Part I Simulation

1 Question (a)

1.1 Sample and simulate data

We define a Gaussian Process (GP) on a 2D domain $x \in [0,1] \times [0,1]$, where the latent variables follow$u \sim \mathcal{N}(0, K)$, and the covariance matrix is given by $K_{ij} = k(x_i, x_j)$. The kernel function $k(x, x')$ is a Gaussian kernel, defined as $k(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{||x-x'||}{2l^2}\right)$, where l is the length-scale parameter. The latent variables are defined on a regular $D \times D$ grid, and the observation data is generated by randomly subsampling the latent variables and adding noise. The observation model is $v = Gu + \epsilon, \epsilon \sim \mathcal{N}(0, I)$, where G is a selection matrix constructed based on the indices of the randomly subsampled points.

Use these definitions and steps to simulate the process and visualize the generated data, we can finally get the following result.

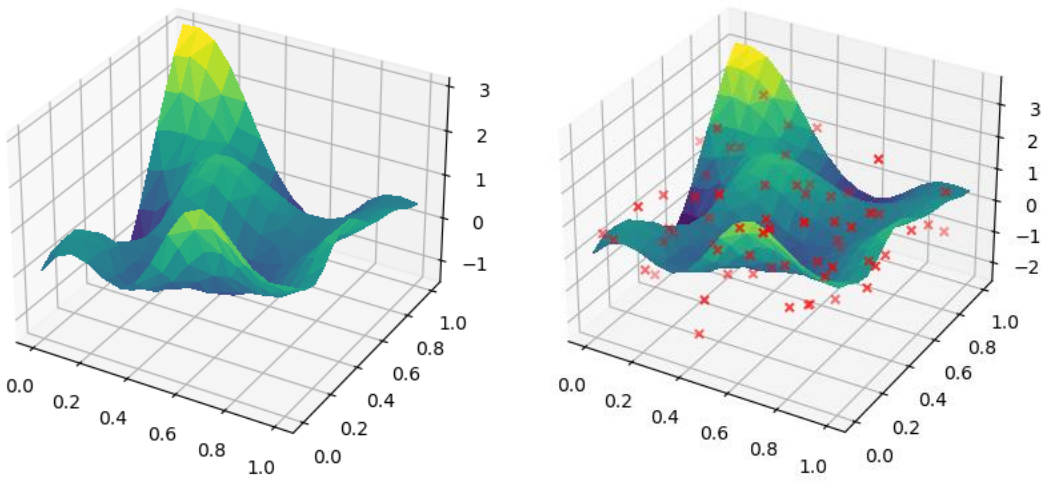
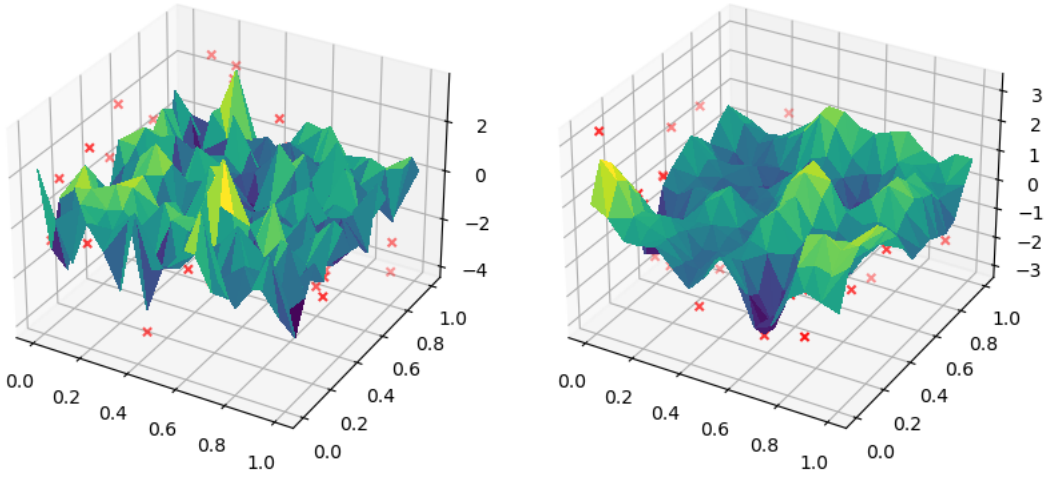


