Package ‘SticsRFiles’

July 17, 2024

Title  Read and Modify 'STICS' Input/Output Files
Version  1.4.0
Date  2024-07-16
Description  Manipulating input and output files of the 'STICS' crop model. Files are either 'JavaSTICS' XML files or text files used by the model 'fortran' executable. Most basic functionalities are reading or writing parameter names and values in both XML or text input files, and getting data from output files. Advanced functionalities include XML files generation from XML templates and/or spreadsheets, or text files generation from XML files by using 'xslt' transformation.
License  LGPL (>= 3)
URL  https://github.com/SticsRPacks/SticsRFiles,
     https://doi.org/10.5281/zenodo.4443206
BugReports  https://github.com/SticsRPacks/SticsRFiles/issues
Depends  R (>= 4.0.0)
Imports  cli, crayon, curl, data.table, dplyr (>= 1.0.0), lifecycle, lubridate, methods, rlang, rstudioapi, stringr, tibble, tidy, tidyselect, tools, utils, XML, xml2, xslt
Suggests  covr, formatR, knitr, learnr, readxl, rmarkdown, spelling, testthat
VignetteBuilder  knitr
ByteCompile  true
Encoding  UTF-8
Language  en-US
RoxygenNote  7.3.1
Collate  'add_node_to_doc.R' 'add_stics_nodes.R' 'all_in_par.R'
     'all_out_var.R' 'attributes_list2matrix.R'
     'check_choice_param.R' 'check_java_path.R'
     'check_java_workspace.R' 'check_output_files.R'
     'check_param_names.R' 'col_names_to_var.R'
'compute_date_from_day.R' 'compute_day_number.R'
'convert_xml2txt.R' 'convert_xml2txt_int.R' 'download_data.R'
'download_usm_xml.R' 'exist_param_xml.R'
'exists_javistics_pref.R' 'exists_param.R'
'expand_stics_names.R' 'extract.R' 'global.R' 'file_document.R'
'find_names.R' 'force_param_values.R' 'gen_climate.R'
'gen_inh_doc.R' 'gen_inh_xml.R' 'gen_new_travail.R' 'gen_obs.R'
'gen_paramstl.R' 'gen_solv_xsl_file.R' 'gen_sols_xml.R'
'gen_sta_doc.R' 'gen_std_xml.R' 'gen_tec_doc.R' 'gen_tec_xml.R'
'gen_usms_sols_doc.R' 'gen_usms_xml.R' 'gen_usms_xml2txt.R'
'gen_varmod.R' 'get_climate_txt.R' 'get_cultivars_list.R'
'get_cultivars_param.R' 'get_file.R' 'get_file_int.R'
'get_formalisms_xml.R' 'get_java_workspace.R'
'get_lai_forcing.R' 'get_name_value_file_value.R' 'get_obs.R'
'get_option_choice_param_values.R' 'get_options_choices.R'
'get_options_names.R' 'get_param_bounds.R'
'get_param_bounds_xml.R' 'get_param_desc.R'
'get_param_formalisms.R' 'get_param_info_xml.R'
'get_param_names.R' 'get_param_names_xml.R'
'get_param_number.R' 'get_param_txt.R' 'get_param_type.R'
'get_param_value.R' 'get_param_xml.R' 'get_params_dct.R'
'get_params_from_doc.R' 'get_params_from_doc_attr.R'
'get_params_from_doc_node.R' 'get_params_from_table.R'
'get_plant_name.R' 'get_plants_nb.R' 'get_report_results.R'
'get_sim.R' 'get_soils_list.R' 'get_stics_versions_compat.R'
'get_used_param.R' 'get_usms_files.R' 'get_usms_list.R'
'get_values_by_param.R' 'get_varmod.R' 'get_xml_base_doc.R'
'get_xml_base_node.R' 'get_xml_doc_example.R'
'get_xml_files_param_df.R' 'get_xml_stics_version.R'
'init_javistics_pref.R' 'is_os_name.R' 'is_stics_doc.R'
'is_stics_xml.R' 'javistics_cmd_util.R' 'javistics_path.R'
'manage_stics_versions.R' 'merge_nodesets.R'
'read_params_table.R' 'remove_node_from_doc.R'
'remove_parent_from_doc.R' 'replace_string_in_file.R'
'replace_txt_param_value.R' 'set_codeoptim.R'
'replace_xml_nodes.R' 'set_java_workspace.R'
'set_param_txt.R' 'set_param_value.R' 'set_param_xml.R'
'set_sols_param_xml.R' 'set_usms_param_xml.R' 'static_help.R'
'stics_environment.R' 'stics_examples_utils.R'
'upgrade_ini_xml.R' 'upgrade_param_gen_xml.R'
'upgrade_param_newform_xml.R' 'upgrade_plt_xml.R'
'upgrade_sols_xml.R' 'upgrade_sta_xml.R' 'upgrade_tec_xml.R'
'upgrade_usms_xml.R' 'upgrade_workspace_xml.R'
'var_to_col_names.R' 'xml_document.R' 'xml_files_functions.R'
'zzz.R'

