Package 'rsatscan'

May 9, 2026

Type Package

Title Tools for Running the External 'SaTScan' Software using R Classes and Methods

Version 1.0.10

Description The 'SaTScan'(TM) <https://www.satscan.org> software uses spatial and space-time scan statistics to detect and evaluate spatial and space-time clusters. With the 'rsatscan' package, you can run the external 'SaTScan' software from within R using R data formats. To successfully select appropriate parameter settings within 'rsatscan', you must first learn 'SaTScan'.

URL https://www.satscan.org

License GPL-3

Encoding UTF-8

LazyData true

VignetteBuilder knitr

Depends R (>= 3.0.2)

Imports utils, foreign

Suggests knitr, rmarkdown, sf

NeedsCompilation no

Repository CRAN

RoxygenNote 7.3.2

Author Ken Kleinman [aut],
Scott Hostovich [cre],
Amer Moosa [ctb]

Maintainer Scott Hostovich <HostovichS@imsweb.com>

Date/Publication 2026-01-26 15:00:08 UTC

Contents

`charlistopts` *Change list version of paramaters into char vector*

Description

Turns a list of options into a charvar of options

Usage

```
charlistopts(x)
```

Arguments

x	A list.

Details

The resulting charvar has values such as "name=value" where "name" was the named item of the list.

Value

A character vector

Not expected to be used directly.

NHumbersidecas *North Humberside leukemian and lymphoma example– cases*

Description

A data set from North Humberside. The variables are as follows:

Format

A data frame with 191 observations and 2 variables

Details

- locationid: Postal code ID
- numcases: The number of cases observed

Source

Distributed with SaTScan software: https://www.satscan.org

NHumbersidectl *North Humberside leukemian and lymphoma example– controls*

Description

A data set from North Humberside. The variables are as follows:

Format

A data frame with 191 observations and 2 variables

Details

- locationid: Postal code ID
- numcontrols: The number of controls observed

Source

Distributed with SaTScan software: https://www.satscan.org

NHumbersidegeo *North Humberside leukemian and lymphoma example– geography*

Description

A data set from North Humberside. The variables are as follows:

Format

A data frame with 191 observations and 3 variables

Details

- locationid: Postal code ID
- x-coordinate: x-coordinate
- y-coordinate: y-coordinate

Source

Distributed with SaTScan software: https://www.satscan.org

NMcas *New Mexico Brain Cancer example– cases*

Description

A data set from New Mexico. The variables are as follows:

Format

A data frame with 1175 observations and 5 variables

Details

- county: County name
- cases: Number of cases
- year: Year of case
- agegroup: Age group of case
- sex: Sex of case

Source

Distributed with SaTScan software: https://www.satscan.org

NMgeo *New Mexico Brain Cancer example– geography*

Description

A data set from New Mexico. The variables are as follows:

Format

A data frame with 194 observations and 3 variables

Details

- county: The US Postal Service ZIP code
- cases: The number of cases observed
- long: The date on which the cases were observed in that ZIP code

Source

Distributed with SaTScan software: https://www.satscan.org

NMpop *New Mexico Brain Cancer example– population*

Description

A data set from New Mexico. The variables are as follows:

Format

A data frame with 3456 observations and 5 variables

Details

- county: County name
- year: year
- year: population
- agegroup: Age group
- sex: Sex

Source

Distributed with SaTScan software: https://www.satscan.org

NYCfevercas *New York City Fever example– cases*

Description

A data set from New York City. The variables are as follows:

Format

A data frame with 194 observations and 3 variables

Details

- zip: The US Postal Service ZIP code
- cases: The number of cases observed
- long: The date on which the cases were observed in that ZIP code

Source

Distributed with SaTScan software: https://www.satscan.org

NYCfevergeo *New York City Fever example– geography*

Description

A data set from New York City. The variables are as follows:

Format

A data frame with 192 observations and 3 variables

Details

- zip: The US Postal Service ZIP code
- lat: Decimal latitude north
- long: Decimal longitude

Source

Distributed with SaTScan software: https://www.satscan.org

print.satscan *Methods for satscan-class objects*

Description

These functions define the default methods for satscan-class objects, which are the result objects from a call to satscan()

Usage

```
## S3 method for class 'satscan'
print(x, ...)
```

Arguments

x	is a satscan object
...	vestigial, for compatability with the default summary method

Value

x, invisibly. Side effect is to display ss$main, the SaTScan text report

read.col *Read SaTScan output files*

Description

Reads a SaTScan output .dbf file.

