Nearest Neighbor approximation when applied to Gaussian processes.

To this end, a statistical generative model was developed, capable of synthesizing a sample time series that closely resembles the real-world hourly consumption dataset introduced in the previous chapter.

A key feature that favored the use of the Nearest Neighbors approximation was its ability to reduce the complexity of the model, thereby permitting the implementation of a more sophisticated distance function that better describes the underlying trends in the data.

Based on the results obtained in the previous section and guided by common domain knowledge, several key assumptions regarding the behavior of the hourly energy consumption of a generic user were made.
In particular, given the energy consumption of a user during a given hour on a generic day, the following considerations were taken as the starting point for the generative model:

- The consumption is assumed to be similar to, or at least influenced by, the consumption observed during preceding hours;
- Given that a 24-hour pattern was observed in the real-world data, in agreement with empirical knowledge, it is assumed that the consumption tends to resemble that registered on previous days at the same hour;
- A stronger similarity was noted during weekly cycles; consequently, it is assumed that the consumption tends to mirror that registered at the same hour of the same weekday on previous weeks, thereby highlighting a weekly pattern.

These assumptions were drawn from the observation that energy consumption generally follows user habits and tends to exhibit repetitive patterns.
Moreover, under these assumptions, the NNGP approximation is considered particularly well-suited to this scenario.

3.3 NNGP Model

To tailor the NNGP model on the energy consumption dataset, an ad-hoc covariance matrix has been created.
Such matrix is based on three different distance functions, whose purpose is to mimic each of the trends observed in the original data and used as assumptions.

Given s_i and s_j two possible hour-samples, they covariance is then defined as:

$$\begin{aligned} C_\theta(s_i, s_j) = \sigma^2 \exp\Big(- \phi \cdot \mathrm{Dist}_{\mathrm{lin}}(s_i, s_j)\Big) \\ + \sigma_{24}^2 \exp\Big(- \phi \cdot \mathrm{Dist}_{24}(s_i, s_j)\Big) \\ + \sigma_{\mathrm{sett}}^2 \exp\Big(- \phi \cdot \mathrm{Dist}_{\mathrm{sett}}(s_i, s_j)\Big), \quad s_i, s_j \in S, \end{aligned}$$

where the distances are defined as:

- $\mathrm{Dist}_{\mathrm{lin}} = |s_i - s_j|$;
- $\mathrm{Dist}_{24} = \min\left(\mathrm{Dist}_{\mathrm{lin}} \mod 24, 24 - (\mathrm{Dist}_{\mathrm{lin}} \mod 24)\right)$;
- $\mathrm{Dist}_{\mathrm{sett}} = \min\left(\mathrm{Dist}_{\mathrm{lin}} \mod 168, 168 - (\mathrm{Dist}_{\mathrm{lin}} \mod 168)\right)$,

For what concerns the other parameters below there is a brief introduction, but they will be better explained later in this chapter:

- σ standard deviation for the samples close in time, it defines their order of variability
- σ_{24} standard deviation for the samples registered at the same hour but on preceding days, it defines their order of variability
- σ_{sett} standard deviation for the samples registered at the same hour of the same weekday but on preceding weeks, it defines their order of variability
- ϕ decides how fast the linear distance's contribution decays
- ϕ_{24} decides how fast the daily distance's contribution decays
- ϕ_{sett} decides how fast the weekly distance's contribution decays

3.3.1 Implementation of the Model

To apply the Nearest Neighbors approximation for the covariance matrix, a set of matrices is defined as follows:

- `NN_matrix`: In the i-th row, the values of the neighbors corresponding to the i-th sample are stored in order.

- `NN_idx_matrix`: In the i-th row, the indices of the neighbors corresponding to the i-th sample are stored in order.
- `NN_dist_matrix`: In the i-th row, the linear distances between the i-th sample (s_i) and its neighbors (s_j) are stored. This distance depends solely on time:
$$\text{Dist}_{\text{lin}} = |s_i - s_j|;$$
- `NN_dist_matrix_24`: In the i-th row, the circular distances modulo 24 (representing daily cycles) between the i-th sample and its neighbors are stored. This distance is defined as:
$$\text{Dist}_{24} = \min\Big(\text{Dist}_{\text{lin}} \bmod 24,\ 24 - \big(\text{Dist}_{\text{lin}} \bmod 24\big)\Big);$$
- `NN_dist_matrix_sett`: In the i-th row, the circular distances modulo 168 (representing weekly cycles) between the i-th sample and its neighbors are stored. This distance is defined as:
$$\text{Dist}_{\text{sett}} = \min\Big(\text{Dist}_{\text{lin}} \bmod 168,\ 168 - \big(\text{Dist}_{\text{lin}} \bmod 168\big)\Big).$$

In the Nearest Neighbors approximation, it is assumed that each sample is influenced only by a selected subset of samples defined as its "neighbors," rather than by the entire dataset.
This assumption serves to reduce the computational complexity of the model. For the purpose of modeling hourly energy consumption, the neighbors of a given sample s_i are classified into the following categories:

- **Immediate Past Neighbors**: A sample s_j is classified as an immediate past neighbor if $j = i - k$ for $k = 1, \ldots, 18$.
- **Previous Day Neighbors**: A sample s_j is classified as a previous day neighbor if $j \in \{i - 25,\ i - 26,\ i - 27\}$.
- **Weekly Cycle Neighbors**: A sample s_j is classified as a weekly cycle neighbor if $j \in \{i - (7 \cdot 24 + 1),\ i - (7 \cdot 24 + 2),\ i - (7 \cdot 24 + 3)\}$.

Thus, for each sample, given that N is the length of the time series, the full $\mathbb{R}^{N \times N}$ distance matrix is approximated by a reduced $\mathbb{R}^{N \times \texttt{num\_neighbors}}$ matrix.

3.4 Choice of the priors

An important step in constructing every Bayesian model is the selection of prior distributions.
In theory, given a well-structured model and unlimited computing power, the definition of the priors may not be critical.
However, when computational resources are limited, the proper setup of priors becomes fundamental for achieving acceptable results within a short time span.
Consequently, common domain knowledge and statistical theory have been applied to define these priors, while still keeping them non informative.
The following subsections describe the choices made for the different groups of parameters.

3.4.1 Priors for σ, σ_{24}, and $\sigma_{\mathbf{sett}}$

It is worth noting that the priors have been applied to the squared transformations of each σ parameter.
Domain knowledge suggests that most household energy contracts impose a limit of 3 kW/h. Hence, the range of possible values for the variance parameters has been constrained accordingly.

- A strict condition (> 0) has been forced for all the σ parameters to ensure that the variance is non-negative.
- Based on the 3 kW/h limit, the maximum variance in a time series would occur if consumption alternates between 0 and 3 kW/h, resulting in a variance of 9 kW/h. However, since a logarithmic transformation has been applied to the series, the maximum variation becomes $\log(9) \sim 2$.

Consequently, the priors have been defined as:

$$\sigma_{\text{sq}} \sim \text{inv\_gamma}(1, 1);$$

$$\sigma_{\text{24sq}} \sim \text{inv\_gamma}(1, 1);$$

$$\sigma_{\text{settsq}} \sim \text{inv\_gamma}(1, 1).$$

This configuration ensures that the probability density is concentrated around the desired value range without rendering the priors overly informative (see Fig. 3.1).

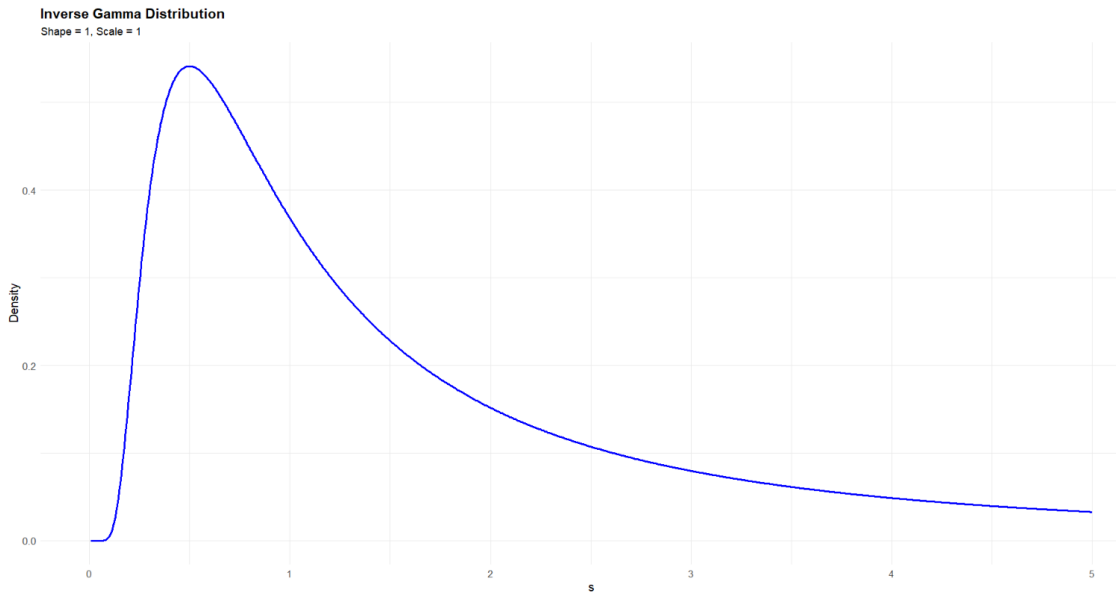


Figure 3.1: Inverse Gamma Prior Density

3.4.2 Priors for ϕ, ϕ_{24}, and ϕ_{sett}

The selection of the priors for the parameters ϕ is guided by their role in the covariance matrix, where they control the rate of decay of the dependency between two samples.
These parameters determine the strength of the contribution of the distance function to the covariance, and their ranges were chosen based on the desired model behavior in limit cases rather than on specific domain knowledge.
Therefore, a Uniform(a, b) distribution was selected for each parameter.
The ranges for a and b were set by analyzing the model's behavior at the limit cases:

- **Linear Dependency (ϕ)**: The parameter ϕ influences the linear dependency between samples according to

$$\exp\Big(-\phi \cdot \text{Dist}_{\text{lin}}(s_i, s_j)\Big).$$

The maximum linear distance between two samples corresponds to the length of the series (365 days × 24 hours), and the minimum distance is 1 (excluding zero).
Therefore, the prior for ϕ is defined as:

$$\phi \sim \text{Uniform}\Big(\frac{3}{365 \cdot 24}, 3\Big).$$

The lower bound ensures that the correlation remains strong for samples close in time and decays to approximately $e^{-3} \approx 0.05$ at a one-year distance:

$$\exp\Big(-\frac{3}{365 \cdot 24} \cdot 365 \cdot 24\Big) \to e^{-3}.$$

The upper bound ensures rapid decay such that even at the minimum distance the correlation is weak.
If simulation results yield a value near the upper bound, it would indicate that the linear dependency is not significantly impacting the model.