NeedsCompilation no

Author Patrice Lecharpentier [aut, cre]

(<https://orcid.org/0000-0002-4044-4322>),
Remi Vezy [aut] (<https://orcid.org/0000-0002-0808-1461>),
Samuel Buis [aut] (<https://orcid.org/0000-0002-8676-5447>),
Michel Giner [aut] (<https://orcid.org/0000-0002-9310-2377>),
Timothee Flutre [ctb],
Thomas Robine [ctb],
Amine Barkaoui [ctb],
Patrick Chabrier [ctb],
Julie Constantin [rev],
Dominique Ripoche [rev],
Marie Launay [rev],
Alain Mollier [rev],
Christine Le Bas [rev],
Joel Leonard [rev]

Maintainer  Patrice Lecharpentier <patrice.lecharpentier@inrae.fr>

Repository  CRAN

Date/Publication  2024-07-17 12:20:02 UTC

Contents

compute_date_from_day ........................................ 4
compute_day_from_date ........................................ 5
convert_xml2txt .................................................. 6
download_data ...................................................... 7
download_usm_csv .................................................. 8
download_usm_xl .................................................... 9
force_param_values ................................................. 10
gen_ini_xml .......................................................... 11
gen_obs ............................................................... 12
gen_sols_xml ........................................................ 14
gen_sta_xml .......................................................... 15
gen_tec_xml .......................................................... 17
gen_usms_xml ......................................................... 18
gen_usms_xml2txt ................................................... 20
gen_varmod ............................................................. 21
get_climate_txt ...................................................... 23
get_cultivars_list .................................................. 24
get_cultivars_param ............................................... 24
get_examples_path .................................................. 25
get_lai_forcing ..................................................... 26
get_obs ............................................................... 26
get_param_info ....................................................... 28
get_param_txt ........................................................ 29
get_param_xml ........................................................ 33
get_plants_nb ......................................................... 35
get_report_results ................................................... 36
get_sim ............................................................... 37
gen_soils_list ....................................................... 38
compute_date_from_day

Convert day number into date

Description

Computes the date corresponding to a given day number (or vector of) with reference to a start year. Typically, the start year should be the year of a STICS simulation start, but not necessarily.

Usage

compute_date_from_day(day, start_year)

Arguments

day
day number(s) to be converted

start_year
year to be used as time reference (simulation start year).

Value

Date vector
compute_day_from_date

Author(s)
Timothee Flutre

Examples
compute_date_from_day(day = 290, start_year = 1994)
compute_date_from_day(day = 700, start_year = 1994)

compute_day_from_date  Convert date into day number

Description
Computes the day number corresponding to a given date (or vector of) from the first day of a start year. Typically, the start year should be the year of a STICS simulation start. Leap years are properly handled.

