Introduction to RSTGAM

Overview

In epidemic modeling, outliers can distort parameter estimation and lead to misguided public health decisions. While many existing methods aim to achieve robustness, the ability to simultaneously detect outliers is equally important for identifying potential disease hotspots.

The **RST-GAM** package addresses this challenge by introducing a robust spatiotemporal generalized additive model that integrates a mean-shift parameter to quantify and mitigate the effects of outliers. This is combined with an appropriately designed adaptive Lasso regularization. The package leverages univariate polynomial splines and bivariate penalized splines over triangulations (BPST) to estimate functional forms, and employs a data-thinning approach to construct adaptive weights.

This vignette demonstrates how to: - Install the package. - Simulate a dataset. - Use the main functions to fit a model. - Visualize the results.

Installation

To install the package, you can use the following commands:

```
# install from GitHub
remotes::install_github("haomingsj98/RSTGAM")

# load the required packages
library(dplyr)
library(ggplot2)
library(latex2exp)
library(RSTGAM)
library(mgcv)
library(Triangulation)
library(BPST)
```

Example

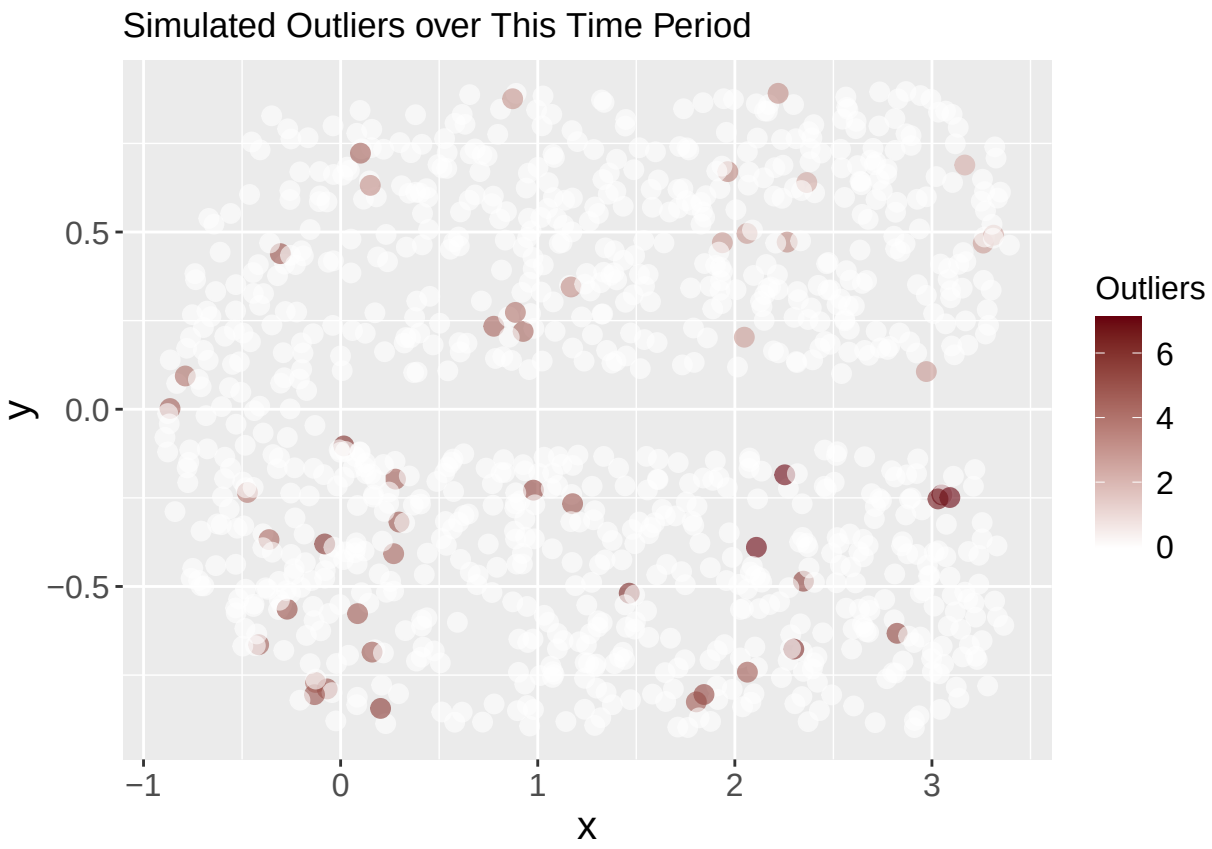
We demonstrate the usage of the package by simulating a spatiotemporal dataset with $n = 1000$ locations and 3 time points. The `Simulation_Data` function creates the example dataset, with 50 outliers generated at random locations. Below is a plot of the simulated response variables Y at time $t = 3$.

```
set.seed(123)
n = 1000 # number of location
t = 3 # number of time point
test_data = Simulation_Data(n, t, off = 50)
```

```r
# plot the simulated counts and mean parameter at each location
tibble(x = test_data$S[,1], y = test_data$S[,2],
       counts = test_data$Y[,3], offset = test_data$off) %>%
  ggplot(aes(x = x, y = y, color = counts)) +
  geom_point(alpha = 0.6, size = 3) +
  ggtitle('Simulated Counts Data at Time 3') +
  scale_colour_gradient(low = '#7fcdbb', high = '#67000d', name = 'Counts') +
  theme(axis.text=element_text(size=12), legend.text = element_text(size=12),
        axis.title = element_text(size=15), legend.title = element_text(size=12))
```