Figure 1 Original Data and generated noisy data

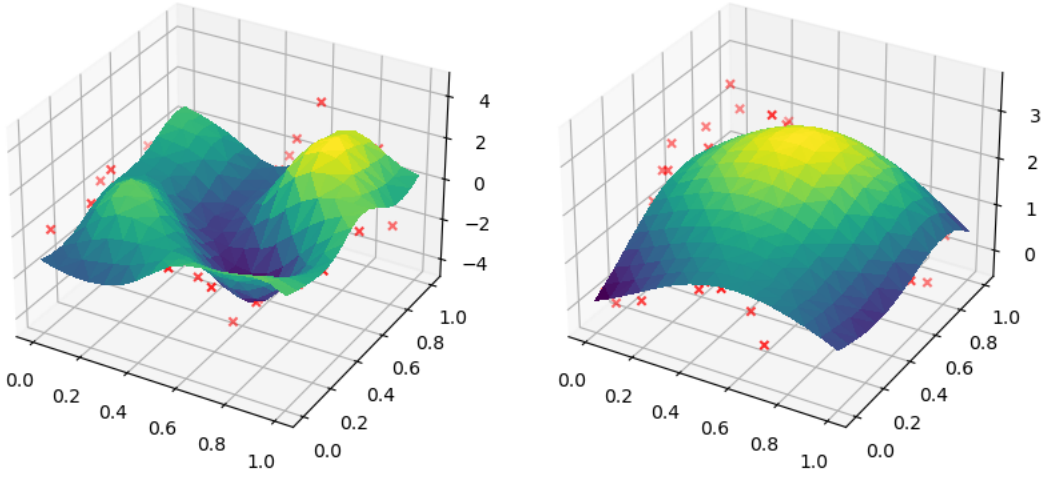
From the 3D plot, we can see that the observed data v, after sampling and adding noise, is distributed near the original data u.

1.2 Compare result with different length-scale

By varying length-scale l in Gaussian Kernel function, results as follows can be obtained.



(a) $l = 0.01$ (b) $l = 0.1$



(c) $l = 0.2$ (b) $l = 0.5$

Figure 2 GP Priors with different length-scale

Varying the length-scale l in the Gaussian kernel function has a significant impact on the generation of prior GP data. From the plots, it is evident that when l is very small, the data exhibits high variability and appears jagged, with sharp fluctuations. This is because a smaller l causes the kernel to emphasize local correlations, making the process more sensitive to small changes in input, leading to rapid oscillations. As l increases, the data gradually transforms into smoother surfaces, with variations becoming less pronounced and the overall behavior trending towards a more stable, flat pattern. A larger l induces stronger global correlations, which can be deducted from Gaussian Kernel function where only larger difference of x can generate large variance, and this smooths out the fluctuations and makes the GP more coherent across the domain.

2 Question (b)

2.1 Derive the formula from $p(u)$ and $p(v|u)$

From the definition we can get $p(u) = N(0, K) \propto e^{-\frac{1}{2}u^T K^{-1} u}$, so $log(u) = -\frac{1}{2}u^T K^{-1} u + const$. The noisy data $v = Gu + \epsilon$ is generated based on u and its probability can be written as conditional form, $p(v|u) = N(Gu, I) \propto e^{-\frac{1}{2}(v-Gu)^T(v-Gu)}$.

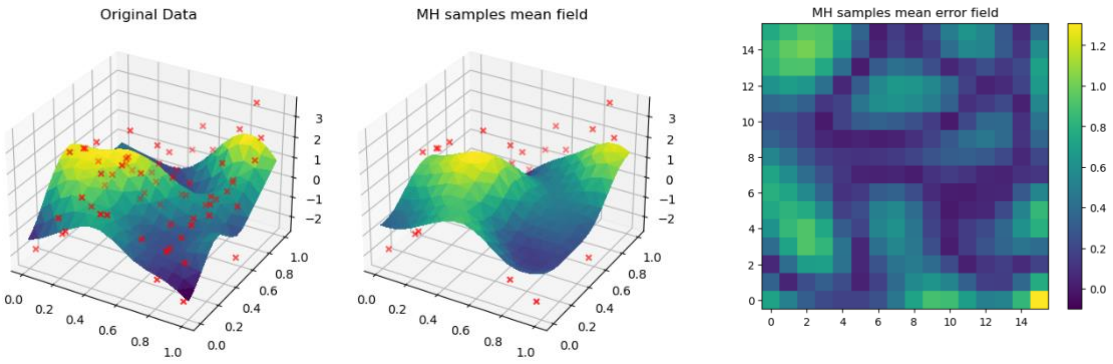
2.2 Complete the $GRW - MH$ and pCN algorithms