Usage
compute_day_from_date(
  date,
  start_year = NULL,
  start_date = lifecycle::deprecated()
)

Arguments
date  date(s) vector to be converted, in the character format ("YYYY-MM-DD") or Date format
start_year  year to be used as time reference (simulation start year). Optional.
start_date  [Deprecated] start_date is no longer supported, use start_year instead.

Value
numeric vector

Author(s)
Timothee Flutre
Examples

date <- as.Date("2015-02-10")
compute_day_from_date(date = date)

compute_day_from_date(date = "2015-02-10", start_year = 2014)

date <- as.Date("2009-02-10")
compute_day_from_date(date = date, start_year = 2008)

dates <- c(as.Date("2008-02-10"), as.Date("2009-02-10"))
compute_day_from_date(date = dates, start_year = 2008)

convert_xml2txt
Transforming a STICS xml file into a text file

Description

The input file according to his type (ini,plant,tec,station,soil,par) is converted to a text file readable by the STICS model (ficini.txt, ficplt1.txt,...)

Usage

```r
convert_xml2txt(
  file,
  plant_id = 1,
  out_dir = NULL,
  save_as = NULL,
  stics_version = "latest",
  xml_file = lifecycle::deprecated(),
  plt_num = lifecycle::deprecated(),
  out_file = lifecycle::deprecated()
)
```

Arguments

- `file` : Path (including name) of the xml file to convert
- `plant_id` : The plant identifier (main crop: 1 ; associated crop: 2)
- `out_dir` : Path of the directory where to generate the file. Optional, set to the path of the input xml file by default
- `save_as` : Name of the output file (optional, default: fixed name for STICS)
- `stics_version` : the STICS files version to use (optional, default to latest).
- `xml_file` : [Deprecated] xml_file is no longer supported, use file instead.
- `plt_num` : [Deprecated] plt_num is no longer supported, use plant_id instead.
- `out_file` : [Deprecated] out_file is no longer supported, use save_as instead.
download_data

Value
None

Examples

```r
## Not run:
xml_path <- "/path/to/corn_plt.xml"
javistics_path <- "/path/to/JavaSTICS/folder"
convert_xml2txt(file = xml_path, javistics = javistics_path)

## End(Not run)
```

---

download_data  

*Description*

Download locally the example data from the data repository in the SticsRPacks organisation.

*Usage*

```r
download_data(
  out_dir = tempdir(),
  example_dirs = NULL,
  stics_version = "latest",
  dir = lifecycle::deprecated(),
  version_name = lifecycle::deprecated()
)
```

*Arguments*

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>out_dir</td>
<td>Path of the directory where to download the data</td>
</tr>
<tr>
<td>example_dirs</td>
<td>List of use case directories names (optional)</td>
</tr>
<tr>
<td>stics_version</td>
<td>Name of the STICS version. Optional, by default the latest version returned by get_stics_versions_compat() is used.</td>
</tr>
<tr>
<td>dir</td>
<td>[Deprecated] dir is no longer supported, use out_dir instead.</td>
</tr>
<tr>
<td>version_name</td>
<td>[Deprecated] file_path is no longer supported, use file instead.</td>
</tr>
</tbody>
</table>

*Value*

The path to the folder where data have been downloaded

*Examples*

# Getting data for a given example : study_case_1 and a given STICS version
download_data(example_dirs = "study_case_1", stics_version = "V9.0")
download_usm_csv  

**Description**

The file is an example that can be used for generating JavaSTICS usms.xml input file from parameters values stored in a CSV file using the function `gen_usms_xml`.

**Usage**

```r
download_usm_csv(
  file = NULL,
  out_dir = tempdir(),
  stics_version = "latest",
  overwrite = FALSE,
  verbose = FALSE,
  csv_name = lifecycle::deprecated(),
  version_name = lifecycle::deprecated(),
  dest_dir = lifecycle::deprecated()
)
```

**Arguments**

- `file` Name of a csv file (optional, not used for the moment)
- `out_dir` Directory path where to copy the csv file (default: `tempdir()`)
- `stics_version` Name of the STICS version. Optional, by default the latest version returned by `get_stics_versions_compat()` is used.
- `overwrite` Optional logical, TRUE for overwriting files, FALSE otherwise (default)
- `verbose` Logical value for displaying information while running
- `csv_name` [Deprecated] `csv_name` is no longer supported, use `file` instead.
- `version_name` [Deprecated] `version_name` is no longer supported, use `stics_version` instead.
- `dest_dir` [Deprecated] `dest_dir` is no longer supported, use `out_dir` instead.