- **Daily Dependency (ϕ_{24})**: The parameter ϕ_{24} regulates the daily dependency between samples via

$$\exp\Big(-\phi_{24} \cdot \text{Dist}_{24}(s_i, s_j)\Big).$$

In this formulation, the maximum daily distance between two samples is 12 (due to the modulo 24 operation), while the minimum distance is 1 (excluding zero).
Thus, the prior for ϕ_{24} is defined as:

$$\phi_{24} \sim \text{Uniform}\Big(\frac{3}{24}, 3\Big).$$

The lower bound ensures strong correlation for samples corresponding to the same hour on different days, with the correlation decaying to nearly $e^{-3} \approx 0.05$ at a 12-hour distance:

$$\exp\Big(-\frac{3}{24} \cdot 12\Big) \to e^{-\frac{3}{2}}.$$

The upper bound ensures rapid decay even at the minimum distance, implying a weak daily dependency if selected by the simulation.

- **Weekly Dependency (ϕ_{sett})**: The parameter ϕ_{sett} governs the weekly dependency between samples, as expressed by:

$$\exp\Big(-\phi_{\text{sett}} \cdot \text{Dist}_{\text{sett}}(s_i, s_j)\Big).$$

Here, the maximum weekly distance (after the modulo 168 operation) is 84, and the minimum distance is 1 (excluding zero).
Consequently, the prior for ϕ_{sett} is defined as:

$$\phi_{\text{sett}} \sim \text{Uniform}\Big(\frac{3}{24 \cdot 7}, 3\Big).$$

The lower bound ensures strong correlation for samples corresponding to the same hour in different weeks, decaying to nearly $e^{-3} \approx 0.05$ at a 12-hour distance:

$$\exp\Big(-\frac{3}{24 \cdot 7} \cdot 12\Big) \to e^{-\frac{3}{14}}.$$

The upper bound ensures that even at the minimum distance the correlation remains weak.
If simulation results yield a value near this upper bound, it would indicate that the weekly dependency is not strongly influencing the model.

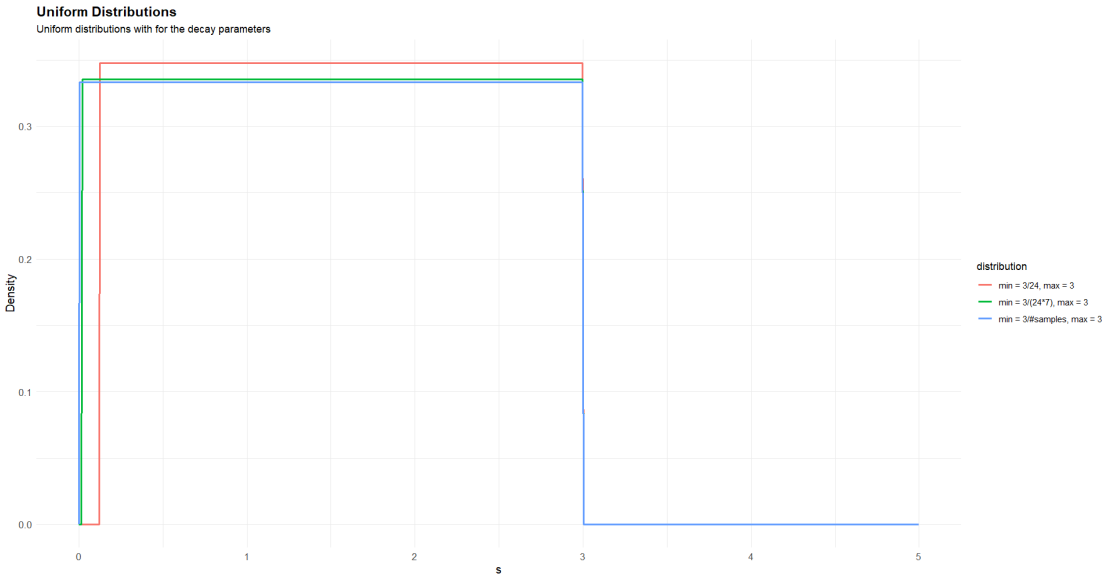


Figure 3.2: Prior distributions for the ϕ parameters

3.4.3 Priors for β

Since the mean of the Gaussian process has been set to zero, the offset parameter β determines the overall mean of the model.
Domain knowledge suggests that, because common household energy contracts typically limit consumption to 3 kW/h, the model's mean should not exceed this value.
Moreover, experimental results confirmed that the simulation runtime was unaffected by the choice of the mean prior—neither a highly informative prior nor a broad, generic one significantly altered performance.
Therefore, a very broad prior has been selected for β:

$$\beta \sim \text{Norm}(0{,}100).$$

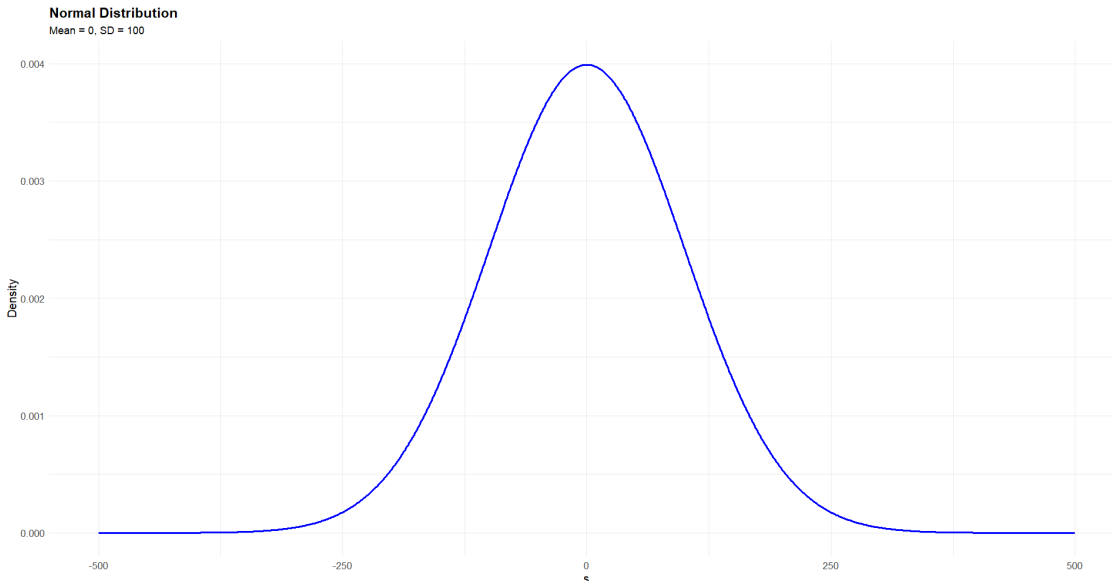


Figure 3.3: Prior density for β

3.4.4 Priors for τ

For the variance parameter τ, the prior has been applied to its squared transformation.
Although no specific domain knowledge is available for τ, it represents a variance and thus must adhere to the same non-negativity constraints as the σ parameters.
Consequently, the same non-informative prior used for the σ parameters has been chosen:

$$\tau \sim \text{inv\_gamma}(1, 1).$$

Chapter 4

Synthetic Data Generation

4.0.1 Approach

In a first attempt to validate the model presented, the latter has been applied to a synthetic series based on the original one.
To create this series the following step have been taken:

- Apply a log transformation to the data: this step was necessary since the original data is always non-negative, while the Gaussian distribution can take values in all $\mathbb{R}$;
- Compute the mean of the original series;
- Set an initial estimate of the same parameters used in the model (ϕ, σ, ϕ_{24}, etc.);
- Create the covariance matrix: the covariance is time-dependent and defined as

$$\begin{aligned} C_\theta(s_i, s_j) = \sigma^2 \exp\Big(- \phi \cdot \mathrm{Dist}_{\mathrm{lin}}(s_i, s_j)\Big) \\ + \sigma_{24}^2 \exp\Big(- \phi \cdot \mathrm{Dist}_{24}(s_i, s_j)\Big) \\ + \sigma_{\mathrm{sett}}^2 \exp\Big(- \phi \cdot \mathrm{Dist}_{\mathrm{sett}}(s_i, s_j)\Big), \quad s_i, s_j \in S, \end{aligned} \tag{4.1}$$

  where this implementation will be explained in depth in the next chapter;
- Draw samples from a multivariate normal distribution with the mean vector and covariance matrix defined above.

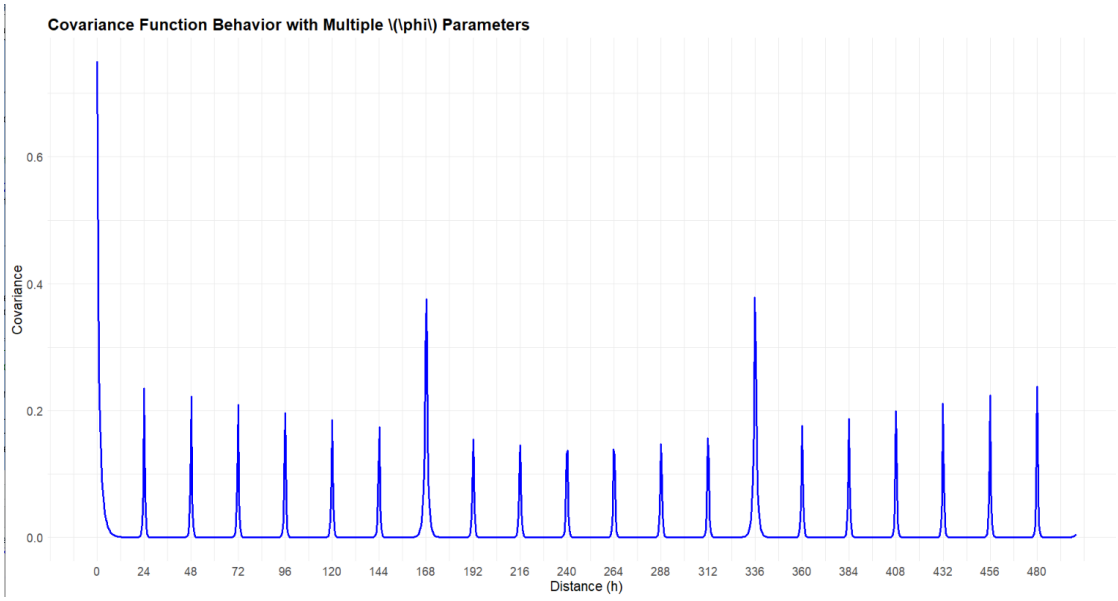


Figure 4.1: Plot of the chosen covariance function behavior

4.0.2 Parameter choice

To generate the model, the following setup was chosen:

Parameter	Value	Parameter	Value
σ^2	0.5	ϕ_{24}^2	2.5
σ_{24}^2	0.5	ϕ_{sett}	1
σ_{sett}^2	0.5	τ^2	1
ϕ	0.5	β	-1.5

Table 4.1: Parameter choices used in the Bayesian analysis

Since the aim of the synthetic data was just validating the model's parameters, the initialization wasn't optimized.
What has drive this particular setup was obtaining three distance functions enough different to have distinct behaviors.
This choice, as can be seen in Fig.(4.2) permits to easily distinguish the three contributions to the variance.

