Package: SticsOnR (via r-universe)

June 2, 2026

Type Package

Title Manage STICS Simulations Running the Executable or JavaStics,

Version 1.3.0

Date 2025-03-20

Description Running simulations with the 'STICS' crop model. Different ways of performing simulations are available and based on three use-cases: 1/ Calling 'JavaStics' command line to run simulations on given situations. 'JavaStics' handles text files generation from 'XML' files; 2/ Calling the model 'fortran' executable directly. Text files must be produced beforehand; and 3/ Calling the model 'fortran' executable through a wrapper. It can handle distributed runs over the machine CPUs and returns simulated output data. The wrapper also provides a concise manner for running simulations using custom parameter values and allows to chain simulations for successive situations.

License LGPL (>= 3)

URL https://github.com/SticsRPacks/SticsOnR, https://doi.org/10.5281/zenodo.4443130

BugReports https://github.com/SticsRPacks/SticsOnR/issues

Depends R (>= 3.6.0)

Imports cli, crayon (>= 1.3.4), doParallel, dplyr, foreach, magrittr, parallel, rstudioapi, stats, tibble, tidyr, SticsRFiles (>= 1.1.3), semver

Suggests covr, knitr, rmarkdown, spelling, testthat

VignetteBuilder knitr

ByteCompile true

Encoding UTF-8

Language en-US

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.3

Config/pak/sysreqs libicu-dev libxml2-dev libxslt-dev libssl-dev

Repository https://pecanproject.r-universe.dev

Date/Publication 2026-04-03 12:23:06 UTC

RemoteUrl https://github.com/SticsRPacks/SticsOnR

RemoteRef HEAD

RemoteSha bfc45b0c0823244422ed29b3da38f0dc9cf15d8f

Contents

`get_version_number` *Get STICS executable version number, or string, or hash*

Description

Get STICS executable version number, or string, or hash

Usage

```
get_version_number(exe_path, numeric = TRUE)
```

Arguments

`exe_path`	executable file path
`numeric`	Logical, `TRUE` (default) to return a numeric version as a semver class vector, or `FALSE` to return a character string with the version

Value

a smvr class vector if `numeric = TRUE` (default), or a character string if `numeric = FALSE`, with an attribute "date" if the version date is provided in the system command output. Or a hash string if the system command output contains a hash.

Examples

```
## Not run:
get_version_number("path/to/stics_exe")
get_version_number("path/to/stics_exe", numeric = FALSE)

## End(Not run)
```

`run_javastics` *Running one or several usms from a javastics workspace*

Description

This function uses basically Stics through his JavaStics command line interface

Usage

```
run_javastics(
  javastics,
  workspace = NULL,
  usm = NULL,
  keep_history = TRUE,
  optim = FALSE,
  verbose = TRUE,
  stics_exe = "modulostics",
  java_cmd = "java"
)
```

Arguments

`javastics`	Path of JavaStics
`workspace`	Path of a JavaStics workspace
`usm`	Vector of USM names. Optional, if provided, the function runs only the given USMs. If not provided, the function runs all the USMs included in workspace.
`keep_history`	Logical value (optional) to keep a copy of history file use `TRUE` (default), `FALSE` otherwise
`optim`	Logical value (optional), `TRUE` to force code_optim value to 1, `FALSE` otherwise (default)
`verbose`	Logical value for displaying information while running
`stics_exe`	The name, executable or path of the stics executable to use (optional, default to "modulostics", see details)
`java_cmd`	The java virtual machine command name or executable path

Details

stics_exe may be :

1. a model name pointing to a stics executable as done in JavaStics, e.g. "modulostics" for stics_modulo.exe, the standard version of the model shipping with JavaStics;
2. a stics executable file available from the bin folder in JavaStics, e.g. "stics_modulo.exe";
3. a path to a stics executable file, eg. "C:/Users/username/Desktop/stics.exe". NB: this file cannot be named stics_modulo.exe because it is the name of the standard STICS shipping with JavaStics (overwriting is not allowed).

Value

A list in which each element contains: usm "name", "error" status (logical) and an output "message" (JavaStics commandline execution output)

Examples

```
## Not run:
run_javastics("/path/to/JavaSTICS/folder", "example")
run_javastics("/path/to/JavaSTICS/folder", "/path/to/workspace")
run_javastics("/path/to/JavaSTICS/folder", "example", c("wheat", "pea"))
run_javastics("/path/to/JavaSTICS/folder", usm = c("wheat", "pea"))
run_javastics("/path/to/JavaSTICS/folder",
  usm = c("wheat", "pea"), optim = TRUE
)

## End(Not run)
```

---

run_stics *Running usm(s) from a/several directory(ies) or a/several subdirectory(ies) named with usm name*

Description

This function uses Stics directly through a system call

Usage

```
run_stics(stics_exe, workspace, usm = NULL, check = TRUE, verbose = FALSE)
```