**Value**

A vector of copied files path.

**Examples**

```r
download_usm_csv()
```
Description

The file is an example that can be used for generating JavaSTICS input files from parameters values stored in Excel spreadsheet format (USMs, Ini, Soils, Tec, Station, ...). Each sheet contains parameters values to insert into XML files, with the help of these functions: `gen_usms_xml`, `gen_sols_xml`, `gen_tec_xml`, `gen_sta_xml`, `gen_usms_xml`, `gen_ini_xml`

Usage

download_usm_xl(
  file = NULL,
  out_dir = tempdir(),
  stics_version = "latest",
  overwrite = FALSE,
  verbose = FALSE,
  xl_name = lifecycle::deprecated(),
  version_name = lifecycle::deprecated(),
  dest_dir = lifecycle::deprecated(),
  ...
)

Arguments

- **file**: Name of an Excel file (optional, not used for the moment)
- **out_dir**: Directory path where to copy the Excel file (optional, default: `tempdir()`)
- **stics_version**: Name of the STICS version. Optional, by default the latest version returned by `get_stics_versions_compat()` is used.
- **overwrite**: Optional logical, TRUE for overwriting files, FALSE otherwise (default)
- **verbose**: Logical value for displaying information while running
- **xl_name**: [Deprecated] `xl_name` is no longer supported, use file instead.
- **version_name**: [Deprecated] `version_name` is no longer supported, use `stics_version` instead.
- **dest_dir**: [Deprecated] `dest_dir` is no longer supported, use `out_dir` instead.
- **...**: Additional arguments to be passed

Value

A vector of copied files path.

Examples

download_usm_xl()
force_param_values

Generates files to force parameters values in STICS simulations

Description

Generates a param.sti file and sets code optim in new_travail.usm to force parameters values in STICS simulations (this function is typically called before SticsOnR::run_stics())

Usage

```r
force_param_values(
  workspace,  # Path of the workspace containing the STICS (txt) input files.
  values,     # named vector of parameter values to force. See Details for more information.
  javastics,  # Path of JavaSTICS
  param_values = lifecycle::deprecated()
)
```

Arguments

- `workspace` Path of the workspace containing the STICS (txt) input files.
- `values` named vector of parameter values to force. See Details for more information.
- `javastics` Path of JavaSTICS
- `param_values` [Deprecated] param_values is no longer supported, use values instead.

Details

This function operates on STICS text input files. Do not use it before calling gen_usms_xml2txt(), otherwise param.sti and new_travail.usm files will be overwritten.

This function has been created to be called before SticsOnR::run_stics(). It can not work with SticsOnR::run_javastics(), that will overwrite param.sti and new_travail.usm files.

values can contain NA. In this case, the corresponding parameter(s) will not be forced (default value(s), i.e. read in STICS input files, will be used). If values==NA or values==NULL, not any parameter will be forced (all default values used).

Value

A logical status TRUE if successful, FALSE otherwise

See Also

SticsOnR::run_stics()
Examples

```r
## Not run:
example_txt_dir <- get_examples_path(file_type = "txt")
force_param_values(example_txt_dir,
                  setNames(object = c(220, 330), c("stlevamf", "stamflax")),
                  javastics = "/path/to/javastics"
)
## End(Not run)
```

### Description

Generate STICS ini xml file(s) from a template or an input file

### Usage

```r
gen_ini_xml(
  param_df,
  file = NULL,
  out_dir,
  crop_tag = "Crop",
  stics_version = "latest",
  ini_in_file = lifecycle::deprecated(),
  param_table = lifecycle::deprecated(),
  out_path = lifecycle::deprecated()
)
```