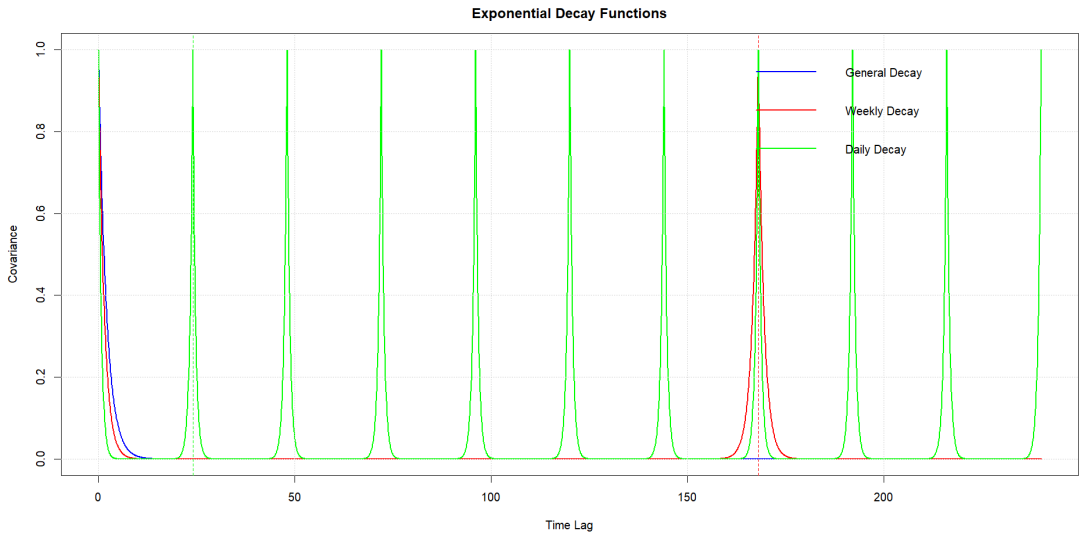


Figure 4.2: The plot shows the behavior of the three distance function used inside the covariance of the model. The less they overlap, the easier it should be to see their influence on the simulations.

4.0.3 Results and comparison with original series

The original and synthetic series were compared using their respective autocorrelation functions (ACFs), as this is the simplest method to check if the two series share similar behavior. The analysis can be divided into two parts, as shown in Fig. (4.4):

- **Short-Term Patterns**: The synthetic series replicates the short-term behavior of the original series, having a peak in the first few indices and various spikes at lag 24;

- **Long-Term Patterns**: Over longer lags (over 168 hours, a week), the synthetic series exhibits a weekly correlation pattern. However, the periodicity is less distinct compared to the original series, as the correlation weakens with increasing lag.

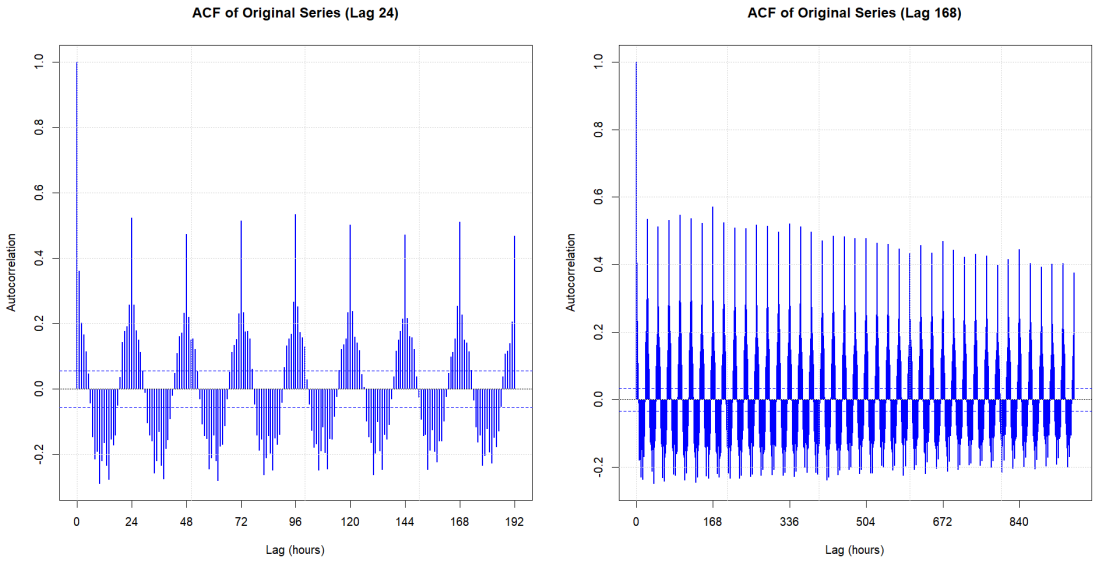


Figure 4.3: ACF plots with lag 24 (left) and 168 (right) of the original series of energy consumption data.

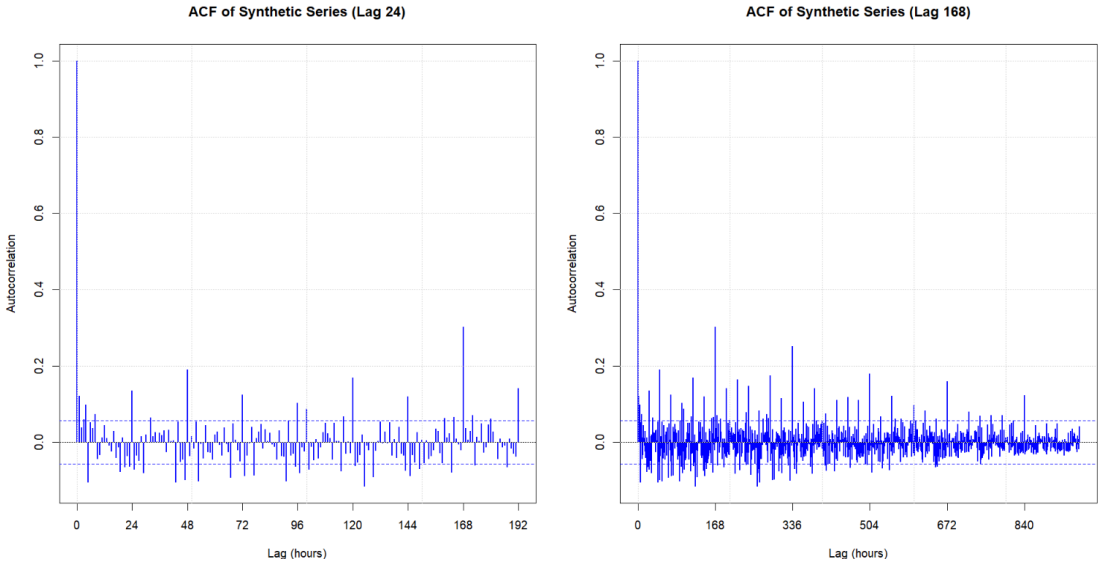


Figure 4.4: ACF plots with lag 24 (left) and 168 (right) of synthetic data, with 1200 samples generated.

- Top Row (ACF over 5 weeks): The ACF starts high and gradually decreases, showing weekly repetitive pattern. Visible 128-hour correlation.

The periodicity isn't as clear as in the original series, as it becomes less correlated as lag increases.

- Bottom Row (ACF over shorter lags): visible peaks for the daily, 24-hours correlations

In both cases the peaks seem to be on spot, but the overall sinusoidal trend isn't visible in the synthetic series.

Chapter 5

Results

In this chapter the validation of the proposed model and its application on the real-world dataset are presented.
Fine tuning and optimization were required to obtain satisfactory results with the available computational resources, which won't be shown in this work.
The results are organized as follows:

- Visualization of the simulated parameter series;
- Comparison of the simulated parameter distributions with their true values;
- Visual comparison between the reference series and the simulated series;
- Analysis of lag trends using lag plots;

5.1 Application to Synthetic Data

As a preliminary step, the model was validated against the synthetic series generated using known parameter values.
Due to the model's complexity and the limitations of an 8-core machine, the total number of iterations was limited to 2000.
It is important to note that a Stan model does not output a single final value; rather, it generates a series of samples that represent possible outcomes.
Over iterations, these samples should converge toward the optimal values, forming a dense cloud of points around them.

This validation is crucial because it allows for direct comparison between the simulated parameter series and the true values used to generate the synthetic data.
Moreover, this phase enabled significant computational savings by allowing fine tuning before applying the model to the real dataset.

5.1.1 Parameter Series Comparison

The effectiveness of the model has been first assessed by examining the behavior of the simulated parameter series.
Two key evaluations are performed:

- **Series Analysis:** Evaluating the spread of each parameter's series to determine whether additional iterations or fine tuning are required;
- **Distribution Comparison:** Comparing the resulting distribution of each parameter with the true value used to generate the synthetic data;

Figure 5.1 shows the evolution of all model parameters during the iterations.