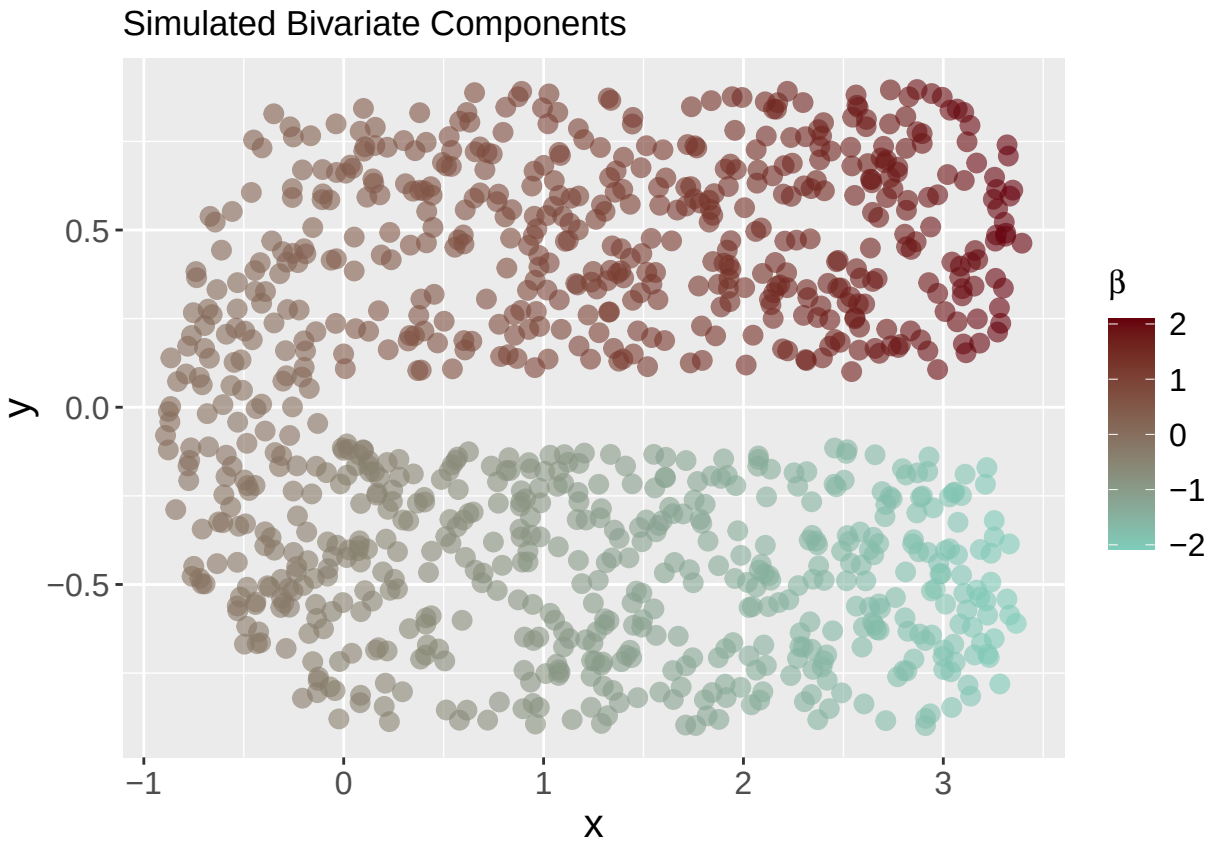
The locations of the simulated outliers are shown below. These correspond to spikes in the response values:

```r
# plot the simulated outliers
tibble(x = test_data$S[,1], y = test_data$S[,2], offset = test_data$off) %>%
  ggplot(aes(x = x, y = y, color = offset)) +
  geom_point(alpha = 0.6, size = 3) +
  ggtitle('Simulated Outliers over This Time Period') +
  scale_colour_gradient(low = 'white', high = '#67000d', name = 'Outliers') +
  theme(axis.text=element_text(size=12), legend.text = element_text(size=12),
        axis.title = element_text(size=15), legend.title = element_text(size=12))
```