### Arguments

- **param_df**: A table (df, tibble) containing the values of the parameters to use (see details)
- **file**: Path of an ini xml file to be used as a template. Optional, if not provided, the function will use a standard template depending on the STICS version (see `stics_version` argument)
- **out_dir**: Path of the directory where to generate the file(s).
- **crop_tag**: Identifier for the crop parameters names related to the main crop, or the associated crop if any (example: Crop is used in the param_table example in the details section below)
- **stics_version**: Name of the STICS version. Optional, used if the file argument is not provided. In this case the function uses a standard template associated to the STICS version.
- **ini_in_file**: [Deprecated] ini_in_file is no longer supported, use file instead.
- **param_table**: [Deprecated] param_table is no longer supported, use param_df instead.
- **out_path**: [Deprecated] out_path is no longer supported, use out_dir instead.
Details

Please see `get_stics_versions_compat()` for the full list of STICS versions that can be used for the argument `stics_version`.

`param_df` is a `data.frame` with the following format:

<table>
<thead>
<tr>
<th>Ini_name</th>
<th>nbplantes</th>
<th>stade0_Crop1</th>
<th>lai0_Crop1</th>
<th>masec0_Crop1</th>
</tr>
</thead>
<tbody>
<tr>
<td>USM_2017_T1_ini.xml</td>
<td>1</td>
<td>snu</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Vill09_ini.xml</td>
<td>1</td>
<td>snu</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Vill10_ini.xml</td>
<td>1</td>
<td>snu</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Vill11_ini.xml</td>
<td>1</td>
<td>snu</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Vill12_ini.xml</td>
<td>1</td>
<td>snu</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Vill13_ini.xml</td>
<td>1</td>
<td>snu</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Vill14_ini.xml</td>
<td>1</td>
<td>snu</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Standard_ini.xml</td>
<td>1</td>
<td>snu</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The first column gives the ini file name (to be generated), all following columns give the parameter value to put in the file, and each line denotes a separate ini file (for e.g. several USMs).

The first column name must contain the keyword ini or Ini or INI as a prefix to be detected (as shown in the table extract above).

If not given (the default, `NULL`), the function returns the template as is.

Value

None

Examples

library(readxl)

xl_path <- download_usm xl(file = "inputs_stics_example.xlsx")
ini_param_df <- read_excel(xl_path, sheet = "Ini")
gen_ini_xml(
  out_dir = tempdir(),
  param_df = ini_param_df[1:2,]
)

Description

Generating observation data files from a data.frame
Usage

```r
gen_obs(
  df,
  out_dir,
  usms_list = NULL,
  obs_table = lifecycle::deprecated(),
  out_path = lifecycle::deprecated()
)
```

Arguments

- **df**
  A data frame containing the values of the observations to use (see Details).
- **out_dir**
  Path of the directory where to generate the file(s).
- **usms_list**
  An optional list of usms names to be used for selecting which files to generate from the `obs_table`
- **obs_table**
  [Deprecated] `obs_table` is no longer supported, use `df` instead.
- **out_path**
  [Deprecated] `out_path` is no longer supported, use `out_dir` instead.

Details

df is a `data.frame` with the following format:

```
  usm_name ian mo jo jul densite lai(n) masec(n) azomes
USM_2017_T1_CI 2017 9 6 249 NA NA 0.31 27.07395
USM_2017_T1_CI 2017 9 20 263 NA NA 0.60 27.90000
USM_2018_T1 2017 10 20 293 NA 0.1 NA NA
USM_2018_T1 2018 5 15 482 NA 1.2 NA NA
```

- `usm_name` column contains usms names which are used as output .obs files names
- `ian`, `mo`, `jo` and `jul` are mandatory (year, month, day and julian date)
- Other columns one per variable contain observations values or NA

@seealso `get_var_info` for getting variable right syntax or searching a variable name.

Value

A return logical status indicating if any error when writing files (FALSE), TRUE when no errors.