The Gaussian Random Walk Metropolis-Hastings algorithm. It works by proposing new samples u' based on a random perturbation of the current sample u, scaled by the prior covariance and a step-size parameter β. The proposed sample is computed as $u' = u + \beta K_c z$, where K_c is the Cholesky factor of K. The log acceptance probability is calculated based on the difference in log-target densities between the proposed and current samples. Samples are accepted or rejected based on a comparison of this probability with a uniformly distributed random number.

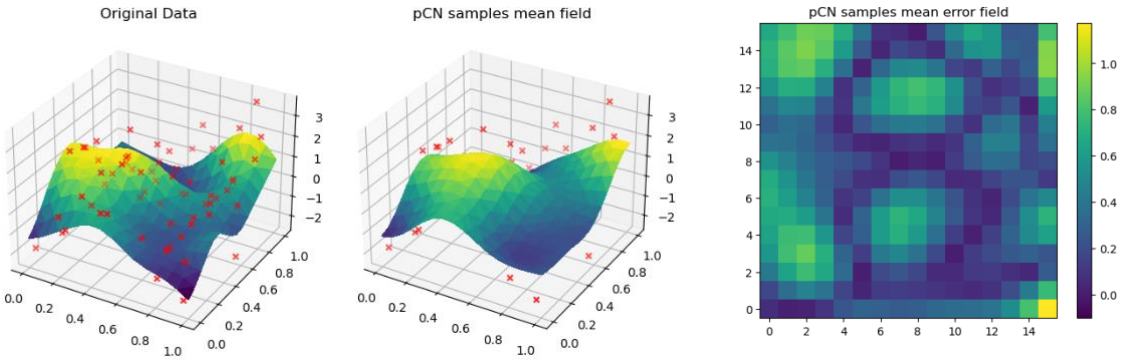
The preconditioned Crank-Nicolson algorithm is designed for sampling under Gaussian priors. Unlike GRW-MH, it directly incorporates the structure of the prior distribution into the proposal mechanism, improving efficiency. The proposed sample is generated as $u' = \sqrt{1 - \beta^2}u + \beta K_c z$, where $z \sim \mathcal{N}(0, I)$. This preserves the prior structure, ensuring that the samples remain consistent with the prior covariance K. The log acceptance probability depends solely on the log-likelihood difference between the proposed and current samples. The prior term cancels due to the symmetry of the proposal.

2.3 Sampling, visualization, and comparison

Sample from $p(u|v)$ using both algorithms, plot the mean u and the absolute error between original u and predicted u. The figures are as follows.



(a) GRW-MH samples



(b) pCN samples

Figure 3: Comparison of the mean and error from GRW-MH and pCN sampling

Both error fields exhibit similar colour contrast, indicating that the two sampling methods—GRW-MH and pCN—produce comparable results in terms of the distribution of the errors. The similar shape of the error fields suggests that both algorithms explore the same target density. However, the underlying mechanisms of the two algorithms differ. Despite the differences in sampling strategies, if enough iterations have been performed for the algorithms to converge, both methods should ultimately draw samples from the same posterior distribution. This is evidenced by the overall similarity in the error fields, reinforcing the idea that, under proper conditions, both algorithms can effectively approximate the target posterior.

2.4 Comparison of different parameters

Vary D, β and compare the result. Set $D = 4$ and $D = 16$, and β=0, 0.2, 0.4, 0.6, 0.8, 1. The results are as follows.

Table1 Experiment result for different parameter settings

		GRW-MH		pCN	
D	Beta	acceptance rate	mean error	acceptance rate	mean error
4	0	1	1.0520	1	1.1676
4	0.2	0.6619	0.4703	0.8183	0.8875
4	0.4	0.3905	0.6384	0.7037	0.4354
4	0.6	0.2092	0.7230	0.4406	0.7098
4	0.8	0.097	0.8044	0.3067	0.8760
4	1.0	0.0374	0.8901	0.0252	0.6107
16	0	1	1.1714	1	1.0055
16	0.2	0.0842	0.5087	0.4468	0.3963
16	0.4	0.003	0.5071	0.1556	0.5680
16	0.6	0	0.9734	0.0212	0.3957
16	0.8	0.0001	1.1254	0.009	0.4374
16	1.0	0	1.3353	0.0006	0.5742