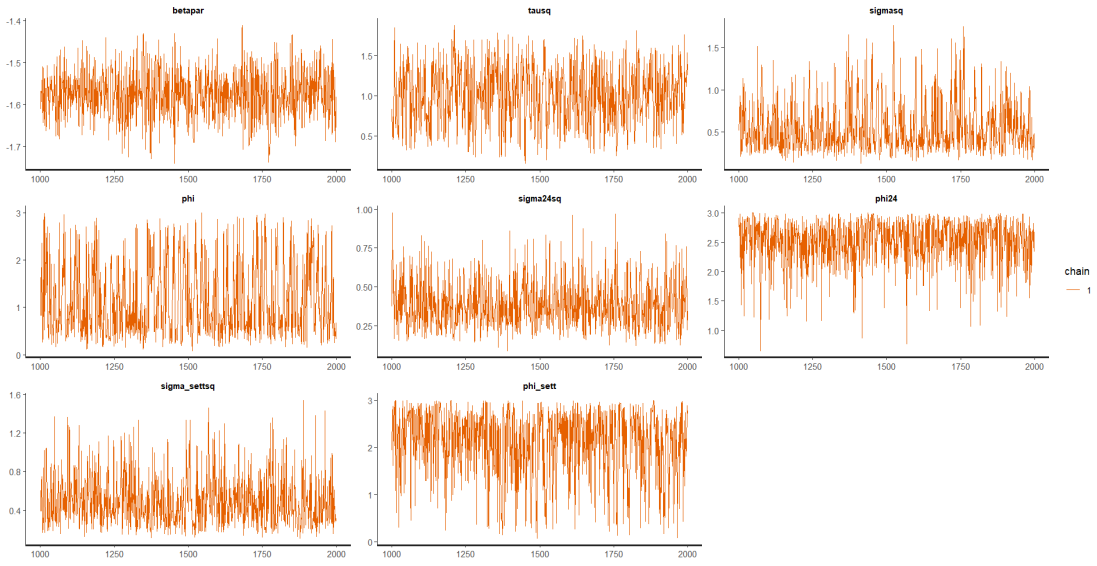


Figure 5.1: Evolution of model parameters over iterations.

Key observations include:

- For parameters like σ^2_{24}, the series exhibit small fluctuations remaining within a narrow value window;

- For others, such as ϕ_{sett}, the dispersion is higher, although a good portion of the samples is already concentered around a specific window;

The primary goal of this first section is to ensure that the true values used in the synthetic series fall within the range spanned by the sampled values. The following summarizes the comparison for each parameter: The following summarizes the comparison for each parameter:

- β: The 90% confidence interval for β is $[-1.5838, -1.5758]$, which securely includes the true value $\beta = -1.57$.

- τ^2: The 95% confidence interval for τ^2 is $[1.0039, 1.0637]$. Meanwhile the 97.5% is $[0.9995, 1.067]$ and includes the true value $\tau^2 = 1$. Both the intervals are very similar, meaning that the samples are tightly distributed;

- σ^2: The 95% confidence interval for σ^2 is $[0.5083, 0.5610]$. The true value $\sigma^2 = 0.5$, is slightly falls slightly off of it.

- σ^2_{24}: The 95% CI for σ^2_{24} is $[0.3755, 0.4043]$, excluding the true value $\sigma^2_{24} = 0.5$.

- σ^2_{sett}: The 95% confidence interval for σ^2_{sett} is $[0.4606, 0.5021]$, which includes the true value $\sigma^2_{\text{sett}} = 0.5$.

- ϕ: The 95% CI for ϕ is $[1.0368, 1.1695]$, which does not include the true value $\phi = 0.5$. In this case the distribution results to be quite sparse.

- ϕ_{24}: The 90% confidence interval for ϕ_{24} is $[2.466, 2.551]$, securely including the true value $\phi_{24} = 2.5$.

- ϕ_{sett}: The 95% CI for ϕ_{sett} is $[2.0745, 2.1977]$, which excludes the true value $\phi_{\text{sett}} = 1$. In this case the prediction is quite off track.

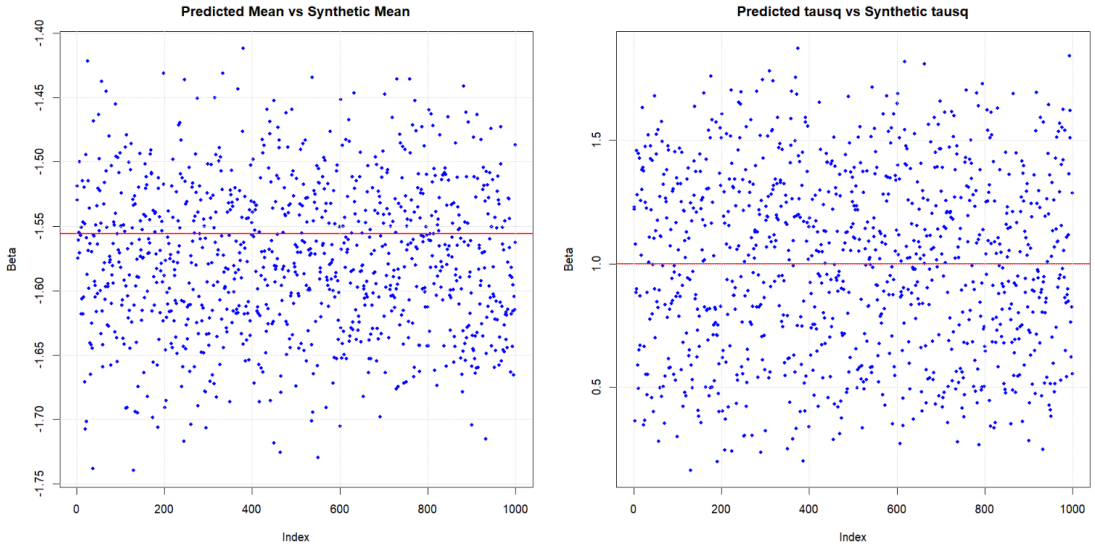


Figure 5.2: Plot of the samples simulated in stan for μ and τ^2. The real value for both parameter is shown as a vertical red line.

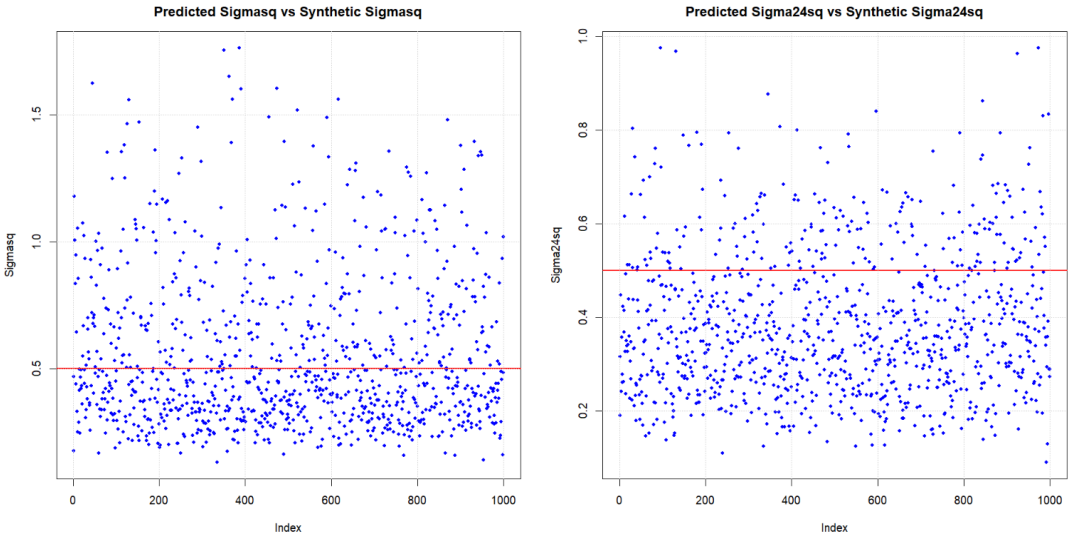


Figure 5.3: Plot of the samples simulated in stan for σ^2 and σ^2_{24}. The real value for both parameter is shown as a vertical red line.

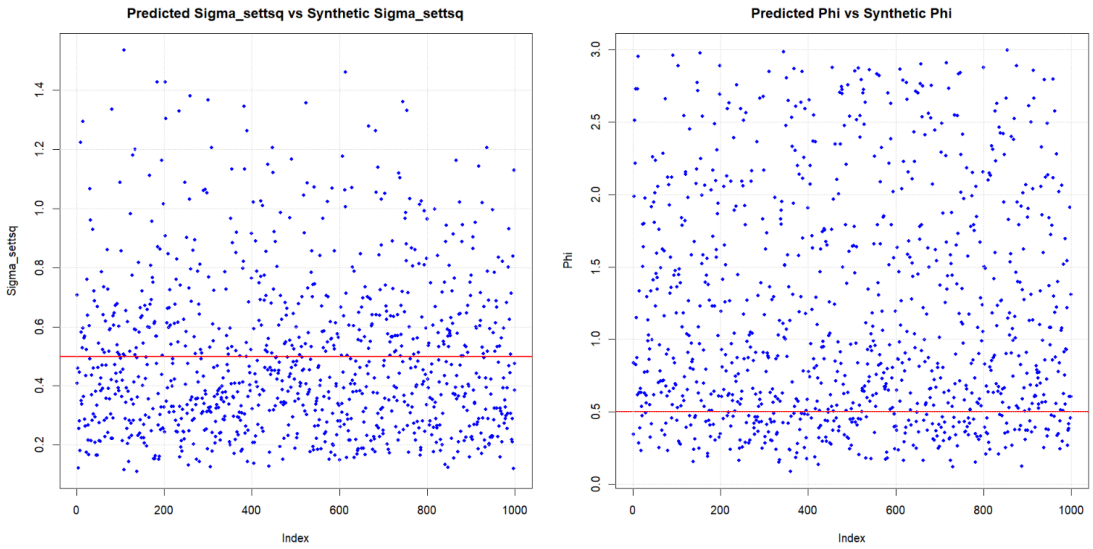


Figure 5.4: Plot of the samples simulated in stan for σ^2_{sett} and ϕ. The real value for both parameter is shown as a vertical red line.