Arguments

stics_exe	Path of Stics the executable file
workspace	Path of the workspace containing the Stics (txt) input files
usm	Vector of USM names. Optional, if provided, the function runs only the given USMs. If not provided, the function runs all the USMs included in workspace.

check	Logical, T for checking the model executable, F otherwise
verbose	Logical value (optional), TRUE to display usms names, FALSE otherwise (default)

Value

A list with usm names and execution error status

Examples

```
## Not run:

# Specifying individual usm directories
run_stics("/home/username/bin/Stics", "/home/username/Work/SticsInputsDir")
run_stics("/home/username/bin/Stics", c(
  "/home/username/Work/SticsInputsDir1",
  "/home/username/Work/SticsInputsDir2"
))

# Specifying a parent directory of usms directories
# running one or several usms
run_stics(
  "/home/username/bin/Stics",
  "/home/username/Work/SticsInputsRootDir", "wheat"
)
run_stics(
  "/home/username/bin/Stics", "/home/username/Work/SticsInputsRootDir",
  c("wheat", "maize")
)
# running all usms
run_stics(
  "/home/username/bin/Stics", "/home/username/Work/SticsInputsRootDir",
)

## End(Not run)
```

---

stics_wrapper *Running usm(s) from txt input files stored in one directory per* situation*, simulated results are returned in a list*

Description

This function uses Stics directly through a system call, can force Stics input parameters with values given in arguments.

Usage

```
stics_wrapper(
  model_options,
  param_values = NULL,
  situation = NULL,
  var = NULL,
  dates = NULL,
  sit_var_dates_mask = NULL
)
```

Arguments

model_options	List containing any information needed by the model. In the case of Stics: javastics (or/and stics_exe) and workspace the path of the directory containing the Stics input data for each USM (one folder per USM where Stics input files are stored in txt format). See stics_wrapper_options() for more information.
param_values	(optional) a named vector or a tibble that contains values of Stics input parameters to use in the simulations. Should have one named column per parameter. An optional column named situation containing the name of the situations (USMs for Stics) allows to define different values of the parameters for different situations. If param_values is not provided, the simulations will be performed using the parameters values defined in the Stics input files referenced in model_options argument. WARNING: up to now, for intercrop situations, plant parameter(s) defined in param_values is(are) only associated to the main crop.
situation	(optional) vector of situations (USMs) names for which results must be returned. Results for all simulated situations are returned if not provided.
var	(optional) vector of variables names for which results must be returned. If not provided, it returns the results for all simulated variables that were already listed in the var.mod (i.e. from the last simulation).
dates	(optional) vector of dates (POSIXct) for which results must be returned. Results for all dates simulated are returned if not provided. If required dates varies between situations, use sit_var_dates_mask argument
sit_var_dates_mask	(optional) List of situations: a named list containing a mask for variables and dates for which simulated values should be returned. Typically a list containing the observations to which simulations should be compared as provided by SticsRFiles::get_obs

Value

A list containing simulated values (sim_list: a list of tibbles (one element per situation) and an error code (error) indicating if at least one simulation ended with an error.

See Also

stics_wrapper_options() for more information on how to provide model_options.

Examples

```
## Not run:

# Specifying the JavaStics folder
javastics <- "/path/to/JavaSTICS/folder"

# Setting the input data folder path, root directory of the usms directories
data_path <- "/path/to/usms/subdirs/root"

# Setting the mandatory simulations options
sim_options <- stics_wrapper_options(
  javastics = javastics,
  workspace = data_path
)

# Running all the usms that have a corresponding input folder in data_path
results <- stics_wrapper(sim_options)

# Running a sublist of usm
usms_list <- c("wheat", "pea", "maize")
results <- stics_wrapper(sim_options, situation = usms_list)

# Applying a single parameter values vector for the sublist of usms
param_values <- c(0.002, 50)
names(param_values) <- c("dlaimax", "durvieF")
results <- stics_wrapper(
  model_options = sim_options,
  situation = usms_list, param_values = param_values
)

# Applying different values of the parameters for the usms
# Let's run usm wheat with c(dlaimax=0.001, durvieF=50),
# usm pea with c(dlaimax=0.001, durvieF=60),
# and usm maize with c(dlaimax=0.001, durvieF=70)
param_values <- data.frame(
  Situation = c("wheat", "pea", "maize"),
  dlaimax = c(0.001, 0.001, 0.001),
  durvieF = c(50, 60, 70)
)
results <- stics_wrapper(
  model_options = sim_options,
  param_values = param_values, situation = c("wheat", "pea", "maize")
)

## End(Not run)
```

`stics_wrapper_options` *Getting/setting a stics_wrapper options list with initialized fields*

Description

This function returns a default options list if called with no arguments, or a pre-formated model option list with checks on the inputs.