And below we plot the true bivariate components, which is a smooth function over a horseshoe domain.

```
# obtain the true bivariate components
m=fs.test(test_data$S[,1],
          test_data$S[,2], b=1)

true_beta = 0.5*m

tibble(x = test_data$S[,1], y = test_data$S[,2],
       Beta = true_beta) %>%
  ggplot(aes(x = x, y = y, color = Beta)) +
  geom_point(alpha = 0.6, size = 3) +
  ggtitle('Simulated Bivariate Components') +
  scale_colour_gradient(low = '#7fcdbb', high = '#67000d', name = TeX('$\\beta$')) +
  theme(axis.text=element_text(size=12), legend.text = element_text(size=12),
        axis.title = element_text(size=15), legend.title = element_text(size=12))
```



We use the `RST_GAM` function to fit the model with the simulated dataset. Before running this function, users must define several hyperparameters:

- **Triangulation**: Includes the triangulation (`Tr`) and vertices (`V`) for the BPST method.
- **Degree and Smoothness Parameters**: The degree parameter (`d`) and smoothness parameter (`r`) for bivariate spline approximation.
- **Univariate Splines**: The number of interior knots (`N`) and the degree parameter (ρ) for univariate spline approximation.
- **Data Thinning**: The number of data thinning folds (`k`) to construct weights for adaptive Lasso.
- **Regularization Parameters**: Two lists of regularization parameters (`lambda1` and `lambda2`) for adaptive Lasso and roughness penalty.

For detailed descriptions of these parameters, please refer to our paper.

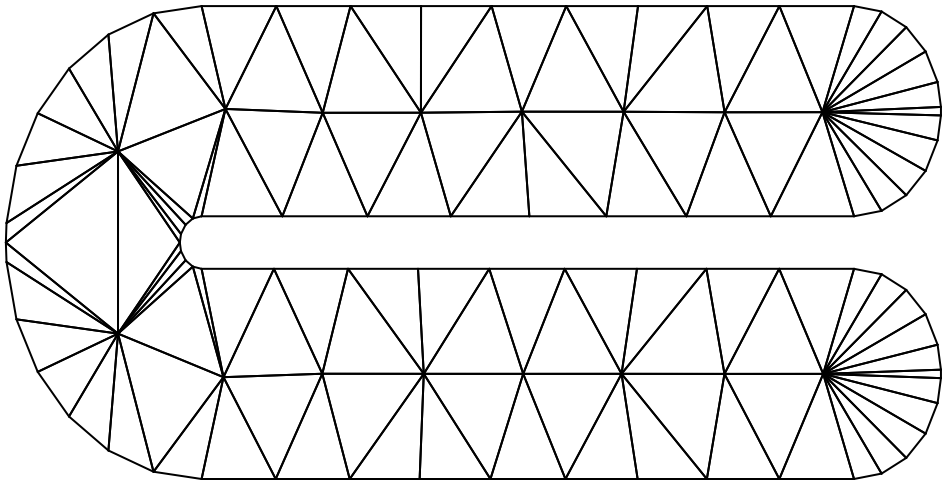
In this example, we use:

- **Cubic splines** ($\rho = 2$) with 4 interior knots ($N = 4$) for univariate spline approximation.
- **Bivariate splines** with degree parameter ($d = 2$) and smoothness parameter ($r = 1$).
- **Data thinning** with one fold ($k = 1$).