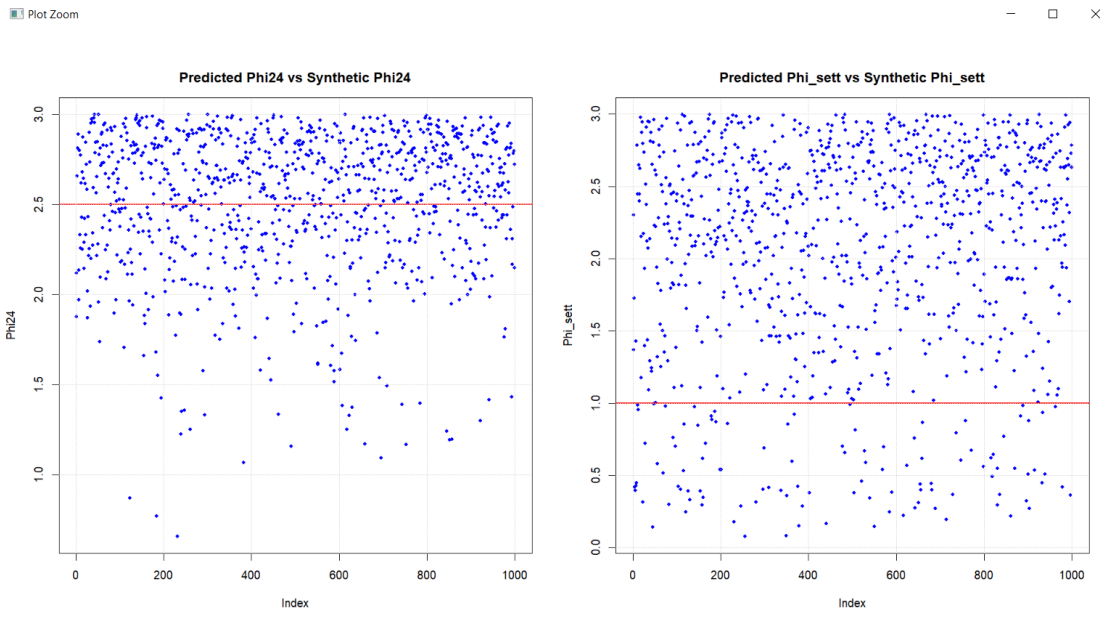


Figure 5.5: Plot of the samples simulated in stan for ϕ_{sett} and ϕ_{24}. The real value for both parameter is shown as a vertical red line.

In summary, the parameter analysis confirms that the true values used to

generate the synthetic series are, within reasonable margins, encompassed by the sampled values.
Despite some parameters exhibiting a wide dispersion or minor fluctuations, the overall behavior of the model is considered adequate for further analysis and for comparison with the synthetic data.

5.1.2 Visual Comparison

Before comparing the synthetic and generated series visually, again it is necessary to give a brief explanation of the Stan model output and how the subsequent comparison process has been outlined.
As noted earlier, Stan produces candidate values for the parameters at each iteration rather than a single final value. The focus here is on generating a noise-free series by drawing new samples from the Gaussian process $w(s)$ conditioned on the observed data.
ù Since both the synthetic series $y(s)$ and $w(s)$ are Gaussian, the posterior distribution after conditioning is given by:

$$w(s) \mid y(s) \sim \text{Norm}\Big(\mu_{w(s)|y(s)}, K_{w(s)|y(s)}\Big),$$

where

- $\mu_{w(s)|y(s)} = \mu_w + \text{Cov}_w \, \text{Cov}_y^{-1} \, (y - \mu_y - \mu_w)$,
- $K_{w(s)|y(s)} = \text{Cov}_w - \text{Cov}_w \, \text{Cov}_y^{-1} \, C_w$.

In this case, since $\mu_w = 0$, the mean simplifies to:

$$\mu_{w(s)|y(s)} = \text{Cov}_w \, \text{Cov}_y^{-1} \, (y - \mu_y),$$

with:

- $\mu_w = 0$, consistent with the assumed distribution of $w(s)$;
- Cov_w defined by the custom covariance function for the energy consumption dataset;
- $\text{Cov}_y = \text{Cov}_w + \tau^2 I_n$, since the covariance of the sum of two independent Gaussian processes is the sum of their covariances;

The covariance remains:

$$K_{w(s)|y(s)} = \mathrm{Cov}_w - \mathrm{Cov}_w \, \mathrm{Cov}_y^{-1} \, C_w.$$

For visual comparison, a single "representative" series is extracted from the posterior $w(s) \mid y(s)$ by taking the mean of all draws, denoted as w_{mean}. Figure 5.6 displays w_{mean} alongside the synthetic series.
The two series are very similar, although w_{mean} appears less dispersed and more stable.

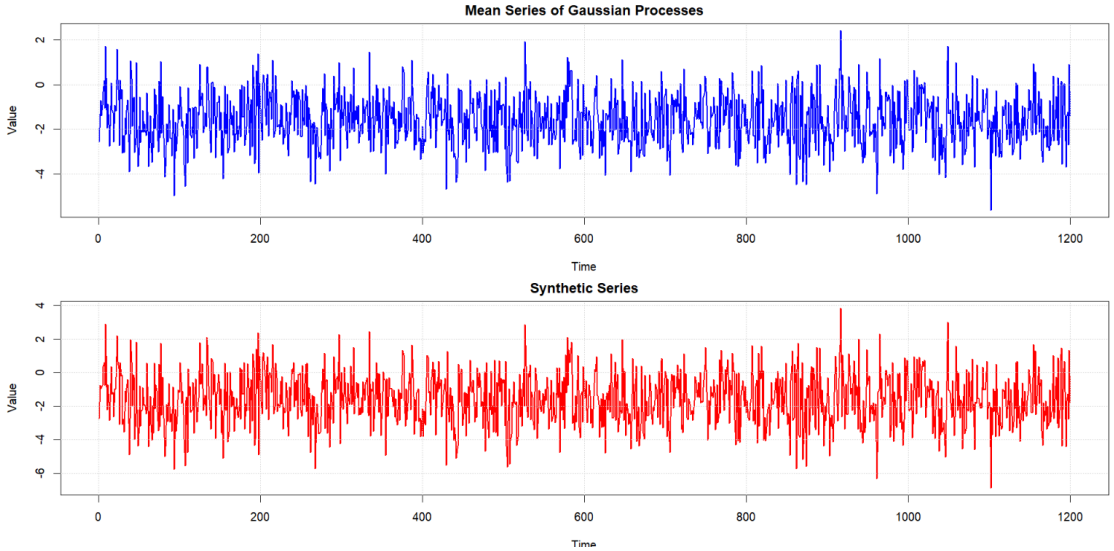


Figure 5.6: Comparison between w_{mean} and the synthetic series.

Lag plots further validate the similarity in behavior between the synthetic series and w_{mean}.
Figures 5.7 and 5.8 illustrate that both series exhibit comparable underlying patterns, particularly with respect to the 24-hour and 168-hour cycles.

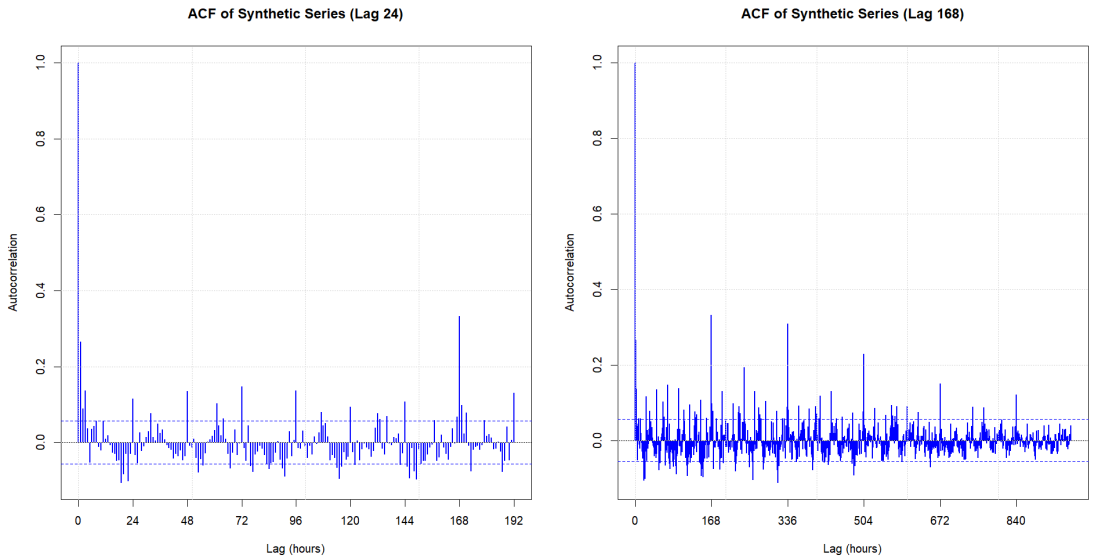


Figure 5.7: ACF plots with lag 24 (left) and 168 (right) of the synthetic series

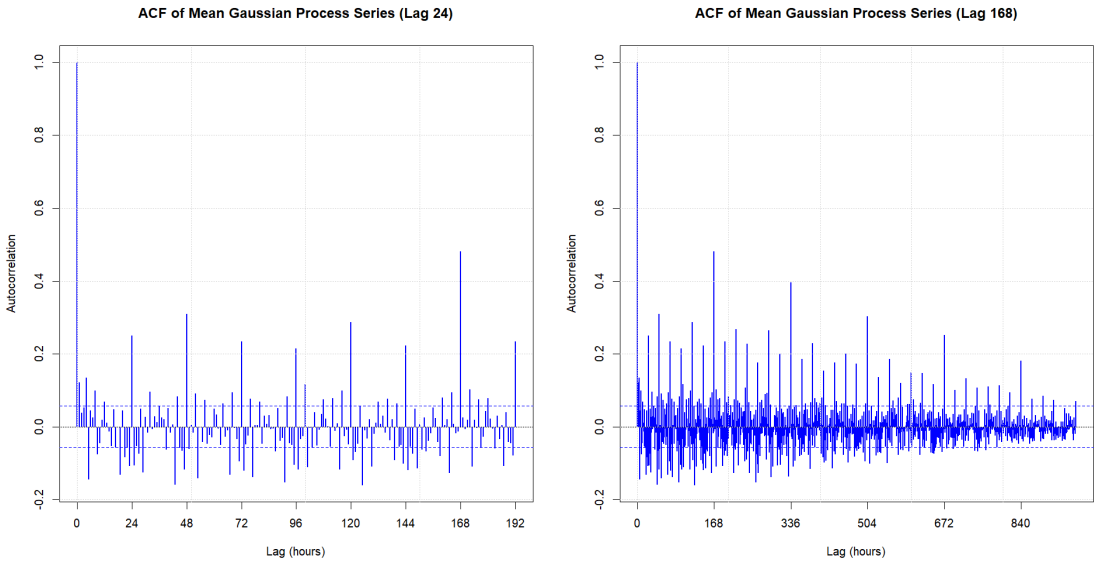


Figure 5.8: ACF plots with lag 24 (left) and 168 (right) of w_{mean}, showing refined and repetitive patterns

5.2 Applying the Model on the Real Data

After validating the model on synthetic data, the final step involves applying it to the original dataset. The objective is to verify whether the generated representative series, w_{mean}, exhibits the same key features as the real data.