Usage

```
read.col(location, file)
```

Arguments

location	A directory location, including the trailing "/"
file	A file name, without the extension.

Details

This is expected to be a purely internal function. It's called by `satscan()` with the location and file name provided to that function. Since it's nothing more than `foreign::read.dbf()`, it's probably nor necessary to even have it as a function.

Value

A data frame.

read.gis *Read SaTScan output files*

Description

Reads a SaTScan output .dbf file.

Usage

```
read.gis(location, file)
```

Arguments

location	A directory location, including the trailing "/"
file	A file name, without the extension.

Details

This is expected to be a purely internal function. It's called by `satscan()` with the location and file name provided to that function. Since it's nothing more than `foreign::read.dbf()`, it's probably nor necessary to even have it as a function.

Value

A data frame.

`read.llr` *Read SaTScan output files*

Description

Reads a SaTScan output .dbf file.

Usage

```
read.llr(location, file)
```

Arguments

`location`	A directory location, including the trailing "/"
`file`	A file name, without the extension.

Details

This is expected to be a purely internal function. It's called by `satscan()` with the location and file name provided to that function. Since it's nothing more than `foreign::read.dbf()`, it's probably nor necessary to even have it as a function.

Value

A data frame.

`read.rr` *Read SaTScan output files*

Description

Reads a SaTScan output .dbf file.

Usage

```
read.rr(location, file)
```

Arguments

`location`	A directory location, including the trailing "/"
`file`	A file name, without the extension.

Details

This is expected to be a purely internal function. It's called by `satscan()` with the location and file name provided to that function. Since it's nothing more than `foreign::read.dbf()`, it's probably nor necessary to even have it as a function.

Value

A data frame.

`read.satscanmain` *Read SaTScan output files*

Description

Reads a SaTScan output .dbf file.

Usage

```
read.satscanmain(location, file)
```

Arguments

`location`	A directory location, including the trailing "/"
`file`	A file name, with any extensions.

Details

This is expected to be a purely internal function. It's called by `satscan()` with the location and file name provided to that function. Since it's nothing more than `readLines()`, it's probably nor necessary to even have it as a function.

Value

A data frame.

`read.sci` *Read SaTScan output files*

Description

Reads a SaTScan output .dbf file.

Usage

```
read.sci(location, file)
```

Arguments

`location`	A directory location, including the trailing "/"
`file`	A file name, without the extension.

Details

This is expected to be a purely internal function. It's called by `satscan()` with the location and file name provided to that function. Since it's nothing more than `foreign::read.dbf()`, it's probably nor necessary to even have it as a function.

Value

A data frame.

`rsatscan` *Tools for Running the External 'SaTScan' Software using R Classes and Methods.*

Description

The 'SaTScan'(TM) https://www.satscan.org software uses spatial and space-time scan statistics to detect and evaluate spatial and space-time clusters. With the 'rsatscan' package, you can run the external SaTScan software from within R using R data formats. To successfully select appropriate parameter settings within 'rsatscan', you must first learn 'SaTScan'.

Details

The parameter files are constructed in R using the `ss.options` function and written to the OS using the `write.ss.prm` function. SaTScan is run using the `satscan` function. The `satscan` function returns a `satscan-class` object that has a slot for every possible file that SaTScan makes, plus one for the parameter file you used to generate the output.

The package also includes `write.???` functions which will write case, control, geography, population, etc., files in the format expected by SaTScan, if you happen to have them (or make them) in R and want to write them into the OS for SaTScan to use.

There are summary and print methods for satscan-class objects. There are also plot methods in the sf package, which can be used if the sf package and sf packages are installed and SaTScan generated a shapefile.

For a new analysis, always start with the SaTScan GUI to test parameter settings, run an example analysis, etc. Once parameter settings are finalized, open the parameter file in a text editor and use as a template for defining each parameter in R syntax using the ss.options() or ss.options.extra() function. For example, per R syntax, string values and parameter names with special characters must be in quotes. See the ss.options and ss.options.extra sections for more details.