Examples

```r
xl_path <- download_usm_xl(file = "inputs_stics_example.xlsx")
obs_df <- read_params_table(file = xl_path, sheet_name = "Obs")
gen_obs(df = obs_df, out_dir = "/path/to/dest/dir")
```
Description

Generate STICS sols xml file from a template or an input file

Usage

gen_sols_xml(
  file,
  param_df,
  template = NULL,
  stics_version = "latest",
  sols_in_file = lifecycle::deprecated(),
  sols_param = lifecycle::deprecated(),
  sols_out_file = lifecycle::deprecated(),
  sols_nb = lifecycle::deprecated()
)

Arguments

file Path (including name) of the sols file to generate.
param_df A table (df, tibble) containing the values of the parameters to use (see details)
template Path of a soil xml file to be used as a template. Optional, if not provided, the function will use a standard template depending on the STICS version.
stics_version Name of the STICS version. Optional, used if the file argument is not provided. In this case the function uses a standard template associated to the STICS version.
sols_in_file [Deprecated] sols_in_file is no longer supported, use template instead.
sols_param [Deprecated] sols_param is no longer supported, use param_df instead.
sols_out_file [Deprecated] sols_out_file is no longer supported, use file instead.
sols_nb [Deprecated] sols_nb is no longer supported, it is now computed in the function.

Details

Please see get_stics_versions_compat() for the full list of STICS versions that can be used for the argument stics_version.

param_df is a data.frame with the following format:

<table>
<thead>
<tr>
<th>Soil_name</th>
<th>argi</th>
<th>norg</th>
<th>calc</th>
<th>pH</th>
<th>albedo</th>
<th>q0</th>
<th>epc_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>USM_T1</td>
<td>20.35</td>
<td>0.100</td>
<td>0.52</td>
<td>8.23</td>
<td>0.22</td>
<td>9.630</td>
<td>30</td>
</tr>
<tr>
<td>LF1</td>
<td>17.00</td>
<td>1.900</td>
<td>0.00</td>
<td>6.70</td>
<td>0.22</td>
<td>9.360</td>
<td>30</td>
</tr>
</tbody>
</table>
The first column gives the soil name, all following columns give the parameter values to put in the sols.xml file for each soil row.

The first column name must contain the keyword Soil or soil or SOIL as a prefix to be detected (as shown in the table extract above).

If not given (the default, NULL), the function returns the template as is.

### Value

None

### Examples

```r
xl_path <- download_usm_xl(file = "inputs_stics_example.xlsx")

sols_param_df <- read_params_table(file = xl_path, sheet_name = "Soils")
gen_sols_xml(file = file.path(tempdir(), "sols.xml"),
             param_df = sols_param_df)
```