5.2.1 Parameter Series Comparison

When comparing with the original series, no "true" parameter values exist, so the evaluation focuses solely on the behavior of the parameter series.
In this case, the parameter series appear more stable, with reduced fluctuations compared to the synthetic validation.
This increased stability can be attributed to both a higher number of iterations (5000 versus 2000) and a real dataset that is three times longer than the synthetic one.

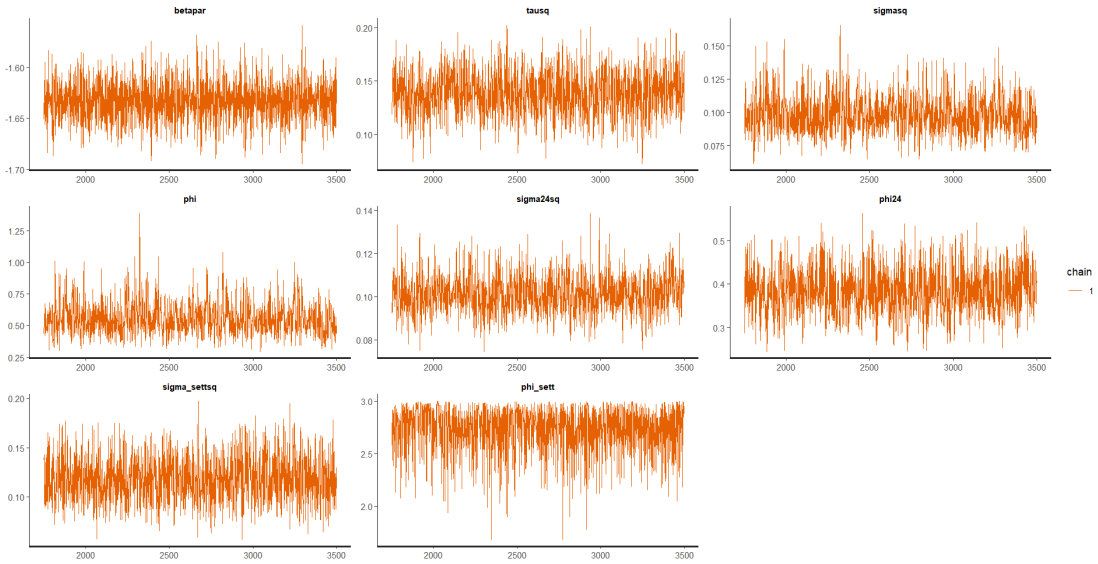


Figure 5.9: Parameter series for the real dataset.

5.2.2 Visual Comparison

For visual evaluation, the original series is compared with the representative series w_{mean}.
The simulated series closely mirrors the original, albeit with a more stable

appearance.
To illustrate this, two views are provided:

- A truncated view that enhances clarity by focusing on a segment of the series (Figure 5.10).

- A full view of the series (Figure 5.11).

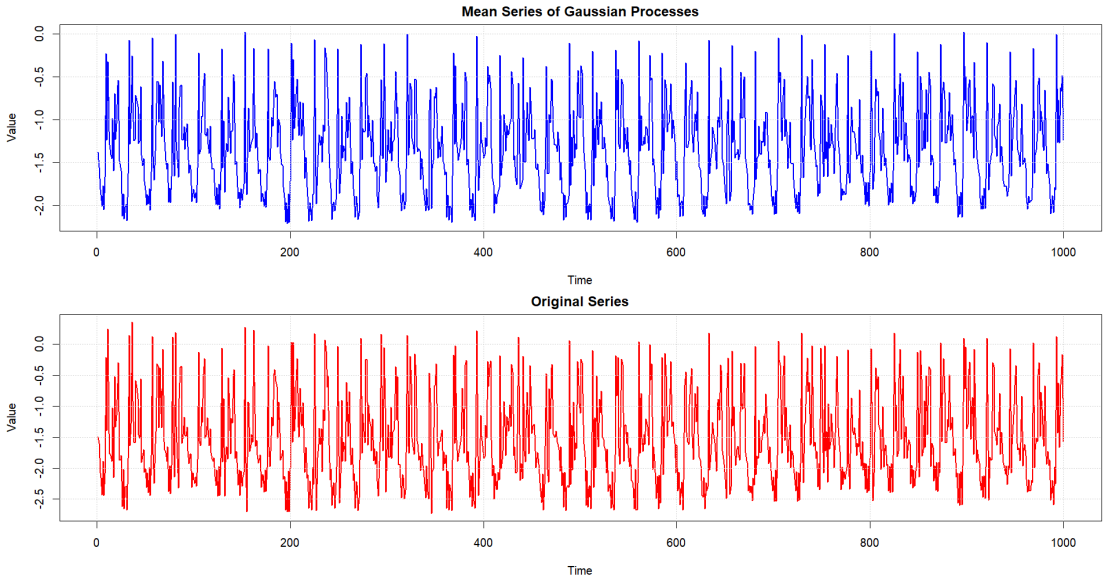


Figure 5.10: Plot of the original and simulated series. To improve the clarity and visualize the similarities between the two, the series have been truncated to the first 1000 samples.

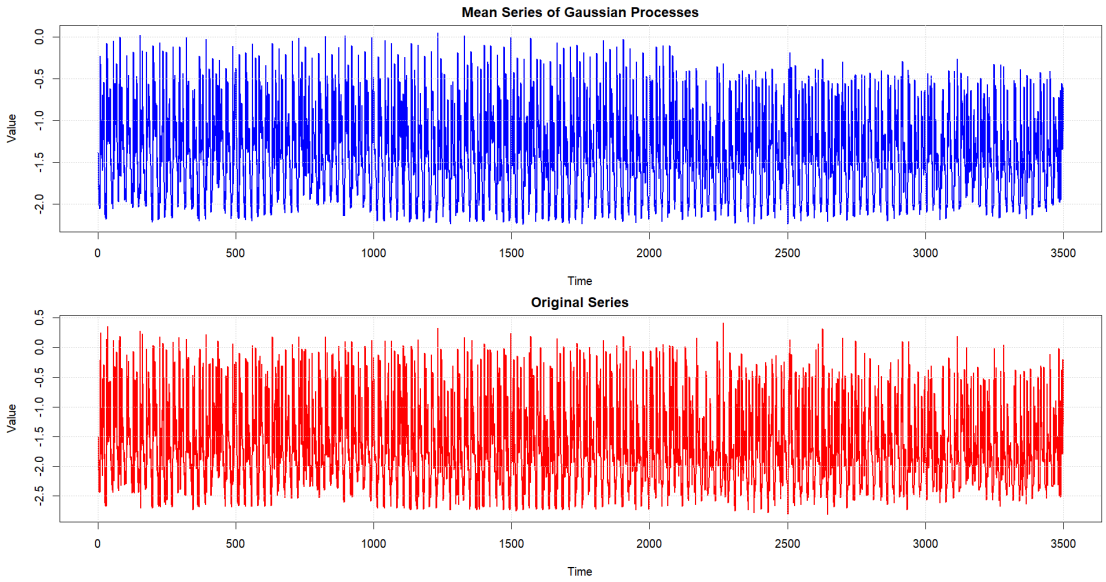


Figure 5.11: Full plot of the original and simulated series.

Lag plots further confirm that the underlying behavior of the series is consistent.
Although both series display similar patterns, the lag plot of w_{mean} exhibits more pronounced peaks, likely due to the noise reduction in its computation.

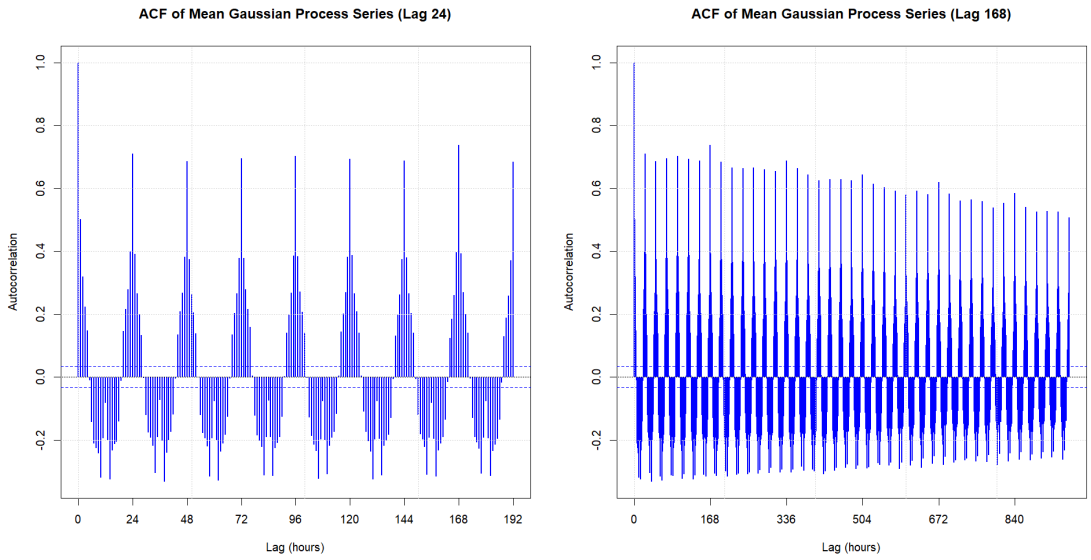


Figure 5.12: ACF plots with lag 24 (left) and 168 (right) of w_{mean}

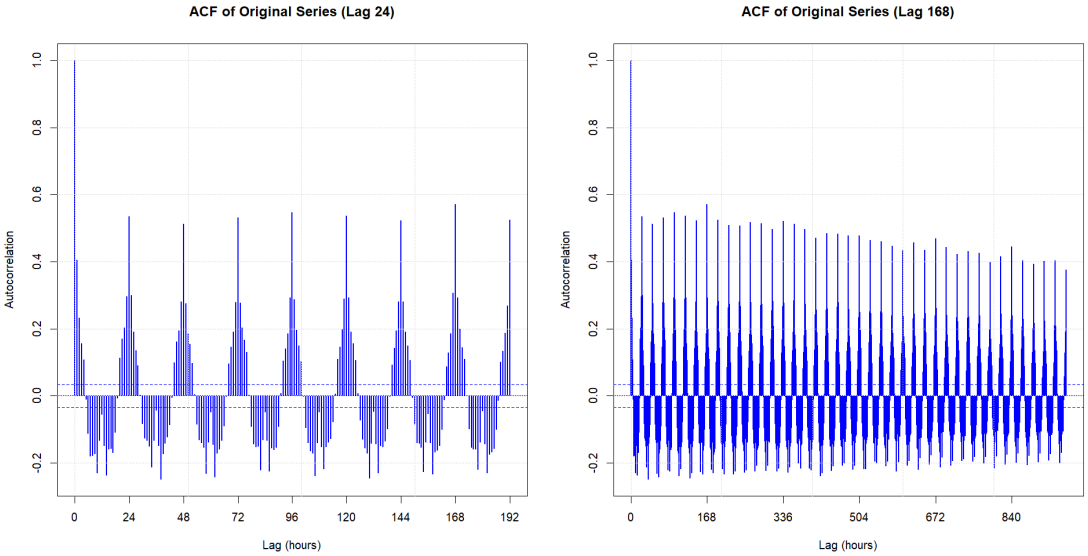


Figure 5.13: ACF plots with lag 24 (left) and 168 (right) of the original energy consumption dataset

