

# Package ‘gstsm’

July 22, 2025

**Title** Generalized Spatial-Time Sequence Miner

**Version** 1.0.0

**Description** Implementations of the algorithms present article  
Generalized Spatial-Time Sequence Miner, original title  
(Castro, Antonio; Borges, Heraldo ; Pacitti, Esther ; Porto, Fabio  
; Coutinho, Rafaelli ; Ogasawara, Eduardo . Generalização de Mineração de  
Sequências Restritas no Espaço e no Tempo. In: XXXVI SBBD -  
Simpósio Brasileiro de Banco de Dados, 2021 <[doi:10.5753/sbbd.2021.17891](https://doi.org/10.5753/sbbd.2021.17891)>).

**License** MIT + file LICENSE

**Encoding** UTF-8

**RoxygenNote** 7.2.1

**Imports** digest

**Suggests** testthat (>= 3.0.0)

**Config/testthat/edition** 3

**NeedsCompilation** no

**Author** Antonio Castro [aut, cre],  
Cássio Souza [aut, ctb],  
Jorge Rodrigues [aut, ctb],  
Esther Pacitti [aut],  
Fábio Porto [aut],  
Florent Masseglia [aut],  
Rafaelli Coutinho [aut, ths],  
Eduardo Ogasawara [aut, ths] (ORCID:  
<<https://orcid.org/0000-0002-0466-0626>>),  
Federal Center for Technological Education of Rio de Janeiro [cph]

**Maintainer** Antonio Castro <[gstsm@eic.cefet-rj.br](mailto:gstsm@eic.cefet-rj.br)>

**Repository** CRAN

**Date/Publication** 2022-10-19 20:05:12 UTC

## Contents

find . . . . . 2

|                                           |           |
|-------------------------------------------|-----------|
| find.default . . . . .                    | 3         |
| find.gstsm . . . . .                      | 3         |
| find_kernel_ranged_group . . . . .        | 4         |
| generate_adjacency_matrix . . . . .       | 4         |
| generate_candidates . . . . .             | 5         |
| generate_candidates.default . . . . .     | 5         |
| generate_candidates.gstsm . . . . .       | 6         |
| gstsm . . . . .                           | 6         |
| merge . . . . .                           | 7         |
| merge.default . . . . .                   | 8         |
| merge.gstsm . . . . .                     | 8         |
| merge_kernel_ranged_groups . . . . .      | 9         |
| merge_open_kernel_ranged_groups . . . . . | 9         |
| mine . . . . .                            | 10        |
| mine.default . . . . .                    | 10        |
| mine.gstsm . . . . .                      | 11        |
| split_groups . . . . .                    | 11        |
| validate_and_close . . . . .              | 12        |
| validate_kernel_ranged_groups . . . . .   | 12        |
| <b>Index</b>                              | <b>14</b> |

---

|      |                          |
|------|--------------------------|
| find | <i>Find - definition</i> |
|------|--------------------------|

---

**Description**

S3 class definition for find method.

**Usage**

find(object, ck)

**Arguments**

- |        |                   |
|--------|-------------------|
| object | a GSTSM object    |
| ck     | set of candidates |

**Value**

Solid Ranged-Group(s) of all candidate sequences

---

|              |                       |
|--------------|-----------------------|
| find.default | <i>Find - default</i> |
|--------------|-----------------------|

---

**Description**

Default method for find. Does nothing.

**Usage**

```
## Default S3 method:  
find(object, ck)
```

**Arguments**

|        |                   |
|--------|-------------------|
| object | a GSTSM object    |
| ck     | set of candidates |

**Value**

Solid Ranged-Group(s) of all candidate sequences

---

|            |                                    |
|------------|------------------------------------|
| find.gstsm | <i>Find - GSTSM implementation</i> |
|------------|------------------------------------|

---

**Description**

GSTSM implementation for find method. Does nothing. The goal is to find the Ranged Groups information for a candidate c.

**Usage**

```
## S3 method for class 'gstsm'  
find(object, ck)
```

**Arguments**

|        |                   |
|--------|-------------------|
| object | a GSTSM object    |
| ck     | set of candidates |

**Value**

Solid Ranged-Group(s) of all candidate sequences

---

find\_kernel\_ranged\_group

*Find Kernel Ranged Group*


---

### Description

The goal is to find the Kernel Ranged Group information for a candidate *c*.

### Usage

```
find_kernel_ranged_group(c, d, gamma, beta, adjacency_matrix)
```

### Arguments

|                         |                            |
|-------------------------|----------------------------|
| <i>c</i>                | candidate                  |
| <i>d</i>                | set of transactions        |
| <i>gamma</i>            | minimum temporal frequency |
| <i>beta</i>             | minimum group size         |
| <i>adjacency_matrix</i> | adjacency matrix           |

### Value

Kernel Ranged-Group(s) of *c* updated

---

generate\_adjacency\_matrix

*Generate Adjacency Matrix*


---

### Description

Helper function that generates an adjacency matrix.