The triangulation used in this example is shown below. Users can generate similar triangulations using the `Triangulation` R package.

```
data("example_dataset")
Tr=example_dataset$Tr
V=example_dataset$V
```

```r
# plot the triangulation of the horseshoe domain
TriPlot(V, Tr)
```



Below, we run our model using the specified parameters and display the outlier detection results in a confusion table. The model successfully identifies all simulated outliers, with only one incorrect classification of a non-outlier location.

```r
# the lambda list for roughness
lambda_start=0.01
lambda_end1=50
nlambda=10
lambda1=exp(seq(log(lambda_start),log(lambda_end1),length.out=nlambda))/3000

# the lambda list for l1
lambda_end2 = 100
lambda2=exp(seq(log(lambda_start),log(lambda_end2),length.out=nlambda))/3000

# run the model
set.seed(123)
start_time = proc.time()
test_model = RST_GAM(Y = test_data$Y, X = test_data$X, S = test_data$S,
                     X_t = matrix(c(log(test_data$I)), ncol = 1), Tr = Tr, V = V,
                     lambda1 = lambda1, lambda2 = lambda2, k = 1)
```

```
## [1] "Stage 1: Construct Basis Splines"
```

```
## [1] "Stage 1 Done"
## [1] "Stage 1: Weight Construction"
## [1] "Stage 2 done"
## [1] "Stage 3: Penalization Parameter Selection"
## [1] "Stage 3 Done"
```

```
end_time = proc.time() - start_time
end_time[[3]]
```

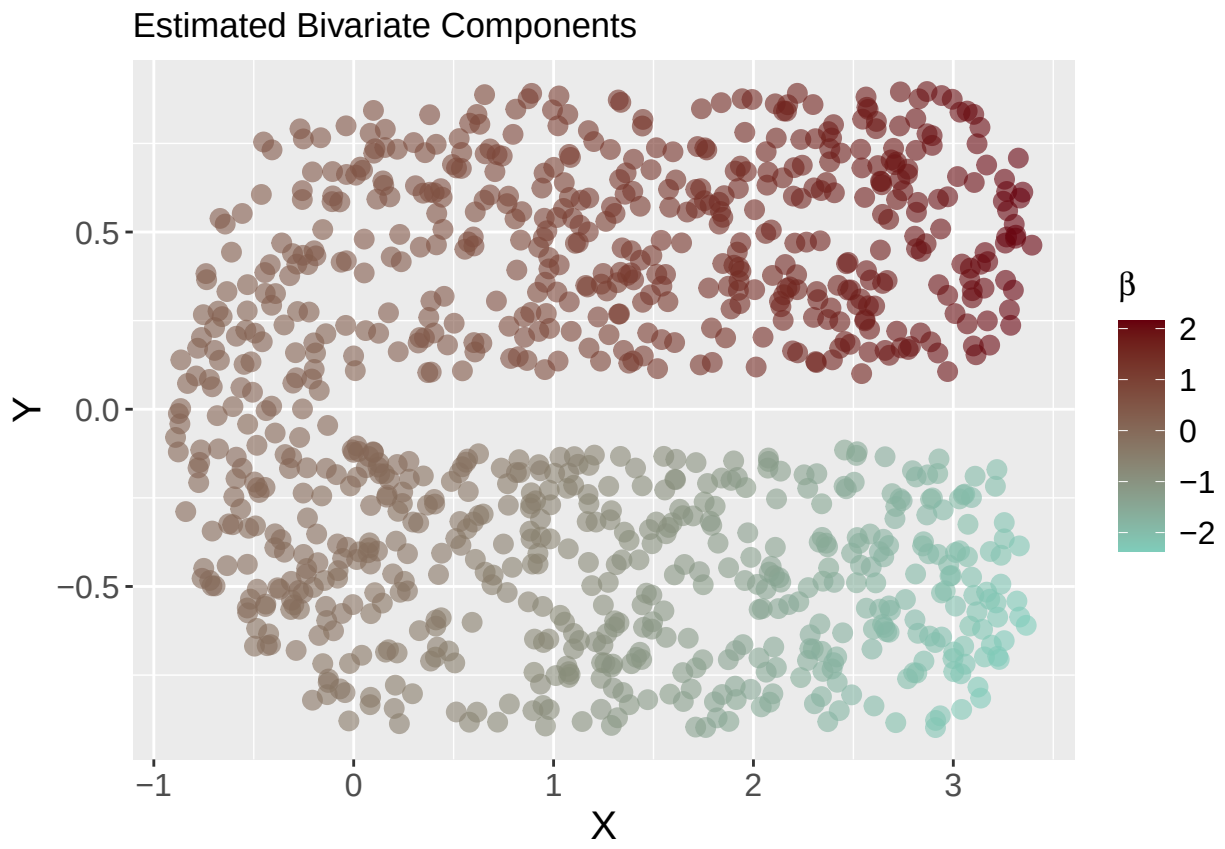
```
## [1] 287.27
```

```
# obtain the true outlier label
off_label = ifelse(test_data$off[test_model$Ind] == 0, 'No Spike', 'Spike')

# obtain the estimated outlier label
slack_label = ifelse(as.vector(test_model$xi_hat) == 0, 'No Spike', 'Spike')
table(Pred = slack_label, True = off_label)
```

```
##           True
## Pred       No Spike Spike
##   No Spike      945     0
##   Spike           1    50
```