5.3 Conclusion

In this work, a generative Nearest Neighbors Gaussian Process (NNGP) model was developed to simulate hourly energy consumption profiles.
The model was designed to capture three different temporal patterns: recent, daily, and weekly.
This have been done by employing a custom covariance function and applying the Nearest Neighbors approximation on Gaussian processes, which has significantly reduced the computational complexity.
Initially the model has been validated on synthetic data.
The analysis of parameter series, distributions, and lag plots demonstrated convergence towards the true values used in data generation.
When applied to the real-world dataset, the representative series w_{mean} closely mirrored the original consumption patterns, showing improved stability and clearer cyclical trends.
These results confirm the model's effectiveness in capturing the intrinsic behavior of energy consumption.
Future work may focus on further refining the model parameters and extending the approach to larger, more diverse datasets.
Additionally, this approach could be followed by a clustering algorithm, in

order to not only be able to create data depending on a particular user, but mroe in general on a specific customer-category.

Chapter 6

Code

6.1 Generation of synthetic data

Listing 6.1: Set up

```
1
2 # Read the dataset
3 data <- read_csv("pinerolo/measurements.csv")
4
5 # Log-transform the data
6 log_series <- log(data$dom2_10[1:1200])
7 time_index <- seq_along(log_series)
8
9 # Series parameters
10 N <- length(log_series)
11 mean_estimate <- mean(log_series)
12
13
14
15 # Paramters configuration
16 phi <- 0.5
17 phi24 <- 2.5
18 phi_weekly <- 1
19
20 sigma_sq <- 0.5
21 sigma24_sq <- 0.5
22 sigma_weekly_sq <- 0.5
23 tausq <- 1
```

Listing 6.2: Covariance Matrix

```r
# Function to compute the covariance for the dataset. It just depends on the time indexes
compute_covariance <- function(N, sigma_sq, phi, sigma24_sq, phi24, sigma_weekly_sq, phi_weekly) {

  # Generate distance indices
  distance_matrix <- abs(outer(1:N, 1:N, "-"))

  # Compute individual distance matrices
  D_linear <- distance_matrix
  D_daily <- pmin(distance_matrix %% 24, 24 - (distance_matrix %% 24))
  D_weekly <- pmin(distance_matrix %% 168, 168 - (distance_matrix %% 168))

  # Compute the covariance matrix
  cov_matrix <- sigma_sq * exp(-phi * D_linear) +
    sigma24_sq * exp(-phi24 * D_daily) +
    sigma_weekly_sq * exp(-phi_weekly * D_weekly)

  return(cov_matrix + diag(tausq, N, N))
}
```

Listing 6.3: Synthetic Data Generation

```r

set.seed(98989)

# Create mean vector and compute covariance matrix
mean_vector <- rep(mean_estimate, N)
cov_matrix <- compute_covariance(N, sigma_sq, phi, sigma24_sq, phi24, sigma_weekly_sq, phi_weekly)

# Generate synthetic series using multivariate normal distribution
synthetic_series <- mvrnorm(
  n = 1,
  mu = mean_vector,
  Sigma = cov_matrix
)
```

6.1.1 Stan model on the synthetic data

6.1.2 Set up variables and matrices for stan model call

Listing 6.4: Distance matrices

```
create_matrices <- function(X, Y, k_neighbors) {
  n <- length(X)

  # The distances for the nearest neighbors you want
  num_neighbors <- length(k_neighbors)

  # Initialize the matrices
  NN_matrix <- matrix(0, nrow = n - 1, ncol = num_neighbors)
  NN_idx_matrix <- matrix(0, nrow = n - 1, ncol = num_neighbors)
  NN_dist_matrix <- matrix(0, nrow = n - 1, ncol = num_neighbors
    )
  NN_dist_matrix_24 <- matrix(0, nrow = n - 1, ncol = num_
    neighbors)
  NN_dist_matrix_sett <- matrix(0, nrow = n - 1, ncol = num_
    neighbors)
  # Loop through each point (xi, yi)
  for (i in 2:n) {
    neighbors <- c()
    neighbor_indices <- c()
    neighbor_distances <- c()
    neighbor_distances_24 <- c()
    neighbor_distances_sett <- c()

    h <- 0
    for (j in 1:num_neighbors) {
      idx <- i - k_neighbors[j]
      if (idx > 0) {
        h <- h + 1
        neighbors[h] <- Y[idx]
        neighbor_indices[h] <- idx
        # Calculate the distance
        xj <- X[idx]
        xi <- X[i]
        distance_lin <- abs(xi - xj)
        distance_24 <- pmin(distance_lin %% 24, 24 - (distance_
    lin %% 24))
        distance_sett <- pmin(distance_lin %% 168, 168 - (
    distance_lin %% 168))
        neighbor_distances <- append(neighbor_distances,
    distance_lin)
```

```
35          neighbor_distances_24 <- append(neighbor_distances_24,
    distance_24)
36          neighbor_distances_sett <- append(neighbor_distances_
    sett, distance_sett)
37        }
38
39
40        # Fill the matrices
41        NN_matrix[i - 1, 1:length(neighbors)] <- neighbors #
    observation value
42        NN_idx_matrix[i - 1, 1:length(neighbor_indices)] <-
    neighbor_indices # observation neighbors indexes
43        NN_dist_matrix[i - 1, 1:length(neighbor_distances)] <-
    neighbor_distances
44        NN_dist_matrix_24[i - 1, 1:length(neighbor_distances_24)]
    <- neighbor_distances_24
45        NN_dist_matrix_sett[i - 1, 1:length(neighbor_distances_
    sett)] <- neighbor_distances_sett
46      }
47    }
48
49
50    return(list(NN_matrix = NN_matrix, NN_idx_matrix = NN_idx_
    matrix, NN_dist_matrix = NN_dist_matrix, NN_dist_matrix_24 =
    NN_dist_matrix_24, NN_dist_matrix_sett = NN_dist_matrix_sett)
    )
51  }
```

Listing 6.5: Compute distances matrices

```
1  y <- synthetic_series # synthetic series data
2  x <- 1:length(y)
3
4  # set up which indexes are defined as neighbors
5  k_neighbors <- c(
6    1, 2, 3, 4, 5, 6,
7    24 * 1 + 1, 24 * 1 + 2, 24 * 1 + 3,
8    7 * 24 + 1, 7 * 24 + 2, 7 * 24 + 3, 7, 8, 9, 10, 11, 12, 13,
     14, 15, 16, 17, 18
9  )
10
11 n_neigh <- 11
12
13 # Compute Distances
14 result <- create_matrices(x, y, k_neighbors)
15
```

```
# Extract the values of the single matrices
NN_matrix <- result$NN_matrix[, 1:n_neigh]
NN_idx_matrix <- result$NN_idx_matrix[, 1:n_neigh]
NN_dist_matrix <- result$NN_dist_matrix[, 1:n_neigh]
NN_dist_matrix_24 <- result$NN_dist_matrix_24[, 1:n_neigh]
NN_dist_matrix_sett <- result$NN_dist_matrix_sett[, 1:n_neigh]
```

Listing 6.6: Neighbor's distances

```
compute_NN_distM <- function(NN_idx_matrix,NN_dist_matrix, NN_
    dist_matrix_24, NN_dist_matrix_sett) {
compute_NN_distM <- function(NN_idx_matrix, NN_dist_matrix, NN_
    dist_matrix_24, NN_dist_matrix_sett) {
  n <- dim(NN_idx_matrix)[1]

  num_neighbors <- length(k_neighbors)


  NN_distM <- matrix(NA, nrow = n, ncol = num_neighbors * (num_
    neighbors - 1) / 2)
  NN_distM_24 <- matrix(NA, nrow = n, ncol = num_neighbors * (
    num_neighbors - 1) / 2)
  NN_distM_sett <- matrix(NA, nrow = n, ncol = num_neighbors * (
    num_neighbors - 1) / 2)

  for (i in 1:n) {
    h <- 1
    for (j in 2:num_neighbors) {
      xi <- NN_idx_matrix[i, j] # select current value of the
    neighbor, for us i = X[i]

      if (!is.na(xi)) {
        for (k in 1:(j - 1)) {
          xk <- NN_idx_matrix[i, k]

          if (!is.na(xk)) {
            distance_lin <- abs(xi - NN_idx_matrix[i, k]) #
    neighbors - (x[i]-j) which is the value of the other neighbor
            NN_distM[i, h] <- distance_lin
            NN_distM_24[i, h] <- pmin(distance_lin %% 24, 24 - (
    distance_lin %% 24))
            NN_distM_sett[i, h] <- pmin(distance_lin %% 168, 168
     - (distance_lin %% 168))
            h <- h + 1
          } else {
            h <- h + 1
```

```
            }
          }
        }
      }
    }

    return(list(NN_distM = NN_distM, NN_distM_24 = NN_distM_24, NN
      _distM_sett = NN_distM_sett))
}
```

Listing 6.7: Get neighbors distances

```

k_neighbors <- k_neighbors[1:n_neigh]

result2 <- compute_NN_distM(
  NN_idx_matrix,
  NN_dist_matrix,
  NN_dist_matrix_24,
  NN_dist_matrix_sett
)
# Extract distances
NN_distM <- result2$NN_distM
NN_distM_24 <- result2$NN_distM_24
NN_distM_sett <- result2$NN_distM_sett
```