### Usage

```
generate_adjacency_matrix(spatial_positions, sigma)
```

### Arguments

|                          |                                   |
|--------------------------|-----------------------------------|
| <i>spatial_positions</i> | set of spatial positions          |
| <i>sigma</i>             | max distance between group points |

### Value

Adjacency Matrix

---

|                     |                                         |
|---------------------|-----------------------------------------|
| generate_candidates | <i>Generate Candidates - definition</i> |
|---------------------|-----------------------------------------|

---

**Description**

S3 class definition for generate\_candidates method.

**Usage**

```
generate_candidates(object, srg)
```

**Arguments**

|        |                            |
|--------|----------------------------|
| object | a GSTSM object             |
| srg    | set of Solid Ranged Groups |

**Value**

candidate sequences of size  $k + 1$

---

|                             |                                      |
|-----------------------------|--------------------------------------|
| generate_candidates.default | <i>Generate Candidates - default</i> |
|-----------------------------|--------------------------------------|

---

**Description**

Default method for generate\_candidates. Does nothing.

**Usage**

```
## Default S3 method:  
generate_candidates(object, srg)
```

**Arguments**

|        |                            |
|--------|----------------------------|
| object | a GSTSM object             |
| srg    | set of Solid Ranged Groups |

**Value**

candidate sequences of size  $k + 1$

---

```
generate_candidates.gstsm
```

*Generate Candidates - GSTSM implementation*

---

### Description

The algorithm combines SRGs that have sequences of size  $k$ , received as input, to generate candidates with sequences of size  $k + 1$ . Let  $x$  and  $y$  be SRGs, the conditions for this to occur are: that we have an intersection of candidates over the time range, intersection over the set of spatial positions ( $x.g \cap y.g$ ), and a common subsequence:  $\langle x.s_2, \dots, x.s_k \rangle = \langle y.s_1, \dots, y.s_{k-1} \rangle$ .

### Usage

```
## S3 method for class 'gstsm'
generate_candidates(object, srg)
```

### Arguments

|        |                            |
|--------|----------------------------|
| object | a GSTSM object             |
| srg    | set of Solid Ranged Groups |

### Value

candidate sequences of size  $k + 1$

---

```
gstsm
```

*GSTSM*

---

### Description

S3 class definition for GSTSM.

### Usage

```
gstsm(sts_dataset, spatial_positions, gamma, beta, sigma)
```

### Arguments

|                   |                                       |
|-------------------|---------------------------------------|
| sts_dataset       | STS dataset                           |
| spatial_positions | set of spatial positions              |
| gamma             | minimum temporal frequency            |
| beta              | minimum group size                    |
| sigma             | maximum distance between group points |

## Details

This algorithm is designed to the identification of frequent sequences in STS datasets from the concept of Solid Ranged Groups (SRG). GSTSM is based on the candidate-generating principle. The goal is to start finding SRGs for sequences of size one. Then it explores the support and the number of occurrences of SRGs for larger sequences with a limited number of scans over the database.

## Value

a GSTSM object

## Examples

```
library("gstsm")

D <- as.data.frame(matrix(c("B", "B", "A", "C", "A",
                           "C", "B", "C", "A", "B",
                           "C", "C", "A", "C", "A",
                           "B", "B", "D", "A", "B",
                           "B", "D", "D", "B", "D"
                           ), nrow = 5, ncol = 5, byrow = TRUE))

ponto <- c("p1", "p2", "p3", "p4", "p5")
x <- c(1, 2, 3, 4, 5)
y <- c(0, 0, 0, 0, 0)
z <- y
P <- data.frame(ponto=ponto, x=x, y=y, z=z, stringsAsFactors = FALSE)

gamma <- 0.8
beta <- 2
sigma <- 1

gstsm_object <- gstsm(D, P, gamma, beta, sigma)

result <- mine(gstsm_object)
```

---

merge

---

Merge - definition

---

## Description

S3 class definition for merge method.

## Usage

```
merge(object, ck)
```

**Arguments**

|        |                   |
|--------|-------------------|
| object | a GSTSM object    |
| ck     | set of candidates |

**Value**

Solid Ranged-Group(s) of all candidate sequences

---

|               |                        |
|---------------|------------------------|
| merge.default | <i>Merge - default</i> |
|---------------|------------------------|

---

**Description**

Default method for merge. Does nothing.

**Usage**

```
## Default S3 method:  
merge(object, ck)
```

**Arguments**

|        |                   |
|--------|-------------------|
| object | a GSTSM object    |
| ck     | set of candidates |

**Value**

Solid Ranged-Group(s) of all candidate sequences

---

|             |                                     |
|-------------|-------------------------------------|
| merge.gstsm | <i>Merge - GSTSM implementation</i> |
|-------------|-------------------------------------|

---

**Description**

Merge - GSTSM implementation

**Usage**

```
## S3 method for class 'gstsm'  
merge(object, ck)
```

**Arguments**

|        |                   |
|--------|-------------------|
| object | a GSTSM object    |
| ck     | set of candidates |

**Value**

Solid Ranged-Group(s) of all candidate sequences

---

|                                   |
|-----------------------------------|
| merge_kernel_ranged_groups        |
| <i>Merge Kernel Ranged Groups</i> |

---

**Description**

The goal is to merge KRGs. Let q and u be two different KRGs from the same candidate sequence. They can be merged into a group  $qu = q \cup u$  as long as they have an intersection and qu has a frequency greater than or equal to the minimum frequency defined by the user.