Currently the package works with SaTScan >= 9.2 and has been tested on Windows 10. Please contact the maintainer if you find success or trouble on other OSes.

Author(s)

Maintainer: Scott Hostovich <HostovichS@imsweb.com>

Authors:

- Ken Kleinman <ken.kleinman@gmail.com>

Other contributors:

- Amer Moosa <MoosaA@imsweb.com> [contributor]

See Also

Useful links:

- https://www.satscan.org

---

satscan *Run SaTScan in the OS*

Description

Calls out into the OS to run SaTScan, with the parameter file specified

Usage

```
satscan(
  prmlocation,
  prmfilename,
  sslocation = "c:/progra~2/satscan",
  ssbatchfilename = "SaTScanBatch",
  cleanup = TRUE,
  verbose = FALSE
)
```

Arguments

prmlocation	A string containing the directory location where the paramter file is located.
prmfilename	A string containg the name of the parameter file, without the extension, i.e., no ".prm".
sslocation	A string containing the directory location where satscanbatch.exe (Windows) is located. The default value is a common location in Windows 7.
ssbatchfilename	Name of the file containing the SaTScan executable. This is likely to be either SaTScanBatch or SaTScanBatch64. Omit the file extension.
cleanup	If true, deletes any SaTScan output files from the OS.
verbose	If true, will display the results in the R console as if running SaTScan in batch. This may be especially useful if you expect SaTScan to take a long time to run.

Details

The parameter file may have been made by the ss.options function or not.

Value

A satscan-class object, which is a list of 8 items, not all of which are always made, depending on SaTScan options and whether the program call was successful or not:

main A character vector containing the main text output from SaTScan. This is probably identical to the material displayed when verbose=True

col A data frame with the basic cluster information dataset SaTScan makes.

rr A data frame with the risk ratio dataset SaTScan makes.

gis A data frame with the geographic information dataset SaTScan makes.

llr A data frame with the log likelihood ratios dataset SaTScan makes.

sci A data frame with the other cluster information dataset SaTScan makes.

shapeclust A list object, of class sf, defined by the sf package. It contains the ESRI shapefile(s) SaTScan makes. This is made only if the sf package is available.

prm A character vector containing the contents of the parameter file you told SaTScan to use.

If an item is not made by SaTScan, it will be NA.

See Also

ss.options, write.ss.prm

Examples

```
## Not run:
## Please see vignette("rsatscan"); example() code doesn't make sense since
## all examples rely on calls to SaTScan in the OS.

## End(Not run)
```

`search_for_param` *find earliest SaTScan parameter set that contains specified parameter*

Description

This function sorts the SaTScan parameter sets by version number. Then, it searches forward from the current version to find which (if any) parameter sets contain the specified parameter. If any parameter set contains the specified parameter, then that parameter set's version number is returned as a string.

Not expected to be used directly.

Usage

```
search_for_param(ssenv, param)
```

Arguments

`ssenv`	the SaTScan environment to search for the specified parameter in
`param`	the parameter to search for in the SaTScan environment

Value

A string specifying the earliest parameter set that contains the specified parameter. If the parameter is not found in any parameter set, then an empty string is returned.

`ss.options` *Set or reset parameters to be used by SaTScan*

Description

Set or reset parameters used by SaTScan using the built-in rsatscan parameter list. Some parameters must be specified using the ss.options.extra() function.

Refer to the SaTScan parameter file after generating it from the SaTScan GUI. The parameter file can be viewed in a text editor.

Usage

```
ss.options(invals = NULL, reset = FALSE, version = NULL)
```

Arguments

invals	A list with entries of the form name=value, where value should be in quotes unless it is a number. Alternatively, may be a character vector whose entries are of the form "name=value". The "name" in either case should be a valid SaTScan parameter name; unrecognized names will generate a warning and will do nothing.
reset	If TRUE, will restore the default parameter values described in the "Details" section.
version	A string of the form "#.#" or "#.#.#" specifying a SaTScan parameter set. If this parameter is NULL or not specified, then parameters are reset based on the latest version of SaTScan.