Usage

```
stics_wrapper_options(
  javastics = NULL,
  stics_exe = "modulostics",
  workspace = NULL,
  parallel = FALSE,
  cores = NA,
  time_display = FALSE,
  verbose = TRUE,
  force = FALSE,
  successive = NULL,
  ...
)
```

Arguments

javastics	Path of JavaStics. Optional, needed if stics_exe is not provided, or if stics_exe relates to an exe in the javastics path (see details)
stics_exe	The name, executable or path of the stics executable to use (optional, default to "modulostics", see details)
workspace	Path of the workspace containing the Stics (txt) input files or path of a single directory containing one sub-folder per USM (named as the USM names) with Stics (txt) input files in them.
parallel	Boolean. Is the computation to be done in parallel ?
cores	Number of cores to use for parallel computation.
time_display	Display time
verbose	Logical value (optional), TRUE to display informations during execution, FALSE otherwise (default)
force	Boolean. Don't check javastics, stics_exe and workspace (default to FALSE, see details)
successive	List of vectors containing the names of the UMSs to consider as successive (e.g. list(c("usm1.1","usm1.2"),c("usm2.1","usm2.2")) defines 2 successions usm1.1->usm1.2 and usm2.1->usm2.2)
...	Add further arguments set the options list values

Details

stics_exe may be :

1. a model name pointing to a stics executable as done in JavaStics, e.g. "modulostics" for stics_modulo.exe, the standard version of the model shipping with JavaStics;

2. a stics executable file available from the bin folder in JavaStics, e.g. "stics_modulo.exe";
3. a path to a stics executable file, eg. "C:/Users/username/Desktop/stics.exe".

`javastics` must be provided for case (1) and (2).

If `force=TRUE`, checks are not done for `javastics`, `stics_exe` and `workspace`. In this case, they are returned as is, and will be checked (and potentially updated to match the right stics executable) only at execution of `stics_wrapper()`. This option is used for portability, when e.g. `stics_wrapper_options()` outputs are sent to a remote.

Value

A list containing Stics model stics_wrapper options

Examples

```
## Not run:
# Getting simulations options and defaults values for the stics_wrapper
# function

stics_wrapper_options()

#> $javastics
#> [1] "unknown"
#>
#> $stics_exe
#> [1] "modulostics"
#>
#> $workspace
#> [1] "unknown"
#>
#> $parallel
#> [1] FALSE
#>
#> $cores
#> [1] NA
#>
#> $time_display
#> [1] FALSE
#>
#> $verbose
#> [1] TRUE
#>
#> $force
#> [1] FALSE

# Setting mandatory simulations options
javastics <- "path/to/javastics"
data_path <- "path/to/data_directory"
sim_options <- stics_wrapper_options(
  javastics = javastics,
  workspace = data_path
)
```

```
# Changing default values (e.g. parallel):
sim_options <- stics_wrapper_options(
  javastics = javastics,
  workspace = data_path,
  parallel = TRUE
)

#> $javastics
#> [1] "path/to/JavaSTICS-v85"
#>
#> $stics_exe
#> [1] "path/to/JavaSTICS-v85/bin/stics_modulo.exe"
#>
#> $workspace
#> [1] "path/to/data"
#>
#> $parallel
#> [1] TRUE
#>
#> $cores
#> [1] NA
#>
#> $time_display
#> [1] FALSE
#>
#> $verbose
#> [1] TRUE
#>
#> $force
#> [1] FALSE

# Using the `force` argument to keep the inputs as is:
sim_options <- stics_wrapper_options(
  javastics = javastics,
  workspace = data_path,
  force = TRUE
)

#> $javastics
#> [1] "path/to/JavaSTICS-v85"
#>
#> $stics_exe
#> [1] "modulostics"
#>
#> $workspace
#> [1] "path/to/data"
#>
#> $parallel
#> [1] FALSE
#>
#> $cores
#> [1] NA
```

```
#>
#> $time_display
#> [1] FALSE
#>
#> $verbose
#> [1] TRUE
#>
#> $force
#> [1] TRUE

# This will be checked and modified by a `do.call()` in `stics_wrapper()`:
do.call(stics_wrapper_options, model_options)

#> $javastics
#> [1] "path/to/JavaSTICS-v85"
#>
#> $stics_exe
#> [1] "path/to/JavaSTICS-v85/bin/stics_modulo.exe"
#>
#> $workspace
#> [1] "path/to/data"
#>
#> $parallel
#> [1] FALSE
#>
#> $cores
#> [1] NA
#>
#> $time_display
#> [1] FALSE
#>
#> $verbose
#> [1] TRUE
#>
#> $force
#> [1] FALSE

# Note the `stics_exe` path that was modified and checked to the path were
# it was found.

## End(Not run)
```

Index