**Usage**

merge\_kernel\_ranged\_groups(c, gamma)

**Arguments**

|       |                            |
|-------|----------------------------|
| c     | candidate                  |
| gamma | minimum temporal frequency |

**Value**

KRG

---

|                                   |
|-----------------------------------|
| merge_open_kernel_ranged_groups   |
| <i>Merge Kernel Ranged Groups</i> |

---

**Description**

The goal of is to stretch KRGs of the same candidate sequence. Its possible if two KRGs have intersection in space and the resulting KRG keeps its frequency equal to or greater than beta.

**Usage**

merge\_open\_kernel\_ranged\_groups(c, timestamp, gamma, beta, adjacency\_matrix)

**Arguments**

|                  |                            |
|------------------|----------------------------|
| c                | candidate.                 |
| timestamp        | current timestamp          |
| gamma            | minimum temporal frequency |
| beta             | minimum group size         |
| adjacency_matrix | adjacency matrix           |

**Value**

Set of updated KRGs

---

|      |                          |
|------|--------------------------|
| mine | <i>Mine - definition</i> |
|------|--------------------------|

---

**Description**

S3 class definition for mine method.

**Usage**

mine(object)

**Arguments**

object            a GSTSM object

**Value**

all Solid Ranged Group(s) found, of all sizes

---

|              |                       |
|--------------|-----------------------|
| mine.default | <i>Mine - default</i> |
|--------------|-----------------------|

---

**Description**

Default method for mine. Does nothing.

**Usage**

```
## Default S3 method:  
mine(object)
```

**Arguments**

object            a GSTSM object

**Value**

all Solid Ranged Group(s) found, of all sizes

---

|            |                                    |
|------------|------------------------------------|
| mine.gstsm | <i>Mine - GSTSM implementation</i> |
|------------|------------------------------------|

---

**Description**

Mine - GSTSM implementation

**Usage**

```
## S3 method for class 'gstsm'  
mine(object)
```

**Arguments**

object            a GSTSM object

**Value**

all Solid Ranged Group(s) found, of all sizes

---

|              |                     |
|--------------|---------------------|
| split_groups | <i>Split Groups</i> |
|--------------|---------------------|

---

**Description**

Helper function that splits groups.

**Usage**

```
split_groups(pos, adjacency_matrix)
```

**Arguments**

pos                      sequence occurrence index  
adjacency\_matrix        possible connection between positions

**Value**

new set based on candidate c found in d.

---

|                    |                           |
|--------------------|---------------------------|
| validate_and_close | <i>Validate and Close</i> |
|--------------------|---------------------------|

---

**Description**

The function receives as input the set of RGs (RG) from a candidate and the minimum size of a group (beta). It starts defining a set of elements that will be removed from the set of RGs, if it does not have the minimum group size.

**Usage**

```
validate_and_close(c, gamma, beta)
```

**Arguments**

|       |                            |
|-------|----------------------------|
| c     | candidate                  |
| gamma | minimum temporal frequency |
| beta  | minimum group size         |

**Value**

validated Greedy-Ranged-Groups.

---

|                               |                                      |
|-------------------------------|--------------------------------------|
| validate_kernel_ranged_groups | <i>Validate Kernel Ranged Groups</i> |
|-------------------------------|--------------------------------------|

---

**Description**

Its objective is to verify that the user thresholds were observed in each RGs, checking if they can still be stretched by keeping the frequency greater than or equal to the minimum gamma and if the minimum group size beta occurs. It takes as input a set of RGs RG of a candidate sequence, the timestamp of the start of the current sliding window timestamp, the user-defined thresholds gamma and beta.

**Usage**

```
validate_kernel_ranged_groups(c, timestamp, gamma, beta)
```

**Arguments**

|           |                            |
|-----------|----------------------------|
| c         | candidate                  |
| timestamp | current timestamp          |
| gamma     | minimum temporal frequency |
| beta      | minimum group size         |

**Value**

Validated Kernel-Ranged-Groups.

# Index

`find`, [2](#)  
`find.default`, [3](#)  
`find.gstsm`, [3](#)  
`find_kernel_ranged_group`, [4](#)  
  
`generate_adjacency_matrix`, [4](#)  
`generate_candidates`, [5](#)  
`generate_candidates.default`, [5](#)  
`generate_candidates.gstsm`, [6](#)  
`gstsm`, [6](#)  
  
`merge`, [7](#)  
`merge.default`, [8](#)  
`merge.gstsm`, [8](#)  
`merge_kernel_ranged_groups`, [9](#)  
`merge_open_kernel_ranged_groups`, [9](#)  
`mine`, [10](#)  
`mine.default`, [10](#)  
`mine.gstsm`, [11](#)  
  
`split_groups`, [11](#)  
  
`validate_and_close`, [12](#)  
`validate_kernel_ranged_groups`, [12](#)