---

**Description**

Generate STICS sta xml file(s) from a template or an input file

**Usage**

```r
gen_sta_xml(
  param_df,
  file = NULL,
  out_dir,
  stics_version = "latest",
  param_table = lifecycle::deprecated(),
)```

---

<table>
<thead>
<tr>
<th>Soil</th>
<th>Value1</th>
<th>Value2</th>
<th>Value3</th>
<th>Value4</th>
<th>Value5</th>
<th>Value6</th>
<th>Value7</th>
</tr>
</thead>
<tbody>
<tr>
<td>LF2</td>
<td>17.00000</td>
<td>1.800</td>
<td>0.00</td>
<td>6.70</td>
<td>0.22</td>
<td>9.360</td>
<td>30</td>
</tr>
<tr>
<td>LAP</td>
<td>22.00000</td>
<td>2.000</td>
<td>0.00</td>
<td>6.50</td>
<td>0.22</td>
<td>9.760</td>
<td>25</td>
</tr>
<tr>
<td>LAS</td>
<td>24.05000</td>
<td>2.500</td>
<td>30.00</td>
<td>8.00</td>
<td>0.22</td>
<td>9.928</td>
<td>30</td>
</tr>
<tr>
<td>LA0</td>
<td>30.00675</td>
<td>2.300</td>
<td>0.50</td>
<td>7.50</td>
<td>0.22</td>
<td>10.400</td>
<td>30</td>
</tr>
<tr>
<td>LC0</td>
<td>22.38750</td>
<td>2.000</td>
<td>10.00</td>
<td>7.90</td>
<td>0.22</td>
<td>9.792</td>
<td>25</td>
</tr>
<tr>
<td>Vill9</td>
<td>25.00000</td>
<td>0.101</td>
<td>0.40</td>
<td>7.90</td>
<td>0.22</td>
<td>10.000</td>
<td>30</td>
</tr>
<tr>
<td>Vill10</td>
<td>14.30000</td>
<td>0.099</td>
<td>1.50</td>
<td>8.20</td>
<td>0.22</td>
<td>9.144</td>
<td>30</td>
</tr>
<tr>
<td>Vill11</td>
<td>11.80000</td>
<td>0.100</td>
<td>0.00</td>
<td>7.30</td>
<td>0.22</td>
<td>8.944</td>
<td>30</td>
</tr>
<tr>
<td>Vill12</td>
<td>14.30000</td>
<td>0.091</td>
<td>0.60</td>
<td>8.30</td>
<td>0.22</td>
<td>9.144</td>
<td>30</td>
</tr>
<tr>
<td>Vill13</td>
<td>16.80000</td>
<td>0.088</td>
<td>0.20</td>
<td>7.80</td>
<td>0.22</td>
<td>9.344</td>
<td>30</td>
</tr>
<tr>
<td>Vill14</td>
<td>15.10000</td>
<td>0.095</td>
<td>1.30</td>
<td>7.90</td>
<td>0.22</td>
<td>9.208</td>
<td>30</td>
</tr>
</tbody>
</table>
sta_in_file = lifecycle::deprecated(),
out_path = lifecycle::deprecated()
)

Arguments

param_df  A table (df, tibble) containing the values of the parameters to use (see details)
file      Path of a sta xml file to be used as a template. Optional, if not provided, the
out_dir   Path of the directory where to generate the file(s).
stics_version Name of the STICS version. Optional, used if the file argument is not pro-
              vided. In this case the function uses a standard template associated to the STICS
param_table [Deprecated] param_table is no longer supported, use param_df instead.
sta_in_file [Deprecated] sta_in_file is no longer supported, use file instead.
out_path   [Deprecated] out_path is no longer supported, use out_dir instead.

Details

Please see get_stics_versions_compat() for the full list of STICS versions that can be used for
the argument stics_version.

param_df is a data.frame with the following format:

<table>
<thead>
<tr>
<th>Sta_name</th>
<th>zr</th>
<th>NH3ref</th>
<th>latitude</th>
<th>patm</th>
<th>aclim</th>
</tr>
</thead>
<tbody>
<tr>
<td>climatex_sta.xml</td>
<td>2.5</td>
<td>0</td>
<td>49</td>
<td>1000</td>
<td>20</td>
</tr>
<tr>
<td>climatex2_sta.xml</td>
<td>2.8</td>
<td>0</td>
<td>49</td>
<td>1000</td>
<td>20</td>
</tr>
<tr>
<td>climatex3_sta.xml</td>
<td>2.2</td>
<td>0</td>
<td>49</td>
<td>1000</td>
<td>20</td>
</tr>
</tbody>
</table>

The first column gives the sta file name (to be generated), all following columns give the parameter
value to put in the file, and each line denotes a separate sta file (for e.g. several USMs).
The first column name must contain the keyword sta or Sta or STA as a prefix to be detected (as
shown in the table extract above).
If not given (the default, NULL), the function returns the template as is.

Value

None

Examples

xl_path <- download_usm_xl(file = "inputs_stics_example.xlsx")
sta_param_df <- read_params_table(file = xl_path, sheet_name = "Station")
gen_sta_xml(out_dir = tempdir(), param_df = sta_param_df)
gen_tec_xml

Generate STICS tec xml file(s) from a template or an input file

Description

Generate STICS tec xml file(s) from a template or an input file

Usage