In addition to outlier detection, our model achieves robust functional form estimations. Below we use the `RSTGAM_plot` function to summarize the outputs from our main function `RST_GAM` and prepare them for visualization. The estimated bivariate component is contained in `summary1`. The resulting plot demonstrates that the model successfully captures the smooth shape of the true bivariate function, with a mean integrated square error (MISE) of 0.03.

```
# obtain the summary results from the model output
summary_model = RSTGAM_plot(test_model, S = test_data$S)

# plot the estimated bivariate components
summary_model$summary1 %>%
  ggplot(aes(x = X, y = Y, color = Beta)) +
  geom_point(alpha = 0.6, size = 3) +
  ggtitle('Estimated Bivariate Components') +
  scale_colour_gradient(low = '#7fcdbb', high = '#67000d', name = TeX('$\\beta$')) +
  theme(axis.text=element_text(size=12), legend.text = element_text(size=12),
        axis.title = element_text(size=15), legend.title = element_text(size=12))
```



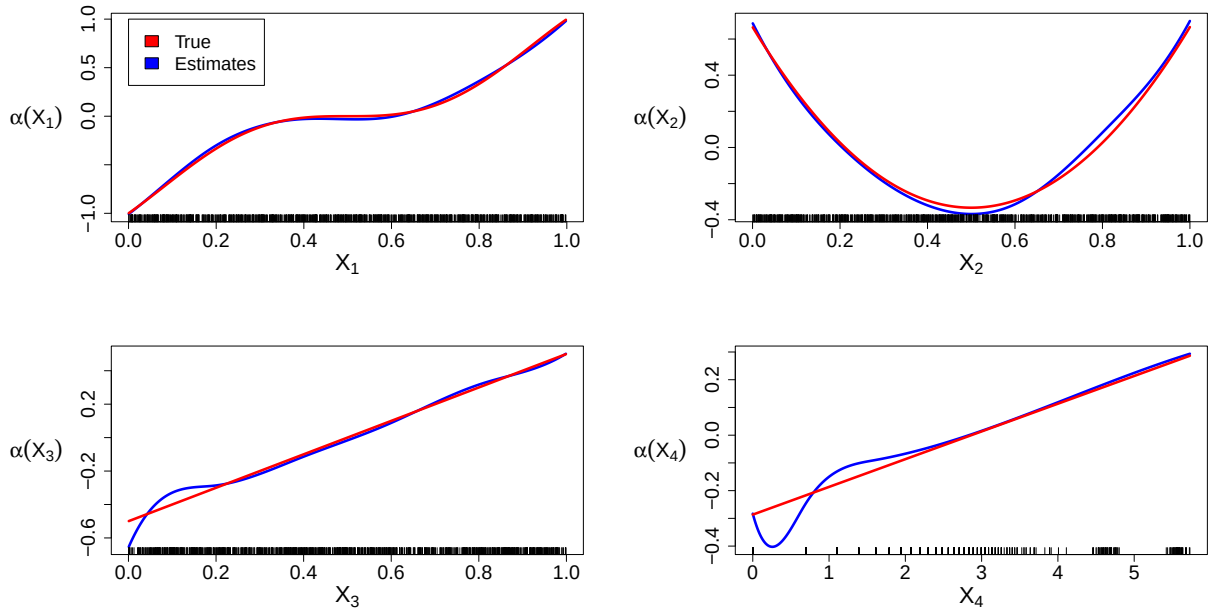

```
# MISE
mean((summary_model$summary1$Beta - true_beta[test_model$Ind])^2)
```

```
## [1] 0.02922356
```