For GRW-MH algorithm, in the 4-dimensional space, the acceptance rate decreases as the step size β increases. The acceptance rate drops sharply from 1.0 at $\beta = 0$ to

0.0374 at $\beta = 1.0$. This trend suggests that the algorithm becomes less efficient as the step size increases, as it is unable to propose samples that are close enough to the target distribution. For pCN, the acceptance rate is much higher than that of GRW-MH at smaller β values, with a decrease in acceptance rate as β increases. At $\beta = 1.0$, the acceptance rate for pCN is still higher than that of GRW-MH. This is because pCN uses a proposal that incorporates prior structure, allowing for more efficient exploration of the target distribution, even at larger values of β.

As the dimensionality increases to $D = 16$, the acceptance rate of GRW-MH decreases significantly across all values of β. The acceptance rate approaches zero for all β values greater than 0.2, indicating that the algorithm becomes highly inefficient in high-dimensional spaces. This is a common issue with GRW-MH in high-dimensional settings, as the random walk proposals may move too far from the target distribution, leading to frequent rejections. For pCN, the acceptance rate also decreases as β increases, but it remains considerably higher than that of GRW-MH for the same β values. Even in the high-dimensional space, pCN remains relatively efficient at proposing samples that are consistent with the posterior, largely due to the structure of the prior covariance being incorporated into the proposal.

A very small β essentially leads to the algorithm getting "stuck" in regions of high probability for long periods, failing to explore the full posterior distribution efficiently. Therefore, while the acceptance rate may be high, the samples may not adequately represent the target distribution.

2.5 Expansion

Compare pCN to GRW-MH method with different settings of latent dimension. The results are as follows.

Figure 4 Acceptance rate of different dimension

The pCN method demonstrates more stability and consistency in acceptance rates compared to the GRW-MH method, especially as the dimension D increases. While GRW-MH experiences a sharp decline in acceptance rates with higher dimensions, pCN maintains a relatively steady rate, making it more suitable for high-dimensional or complex problems. Overall, pCN offers better sampling efficiency and adaptability, particularly in large parameter spaces.

3 Question (c)

3.1 Derive $p(t|u)$

We know that $p(t|u) = \prod_{i=1}^{N} p(t_i|u)$,

$$log[u|t] = \sum_{i=1}^{N} log[p(t_i = 1|u)^{t_i} p(t_i = 0|u)^{1-t_i}]$$

$$= \sum_{i=1}^{N} t_i log[\phi(u_i)] + (1 - t_j) log[1 - \phi(u_j)]$$

Use this to complete function *log_probit_likelihood*.

3.2 Predict t_{true} using MC method

Predict $t_{ture} = probit(u)$. Using Monte-Carlo method to generate prediction:

$$p(t_i^* = 1|t) = \frac{1}{M} \sum_{j=1}^{M} p(t_i^* = 1|u^{(j)}) = \frac{1}{M} \sum_{j=1}^{M} \phi\left(u_i^{(j)}\right)$$

Use this to completed function *predict_t*. And then visualize all the related data.