Listing 6.8: Stan model set up

```

N <- length(x)
M <- length(k_neighbors)

# Set up variables
data <- list(
  N = N,
  M = M,
  Y = y,
  x_0 = rep(1, N),
  NN_ind = NN_idx_matrix,
  NN_dist = NN_dist_matrix,
  NN_distM = NN_distM,
  NN_dist_24 = NN_dist_matrix_24,
  NN_distM_24 = NN_distM_24,
  NN_dist_sett = NN_dist_matrix_sett,
  NN_distM_sett = NN_distM_sett,
  lim_phi = 3 / N,
```

```
  lim_phi24 = 3 / 24,
  lim_phi_sett = 3 / (24 * 7)
)

# Use all avaialble cores and parallelize
rstan_options(auto_write = TRUE)
options(mc.cores = parallel::detectCores())

#Set paramters that will be visualized
parameters <- c(
  "betapar", "tausq", "sigmasq", "phi", "sigma24sq",
  "phi24", "sigma_settsq", "phi_sett"
)
```

Listing 6.9: Stan Model call

```

#Set initial values for the paramters iteration
my_inits <- list(list(
  beta = -1, tau = 1, phi = phi, phi24 = phi24, phi_sett = phi_
    weekly,
  sigmasq = 1, sigma24sq = 1, sigma_settsq = 1
))

# Fit the model using stan
samples <- stan(
  file = "Model.stan",
  data = data,
  init = my_inits,
  pars = parameters,
  iter = 2000,
  chains = 1,
  seed = 123
)
```

6.1.3 Stan model implementation

Listing 6.10: Stan model, Data block

```
data {
  int<lower=1> N;
  int<lower=1> M;

  vector[N] Y;
```

```
6    vector[N] x_0;
7
8    int NN_ind[N - 1, M];
9    matrix[N - 1, M] NN_dist;
10   matrix[N - 1, (M * (M - 1) / 2)] NN_distM;
11
12
13   matrix[N - 1, M] NN_dist_24;
14   matrix[N - 1, (M * (M - 1) / 2)] NN_distM_24;
15   matrix[N - 1, M] NN_dist_sett;
16   matrix[N - 1, (M * (M - 1) / 2)] NN_distM_sett;
17
18   real lim_phi;
19   real lim_phi24;
20   real lim_phi_sett;
21 }
```

Listing 6.11: Stan model, Parameters block

```
1  parameters {
2    real<lower = 0.01> tausq;
3
4    real betapar;
5
6    real<lower=lim_phi, upper= 3>phi;
7    real<lower=lim_phi24, upper= 3> phi24;
8    real<lower=lim_phi_sett, upper= 3> phi_sett;
9
10   real<lower = 0> sigmasq;
11   real<lower = 0> sigma24sq;
12   real<lower = 0> sigma_settsq;
13 }
```

Listing 6.12: Stan model, Functions block

```
1  functions{
2    real nngp_lpdf
3    (vector Y,
4     vector mu,
5     real sigmasq,
6     real tausq,
7     real phi,
8     real phi24,
9     real phi_sett,
10    real sigma24sq,
11    real sigma_settsq,
```

```
12    matrix NN_dist,
13    matrix NN_distM,
14    matrix NN_dist_24,
15    matrix NN_distM_24,
16    matrix NN_dist_sett,
17    matrix NN_distM_sett,
18    int[,] NN_ind,
19    int N,
20    int M)
21
22    {
23      # variables instatiation
24      vector[N] V;
25      vector[N] Y_trasl = Y - mu;
26      vector[N] U = Y_trasl;
27      real vargp =  sigmasq + sigma24sq + sigma_settsq;
28      real w_phi = sigmasq/vargp;
29      real w_phi_sett = sigma_settsq/vargp;
30      real w_phi_24 = sigma24sq/vargp;
31
32      real kappa_p_1 = tausq/vargp +1;
33      int h;
34
35      for (i in 2:N) {
36       int dim = (i <= M) ? (i - 1) : M;
37
38        matrix[dim,dim] iNNdistM ;
39        matrix[dim,dim] iNNCholL ;
40        vector[dim] iNNcorr ;
41        vector[dim] v ;
42        row_vector[dim] v2;
43
44        if(dim == 1){iNNdistM[1, 1] = kappa_p_1;}
45        else{
46          h = 0;
47          for (j in 1:(dim - 1)){
48            for (k in (j + 1):dim){
49              h = h + 1;
50
51              iNNdistM[j, k] = w_phi * exp(- phi * NN_distM[(i -
    1), h])+
52                  w_phi_24*exp(- phi24 * NN_distM_24[(i - 1), h])
    +
53                  w_phi_sett * exp(- phi_sett * NN_distM_sett[(i -
     1), h]);
```

```
54
55                iNNdistM[k, j] = iNNdistM[j, k];
56              }
57            }
58
59            for(j in 1:dim){
60              iNNdistM[j, j] = kappa_p_1;
61            }
62          }
63
64          iNNCholL = cholesky_decompose(iNNdistM);
65
66          iNNcorr = to_vector(w_phi * exp(- phi * NN_dist[(i - 1),
    1: dim]) +
67                w_phi_24*exp(- phi24 * NN_dist_24[(i - 1), 1: dim])
    +
68                w_phi_sett * exp(- phi_sett * NN_dist_sett[(i - 1),
    1: dim]));
69
70
71          v = mdivide_left_tri_low(iNNCholL, iNNcorr);
72
73          V[i] = kappa_p_1 - dot_self(v);
74
75          v2 = mdivide_right_tri_low(v', iNNCholL);
76
77          U[i] = U[i] - v2 * Y_trasl[NN_ind[(i - 1), 1:dim]];
78
79        }
80        V[1] = kappa_p_1;
81
82        return - 0.5 * ( 1 / vargp * dot_product(U, (U ./ V)) + sum
    (log(V))+ N * log(vargp));
83
84      }
85
86  }
```

Listing 6.13: Stan model, Model block

```
1  model {
2    phi ~ uniform(lim_phi,3);
3    phi24 ~ uniform(lim_phi24,3);
4    phi_sett ~ uniform(lim_phi_sett, 3);
5
6    sigmasq  ~  inv_gamma(1,1);
```

```
7    sigma24sq   ~   inv_gamma(1,1);
8    sigma_settsq   ~   inv_gamma(1,1);
9
10
11   tausq ~ inv_gamma(1,1);
12   betapar ~   normal(0, 100);
13
14
15   Y ~ nngp(x_0 * betapar, sigmasq, tausq, phi, phi24, phi_sett,
       sigma24sq, sigma_settsq,
16   NN_dist, NN_distM, NN_dist_24, NN_distM_24, NN_dist_sett,
       NN_distM_sett,
17   NN_ind, N, M);
18 }
```

Bibliography

[1] R. L. Berger G. Casella. *Statistical Inference*. Duxbury, 2002 (cit. on p. 1).

[2] Simplilearn. *What Is Probability Density Function*. URL: https://www.simplilearn.com/tutorials/statistics-tutorial/probability-density-function (cit. on p. 7).

[3] xaktly. *Uniform PDF*. URL: https://xaktly.com/UniformProbDistr.html (cit. on p. 8).

[4] analyticsvidhya. *What are Mean and Variance of the Normal Distribution?* URL: https://www.analyticsvidhya.com/blog/2024/11/mean-and-variance-of-the-normal-distribution/ (cit. on p. 9).

[5] Andrew Gelman, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. *Bayesian Data Analysis*. 3rd. CRC Press, 2013 (cit. on p. 11).

[6] Jose M. Bernardo and Adrian F. M. Smith. *Bayesian Theory*. Wiley, 2000 (cit. on p. 13).

[7] Honglak Lee Chuong B. Do. «Gaussian processes». In: (2008) (cit. on pp. 13, 18).

[8] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006 (cit. on p. 15).

[9] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, 2006. URL: http://www.gaussianprocess.org/gpml/ (cit. on p. 18).

[10] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012 (cit. on p. 18).

[11] Chris Chatfield. *The Analysis of Time Series: An Introduction*. CRC Press, 2003 (cit. on p. 20).

[12] Jan G. De Gooijer and Rob J. Hyndman. «25 Years of Time Series Forecasting». In: *International Journal of Forecasting* (2006) (cit. on p. 20).

[13] Robert H. Shumway and David S. Stoffer. *Time Series Analysis and Its Applications: With R Examples*. Springer, 2017 (cit. on p. 21).

[14] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. *Time Series Analysis: Forecasting and Control*. Wiley, 2015 (cit. on p. 22).

[15] Wikipedia. *Population of Ireland and Europe 1750 to 2005*. URL: `https://commons.wikimedia.org/wiki/File:Population_of_Ireland_and_Europe_1750_to_2005.svg` (cit. on p. 22).

[16] Wikipedia. *Correlogram*. URL: `https://en.wikipedia.org/wiki/Correlogram` (cit. on p. 23).

[17] ooemma83. *How to Interpret ACF and PACF plots for Identifying AR, MA, ARMA, or ARIMA Models*. URL: `https://medium.com/@ooemma83/how-to-interpret-acf-and-pacf-plots-for-identifying-ar-ma-arma-or-arima-models-498717e815b6` (cit. on p. 24).

[18] Wikipedia. *Rstudio*. URL: `https://en.wikipedia.org/wiki/RStudio` (cit. on p. 24).

[19] International Energy Agency. *World Energy Outlook 2021*. IEA, 2021. URL: `https://www.iea.org/reports/world-energy-outlook-2021` (cit. on p. 26).

[20] Tao Hong and Shu Fan. «Probabilistic electric load forecasting: A tutorial review». In: *International Journal of Forecasting* 32.3 (2016), pp. 914–938 (cit. on p. 26).

[21] Yu Wang, Ke Li, and Peng Jiang. «Generating synthetic smart meter data for load prediction model evaluation». In: *IEEE Access* 7 (2019), pp. 9841–9850 (cit. on p. 27).

[22] A. V. Vecchia. «Estimation and Model Identification for Continuous Spatial Processes». In: (1988) (cit. on p. 45).

[23] Abhirup Datta, Sudipto Banerjee, Andrew O. Finley, and Alan E. Gelfand. «Hierarchical Nearest-Neighbor Gaussian Process Models for Large Geostatistical Datasets». In: (2016) (cit. on p. 45).

[24] Stan Development Team. «Stan Modeling Language: User's Guide and Reference Manual». In: (2017) (cit. on p. 46).

[25] M. L. Stein. «Statistical Interpolation of Spatial Data: Some Theory for Kriging». In: (1999) (cit. on p. 47).