Here, we present the results for the estimated univariate components, with information extracted from `X0` and `summary2`. The estimated functions are shown in blue, while the true function forms are depicted in red. The model demonstrates similar performance across all univariate components, accurately recovering the true forms with a low mean integrated square error, highlighting its robustness.

```
# obtain the estimated univariate components
X0 = summary_model$X0
mhat = summary_model$summary2

# the true univariate functions
# beta_1
beta.1=function(x0) 1/2*sin(2*pi*x0)-cos(pi*x0)
# beta_2
beta.2=function(x0) 4*((x0-0.5)^2-2/3*(0.5)^3)
# beta_3
beta.3=function(x0) x0-0.5
# beta_4
beta.4 = function(x0) 0.1 * x0

beta0.1=beta.1(X0[,1])
```

```r
beta0.2=beta.2(X0[,2])
beta0.3=beta.3(X0[,3])
beta0.4=beta.4(X0[,4]) - mean(beta.4(X0[,4]))

# plot
par(mar = c(5, 9, 4, 2))
par(mfrow = c(2,2))

plot(X0[,1], mhat[,1], xlab = TeX('$X_1$'), type = 'l', lwd =3, ylab = '',
     cex.axis=1.7, cex.lab=2, cex = 1.3, pch=16, col = 'blue')
lines(X0[,1], beta0.1, lwd = 3, col = 'red')
mtext(TeX('$\\alpha(X_1)$'), side = 2, line = 3.5, las = 1, cex = 1.5)
custom_ticks <- test_data$X[,1]
axis(side = 1, at = custom_ticks, labels = FALSE, tcl = 0.5)
legend(0, 1, legend = c('True', 'Estimates'), fill = c('red', 'blue'), cex = 1.6)

plot(X0[,2], mhat[,2], xlab = TeX('$X_2$'),  type = 'l', lwd =3, ylab = '',
     cex.axis=1.7, cex.lab=2, cex = 1.3, pch=16, col = 'blue')
lines(X0[,2], beta0.2, lwd = 3,  col = 'red')
mtext(TeX('$\\alpha(X_2)$'), side = 2, line = 3.5, las = 1, cex = 1.5)
custom_ticks <- test_data$X[,2]
axis(side = 1, at = custom_ticks, labels = FALSE, tcl = 0.5)

plot(X0[,3], mhat[,3], xlab = TeX('$X_3$'),  type = 'l', lwd =3, ylab = '',
     cex.axis=1.7, cex.lab=2, cex = 1.3, pch=16, col = 'blue')
lines(X0[,3], beta0.3, lwd = 3,  col = 'red')
mtext(TeX('$\\alpha(X_3)$'), side = 2, line = 3.5, las = 1, cex = 1.5)
custom_ticks <- test_data$X[,3]
axis(side = 1, at = custom_ticks, labels = FALSE, tcl = 0.5)

plot(X0[,4], mhat[,4], xlab = TeX('$X_4$'),  type = 'l', lwd =3, ylab = '',
     cex.axis=1.7, cex.lab=2, cex = 1.3, pch=16, col = 'blue')
lines(X0[,4], beta0.4, lwd = 3,  col = 'red')
mtext(TeX('$\\alpha(X_4)$'), side = 2, line = 3.5, las = 1, cex = 1.5)
custom_ticks <- log(test_data$I)
axis(side = 1, at = custom_ticks, labels = FALSE, tcl = 0.5)
```




```
# MISE
sum((beta0.1-mhat[,1])^2)*0.001
```

```
## [1] 0.0005219908
```

```
sum((beta0.2-mhat[,2])^2)*0.001
```

```
## [1] 0.0008417262
```

```
sum((beta0.3-mhat[,3])^2)*0.001
```

```
## [1] 0.0008541147
```

```
sum((beta0.4-mhat[,4])^2)*0.001
```

```
## [1] 0.001675543
```

Conclusion

The `RST_GAM` package implements our proposed robust spatiotemporal generalized additive model. Users can explore the method by modifying the simulation settings, such as using different bivariate and univariate functions or varying outlier scenarios. For a detailed application to COVID-19 data in the U.S., please refer to our paper.

Reference

Wang, L. and Lai, M.-J. (2019). Triangulation. R package version 1.0.

Yu, S., Wang, G., Wang, L., Liu, C., and Yang, L. (2020). Estimation and inference for generalized geoadditive models. Journal of the American Statistical Association, 115(530):761–774