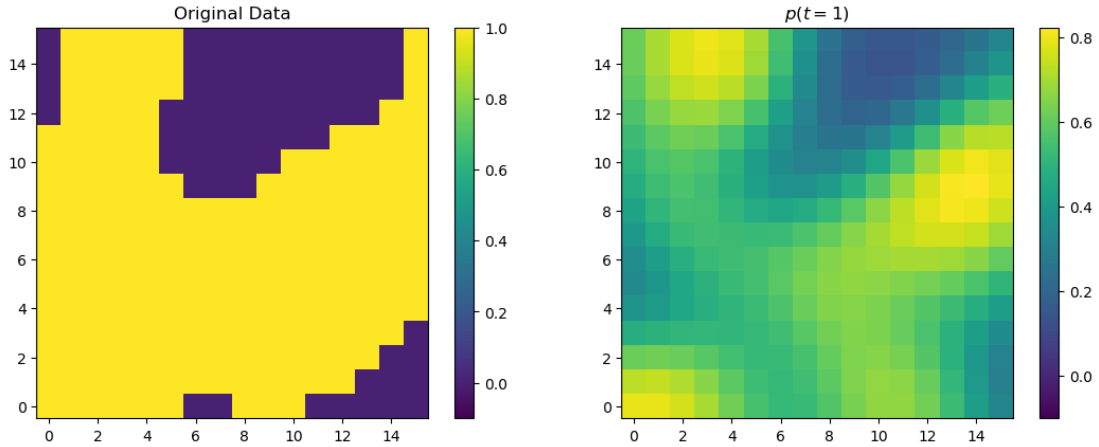


Figure 5 Original t and predicted t using MC

As shown in the figure, the colour and shape distribution of the predicted $p(t = 1)$ heatmap closely resemble those of the true assignments, indicating that the predictions align well with the actual data, suggesting that the model effectively captures the underlying patterns in the data.

4 Question (d)

The assignments obtained by using 0.5 as the probability threshold for prediction are as follows.

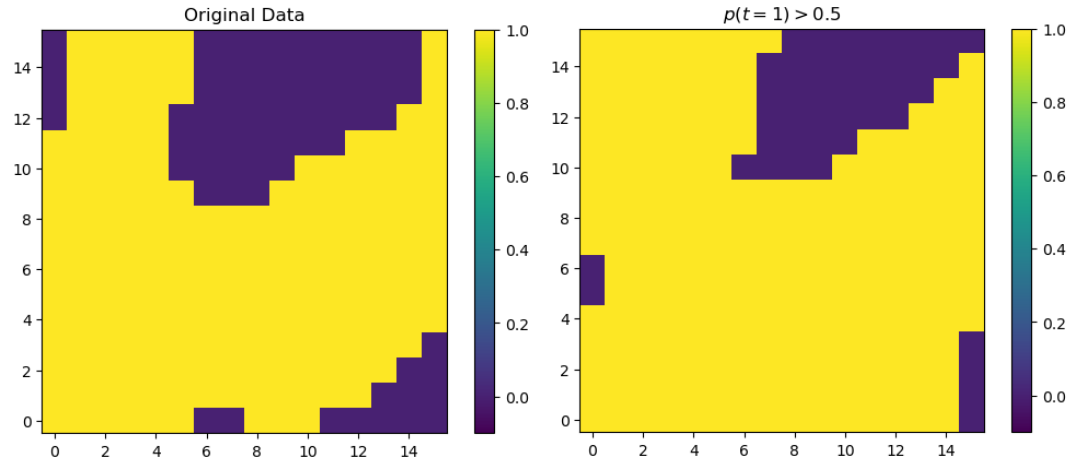


Figure 6 Comparison between original data and predicted assignments

The two results show slight differences in predictions near the boundaries between different categories, but in other regions, the predictions perform well and closely resemble the shape characteristics of the original data.

We now conduct experiments by varying l within the range of 0.01 to 10, using mean error to evaluate the performance. By comparing the average assignment errors, we aim to select the optimal parameter. The mean error data obtained during the experiments are as follows.