Details

`ss.options()` is intended to function like `par()` or `options()`. There is a default set of parameter settings that resembles the one used by SaTScan, except that it produces all possible output files and makes them as .dbf files instead of text.

Value

If `invals == NULL`, returns the current parameter set, as altered by previous calls to `ss.options()` since the last call with `reset=TRUE`. Otherwise returns modified parameter set invisibly. The side effect, if `invals != NULL`, is to set the current values of the parameters per the value of `invals` and `reset`.

SaTScan Versions

The `version` argument defines which parameter set the script uses, not necessarily the version of SaTScan being used to execute the analyses. SaTScan is backwards compatible with older versions of parameter sets. For instance you might create a script that uses the 10.1 parameter set. That parameter set in the script will continue to work as you upgrade your SaTScan executable to newer versions. This is the same way that rsatscan worked up to version 1.0.3 where the script was locked to the 9.2 parameter set but you still could use SaTScan 9.3, 9.4, 9.7, 10.1, etc without access to the newer parameter set options introduced in those versions. As such, users with scripts created with rsatscan prior to version 1.0.4 must explicitly set the parameter set version in their scripts.

Environment Objects

The parameter sets are stored in the 'ssenv' environment object.

WARNING: Clearing your R environment will delete the 'ssenv' object and cause an error when attempting to use any SaTScan parameter sets. The 'rsatscan' library must be reloaded to restore the 'ssenv' object and allow SaTScan parameters to work correctly.

Examples

```
## Not run:
head(ss.options(),3)
ss.options(list(CaseFile="NYCfever.cas"))
```

```
head(ss.options(),3)

# reset; shows whole parameter file without invisible()
invisible(ss.options(reset=TRUE))
head(ss.options(),3)

# Explicitly specifying a parameter set
invisible(ss.options(reset=TRUE, version="9.2"))
head(ss.options(), 3)

## End(Not run)
```

`ss.options.extra` *Add lines to the current SaTScan parameter list*

Description

This function allows the user to add lines to the current list of parameters. It can be used to add parameters that are not allowed using the ss.options() function or to add comments to the parameter file.

Usage

```
ss.options.extra(invals = NULL, section = NULL, new.section = FALSE)
```

Arguments

`invals`	A character vector, which will be added to the current parameter list.
`section`	A character vector of length 1 that specifies the section of the parameter file to add the new parameters to. Sections are denoted in the 'ssenv' object by square brackets.
`new.section`	A logical variable indicating that a new section in the parameter file should be created. (default = FALSE)

Details

The SaTScan parameters and corresponding sections that must be added to the current parameter list using ss.options.extra() are listed below.

`Section="Input"`

`Casefile-SourceLinelistFieldMap=`	Comma separated list of variables in the case input data to be included in the line list output file. For each variable, includes the column number, variable

	type, and variable name, separated by colons.
Section="Multiple Datasets"	
[FileType][X]= [FileType][X]-SourceFieldMap= [FileType][X]-SourceDelimiter= [FileType][X]-SourceFirstRowHeader= [FileType][X]-SourceLinelistFieldMap=	Analogous to parameters in the Input section, repeated for every additional dataset, where [FileType] is CaseFile, ControlFile, or PopulationFile, and [X] may be between 2 and 20.
Section="Polygons"	
Polygon[X]=	The bound region for each Polygon, when using the Continuous Poisson probability model.

Value

`ss.options.extra()` returns `NULL` and adds lines to the parameter set per the values of invals and section.

Examples

```
## Not run:
# Append second data file to the Multiple Data Sets section of the parameter list
ss.options.extra(invals=list(CaseFile2="NYCfever.cas"), section="Multiple Data Sets")
print(ss.options()[67:70])

# Specify columns in the case input file to be included in the line list output file
ss.options.extra(invals=list('CaseFile-SourceLinelistFieldMap'=strwrap(
                                "0:0:\"IndividualID\",
                                 4:1:\"DescriptiveLongitude\",
                                 2:2:\"DescriptiveLatitude\",
                                 3:3:\"AGE\",
                                 1:3:\"GENDER\"")),
                section="Input")

# Can also append to the end of the parameter file by not specifying a section.
# This is useful for adding comments.
# Note that the input value can be specified as a character string instead of a list
# just like 'ss.options()'
ss.options.extra(invals=";This is the end of the parameter list.")
tail(ss.options(), 3)

## End(Not run)
```

subin *Substitute new values into the input object*

Description

Replaces existing values found in one object with new values

Usage

```
subin(x, ssparams)
```

Arguments

x	A character vector of the form "name=value"
ssparams	A character vector with arbitrary lines, currently imagined to be .ss.params