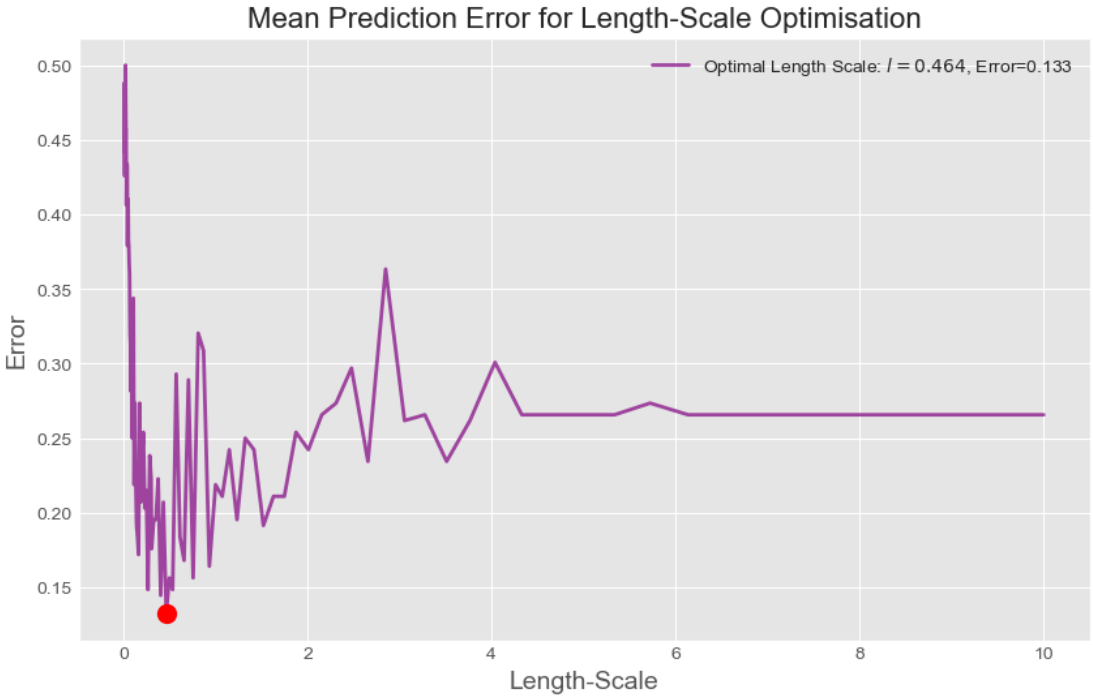


Figure 7 Mean error of different l

A large l increases the smoothness of the kernel function, causing the model to focus on global patterns while ignoring local details, which leads to higher errors. As l grows, the kernel becomes less sensitive to local variations, resulting in underfitting where the model fails to distinguish between different categories. Eventually, the error stabilizes at a higher level as the model's predictions are dominated by global trends and lose connection to the data's local structure.

The l value corresponding to the minimum error is selected for prediction. The results, including the predicted outputs, class assignments, and mean error, are visualized.

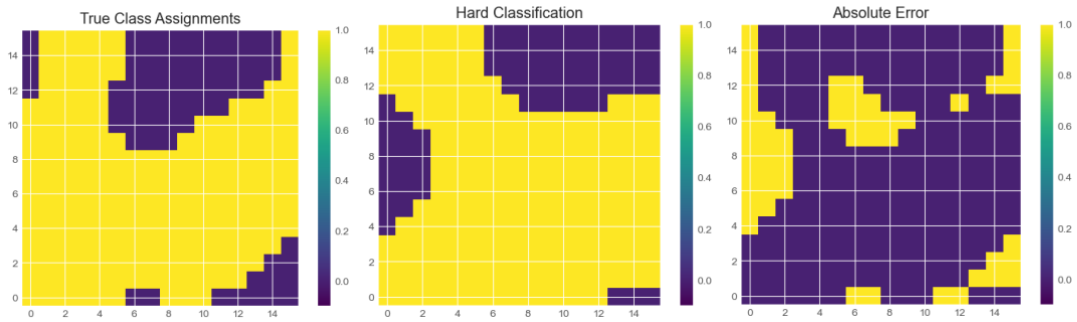


Figure 8 Optimized result and mean error

Part II Spatial Data

5 Question (e)

Derive the full Poisson log-likelihood and complete the function to calculate $p(c|u)$.

$$p(c|u) = p(c|\theta) = \prod_{i=1}^{N} p(c_i|\theta_i) = \sum_{i=1}^{N} \frac{\theta_i^{c_i} e^{-\theta_i}}{c_i!}$$

So, we can get the logarithm form of the result.

$$p(c|u) = \sum_{i=1}^{N} -\theta_i + c_i log(\theta_i) - log(c_i!)$$

And this can be used to generate data using pCN method.

6 Question (f)

6.1 MC derivation and estimation visualization

MC method can be used to estimate the expectation, derivation is as follows.

$$E_{p(c^*|c)}[c^*] = \sum_{k=0}^{\infty} k\, p(c^* = k|c) = \sum_{k=0}^{\infty} k \int p(c^* = k, u|c) du$$

$$= \sum_{k=0}^{\infty} k \int p(c^* = k|u) p(u|c) du$$

Because it has integral from, we can use Monte Carlo method to estimate it.

$$\widehat{c^*} = E_{p(c^*|c)}[c^*] \approx \frac{1}{n}\sum_{r=0}^{\infty}\sum_{j=1}^{n} r\ p\left(c^* = r|\mathrm{u}^{(\mathrm{j})}\right) = \frac{1}{n}\sum_{j=1}^{n} E_{p\left(c^*|u^{*(j)}\right)}[c^*]$$

$$= \frac{1}{n}\sum_{j=1}^{n} E_{p\left(c^*|\theta^{*(j)}\right)}[c^*] = \frac{1}{n}\sum_{j=1}^{n} \theta^{*(j)} = \frac{1}{n}\sum_{j=1}^{n} \exp\left(u^{*(j)}\right)$$

This can be used to infer the expected counts, which can then be compared with the true values. The visualizations are shown below.

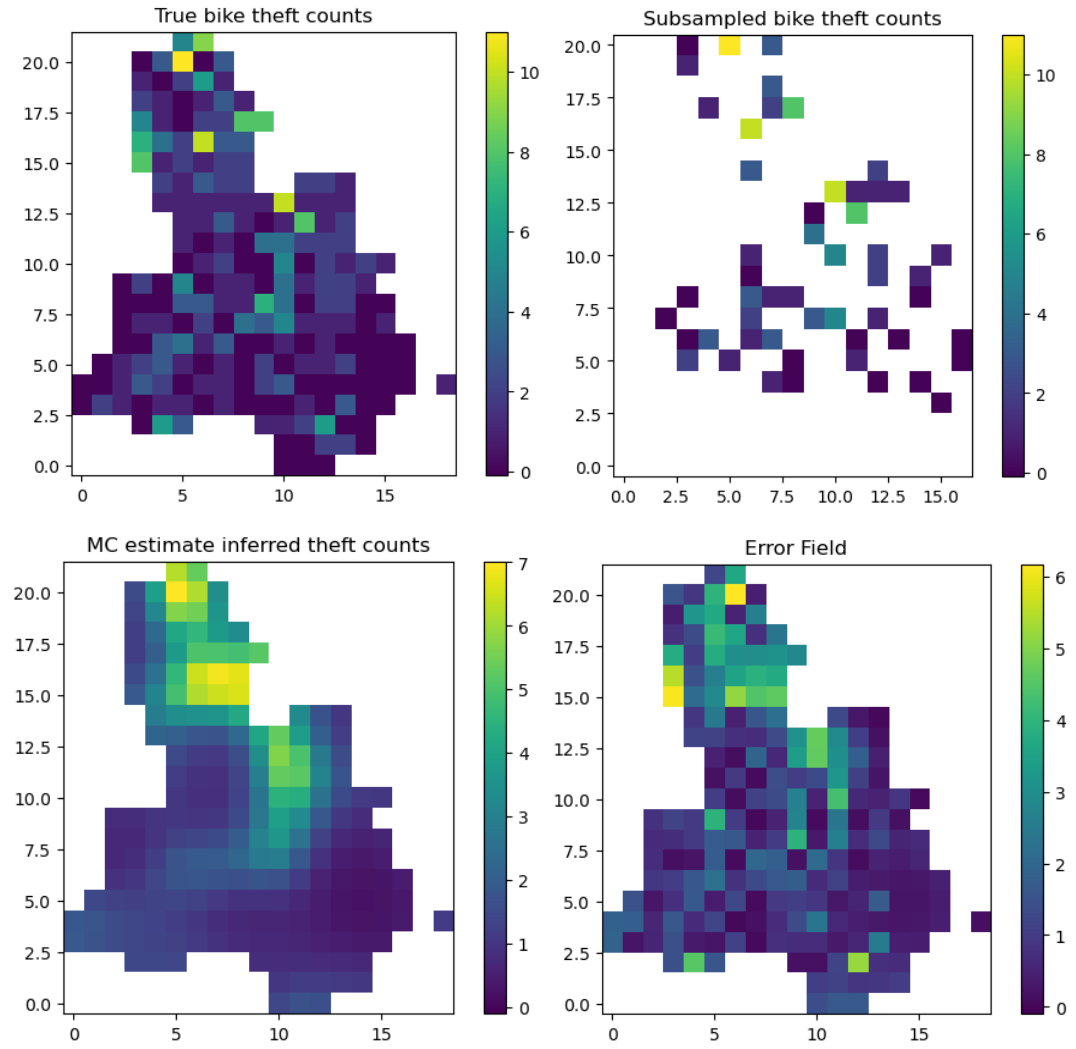


Figure 9 Original data and MC estimation

6.2 Extreme length-scale experiment

Set different length-scale from 0.01 to 10, test the prediction and calculate the mean error. The prediction errors are as follows.