Details

For each line of x, the function: 1) finds the "name" and the "value" 2) checks to see whether the "name" exists in ssparams. If the "name" exists in .ss.params, then the existing line is replaced with that line of x If the "name" does not exist in .ss.params, then later parameter sets are check to see if the "name" exists in them. If the "name" exists in a later parameter set, this is printed as a note to the user. If the "name" is not found in any parameter set, then a warning is given.

Not expected to be used directly.

Value

The modified ssparams.

summary.satscan *Methods for satscan-class objects*

Description

These functions define the default methods for satscan-class objects, which are the result objects from a call to satscan()

Usage

```
## S3 method for class 'satscan'
summary(object, ...)
```

Arguments

object	is a satscan object
...	vestigial, for compatability with the default summary method

Value

object, invisibly. Side effect is to display minimal facts contained in ss

---

`write.adj` *Write a SaTScan adj file*

Description

Write a SaTScan adj file

Usage

```
write.adj(x, location, filename, userownames = FALSE)
```

Arguments

`x`	Your data frame.
`location`	Directory location where the file should be written
`filename`	Name for the output file in the OS; .adj will be added.
`userownames`	If TRUE, will write the row names into the file.

Details

Writes the input data frame to the OS, using the .adj extension. Contents of the data frame should be only what you want SaTScan to see. This is a simple function that calls write.table, since SaTScan just needs ASCII files.

`write.cas` *Write a SaTScan cas (case) file*

Description

Write a SaTScan cas (case) file

Usage

```
write.cas(x, location, filename, userownames = FALSE)
```

Arguments

`x`	Your data frame.
`location`	Directory location where the file should be written
`filename`	Name for the output file in the OS; .cas will be added.
`userownames`	If TRUE, will write the row names into the file.

Details

Writes the input data frame to the OS, using the .cas extension. Contents of the data frame should be only what you want SaTScan to see. This is a simple function that calls write.table, since SaTScan just needs ASCII files.

`write.ctl` *Write a SaTScan ctl (control) file*

Description

Write a SaTScan ctl (control) file

Usage

```
write.ctl(x, location, filename, userownames = FALSE)
```

Arguments

`x`	Your data frame.
`location`	Directory location where the file should be written
`filename`	Name for the output file in the OS; .ctl will be added.
`userownames`	If TRUE, will write the row names into the file.

Details

Writes the input data frame to the OS, using the .ctl extension. Contents of the data frame should be only what you want SaTScan to see. This is a simple function that calls write.table, since SaTScan just needs ASCII files.

`write.geo` *Write a SaTScan geo file*

Description

Write a SaTScan geo file

Usage

```
write.geo(x, location, filename, userownames = FALSE)
```

Arguments

`x`	Your data frame.
`location`	Directory location where the file should be written
`filename`	Name for the output file in the OS; .geo extension will be added.
`userownames`	If TRUE, will write the row names into the file.

Details

Writes the input data frame to a file in the OS, using the .geo extension. Contents of the data frame should be only what you want SaTScan to see. This is a simple function that calls write.table, since SaTScan just needs ASCII files.

`write.grd` *Write a SaTScan grd (grid) file*

Description

Write a SaTScan grd (grid) file

Usage

```
write.grd(x, location, filename, userownames = FALSE)
```

Arguments

`x`	Your data frame.
`location`	Directory location where the file should be written
`filename`	Name for the output file in the OS; .grd will be added.
`userownames`	If TRUE, will write the row names into the file.

Details

Writes the input data frame to the OS, using the .grd extension. Contents of the data frame should be only what you want SaTScan to see. This is a simple function that calls write.table, since SaTScan just needs ASCII files.