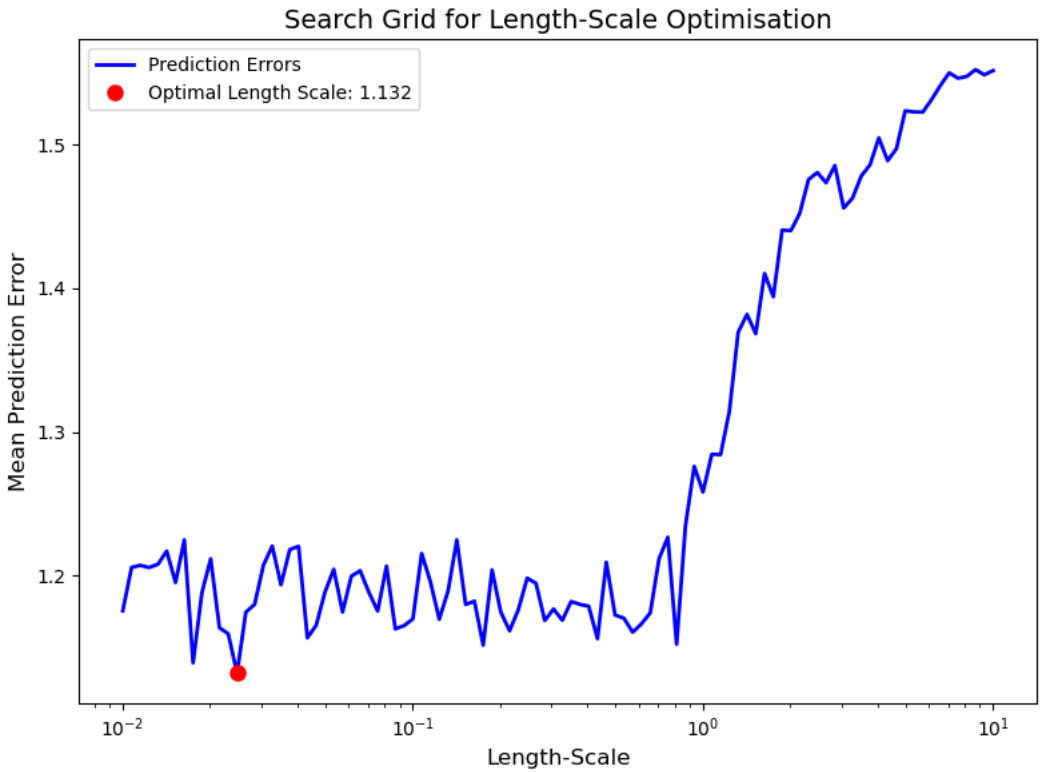


Figure 10 Search Grid for length-scale optimization

From the figure, the minimum error value occurs when l is 1.132. As l increases, the predicted count error first fluctuates and then rises. Ultimately, the optimal l value is selected for visualization, resulting in the following count.

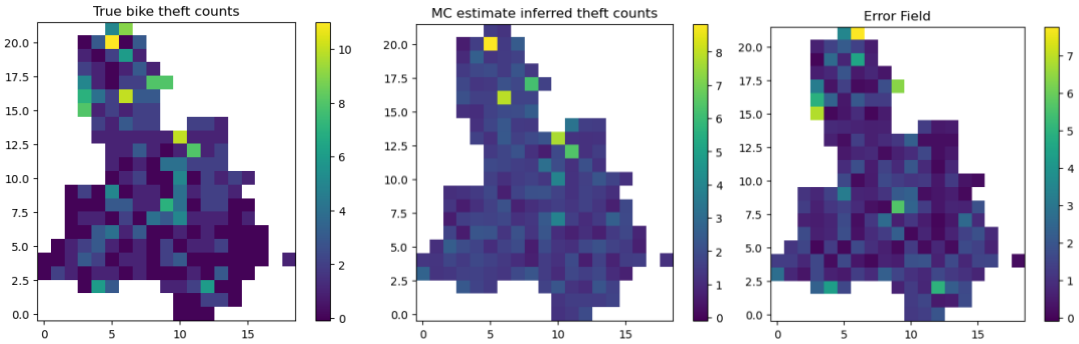


Figure 11 Predicted count result and error

It can be observed that the predictions from MC at the optimal l value perform well. For regions with significantly high counts, the predicted values are closely approximated, as evidenced by the similar colors in the heatmap. The corresponding error heatmap generally shows darker colors, indicating lower prediction errors, which demonstrates the effectiveness of the method mentioned above.