`write.ha` *Write a SaTScan ha (alternative hypothesis) file*

Description

Write a SaTScan ha (alternatove hypothesis) file

Usage

```
write.ha(x, location, filename, userownames = FALSE)
```

Arguments

`x`	Your data frame.
`location`	Directory location where the file should be written
`filename`	Name for the output file in the OS; .ha will be added.
`userownames`	If TRUE, will write the row names into the file.

Details

Writes the input data frame to the OS, using the .ha extension. Contents of the data frame should be only what you want SaTScan to see. This is a simple function that calls write.table, since SaTScan just needs ASCII files.

`write.max` *Write a SaTScan max file*

Description

Write a SaTScan max file

Usage

```
write.max(x, location, filename, userownames = FALSE)
```

Arguments

`x`	Your data frame.
`location`	Directory location where the file should be written
`filename`	Name for the output file in the OS; .max will be added.
`userownames`	If TRUE, will write the row names into the file.

Details

Writes the input data frame to the OS, using the .max extension. Contents of the data frame should be only what you want SaTScan to see. This is a simple function that calls write.table, since SaTScan just needs ASCII files.

`write.met` *Write a SaTScan met file*

Description

Write a SaTScan met file

Usage

```
write.met(x, location, filename, userownames = FALSE)
```

Arguments

`x`	Your data frame.
`location`	Directory location where the file should be written
`filename`	Name for the output file in the OS; .met will be added.
`userownames`	If TRUE, will write the row names into the file.

Details

Writes the input data frame to the OS, using the .met extension. Contents of the data frame should be only what you want SaTScan to see. This is a simple function that calls write.table, since SaTScan just needs ASCII files.

`write.nbr` *Write a SaTScan nbr (neighbor) file*

Description

Write a SaTScan nbr (neighbor) file

Usage

```
write.nbr(x, location, filename, userownames = FALSE)
```

Arguments

`x`	Your data frame.
`location`	Directory location where the file should be written
`filename`	Name for the output file in the OS; .nbr will be added.
`userownames`	If TRUE, will write the row names into the file.

Details

Writes the input data frame to the OS, using the .nbr extension. Contents of the data frame should be only what you want SaTScan to see. This is a simple function that calls write.table, since SaTScan just needs ASCII files.

`write.ntk` *Write a SaTScan ntk (network) file*

Description

Write a SaTScan ntk (network) file

Usage

```
write.ntk(x, location, filename, userownames = FALSE)
```

Arguments

`x`	Your data frame.
`location`	Directory location where the file should be written
`filename`	Name for the output file in the OS; .ntk will be added.
`userownames`	If TRUE, will write the row names into the file.

Details

Writes the input data frame to the OS, using the .ntk extension. Contents of the data frame should be only what you want SaTScan to see. This is a simple function that calls write.table, since SaTScan just needs ASCII files.

`write.pop` *Write a SaTScan pop (population) file*

Description

Write a SaTScan pop (population) file

Usage

```
write.pop(x, location, filename, userownames = FALSE)
```

Arguments

`x`	Your data frame.
`location`	Directory location where the file should be written
`filename`	Name for the output file in the OS; .pop will be added.
`userownames`	If TRUE, will write the row names into the file.

Details

Writes the input data frame to the OS, using the .pop extension. Contents of the data frame should be only what you want SaTScan to see. This is a simple function that calls write.table, since SaTScan just needs ASCII files.

`write.ss.prm` *Write the SaTScan parameter file*

Description

Writes the current set of SaTScan parameters to a specified location in the OS.

Usage

```
write.ss.prm(location, filename, matchout = TRUE)
```

Arguments

location	A directory location, excluding the trailing "/".
filename	The name of the file to be written to the OS; The extension ".prm" will be appended.
matchout	If false, the ResultsFile parameter will not be touched; note that this will likely result in undesirable performance from calls to satcan() using the parameter file. If true, the ResultsFile is reset to share the filename given here.

Details

The current SaTScan options can be reset or modified ss.options() and/or ss.options.extra(). Once they are set as desired, they can be written to the OS using this function.

Value

Nothing. (Invisibly.) Side effect is to write a file in the OS.

See Also

ss.options, ss.options.extra

Examples

```
## Not run:
## Would write the current ss.options() to c:/temp/NYCfever.prm
write.ss.prm("c:/tmp","NYCfever")

## End(Not run)
```

Index