```r
gen_tec_xml(
  param_df = NULL,
  file = NULL,
  out_dir,
  stics_version = "latest",
  na_values = NA,
  param_table = lifecycle::deprecated(),
  tec_in_file = lifecycle::deprecated(),
  out_path = lifecycle::deprecated()
)
```

Arguments

- `param_df`: A table (df, tibble) containing the values of the parameters to use (see details)
- `file`: Path of a tec xml file to be used as a template. Optional, if not provided, the function will use a standard template depending on the STICS version.
- `out_dir`: Path of the directory where to generate the file(s).
- `stics_version`: Name of the STICS version. Optional, used if the `file` argument is not provided. In this case the function uses a standard template associated to the STICS version.
- `na_values`: value to use as missing value in param_table (optional, default : NA)
- `param_table`: [Deprecated] param_table is no longer supported, use param_df instead.
- `tec_in_file`: [Deprecated] tec_in_file is no longer supported, use file instead.
- `out_path`: [Deprecated] out_path is no longer supported, use out_dir instead.

Details

Please see get_stics_versions_compat() for the full list of STICS versions that can be used for the argument stics_version.

param_df is a data.frame with the following format:

<table>
<thead>
<tr>
<th>Tec_name</th>
<th>julres_1</th>
<th>coderes_1</th>
</tr>
</thead>
<tbody>
<tr>
<td>USM_2017_T1_CI_tec.xml</td>
<td>NA</td>
<td>1</td>
</tr>
<tr>
<td>BIN_CANPC_05_SEC_220-0-0_34K_CANPC05T3_Q_tec.xml</td>
<td>110</td>
<td>1</td>
</tr>
<tr>
<td>BIN_AGT_04_IRR_220-0-0_33K_AGT04T2_Q_tec.xml</td>
<td>73</td>
<td>1</td>
</tr>
</tbody>
</table>
The first column gives the tec file name (to be generated), all following columns give the parameter value to put in the file, and each line denotes a separate tec file (for e.g. several USMs).

The first column name must contain the keyword tec or Tec or TEC as a prefix to be detected (as shown in the table extract above).

If not given (the default, NULL), the function returns the template as is.

**Value**

None

**Examples**

```r
xl_path <- download_usm_xl(file = "inputs_stics_example.xlsx")
tec_param_df <- read_params_table(file = xl_path, sheet_name = "Tec")
gen_tec_xml(out_dir = tempdir(), param_df = tec_param_df[1:2, ])
```

---

**Description**

Generate STICS usms xml file from a template or an input file

**Usage**

```r
gen_usms_xml(
  file,
  param_df = NULL,
  template = NULL,
  stics_version = "latest",
  usms_out_file = lifecycle::deprecated(),
  usms_nb = lifecycle::deprecated(),
  usms_param = lifecycle::deprecated(),
  usms_in_file = lifecycle::deprecated()
)
```
Arguments

- **file**: Path (including name) of the usms file to generate.
- **param_df**: A table (df, tibble) containing the values of the parameters to use (see details).
- **template**: Path of an USM xml file to be used as a template. Optional, if not provided, the function will use a standard template depending on the STICS version.
- **stics_version**: Name of the STICS version. Optional, used if the file argument is not provided. In this case the function uses a standard template associated to the STICS version.
- **usms_out_file**: [Deprecated] usms_out_file is no longer supported, use file instead.
- **usms_nb**: [Deprecated] usms_nb is no longer supported, use NA instead.
- **usms_param**: [Deprecated] usms_param is no longer supported, use param_df instead.
- **usms_in_file**: [Deprecated] usms_in_file is no longer supported, use template instead.

Details

Please see `get_stics_versions_compat()` for the full list of STICS versions that can be used for the argument stics_version.

**param_df** is a data.frame with the following format:

<table>
<thead>
<tr>
<th>usm_name</th>
<th>datedebut</th>
<th>datefin</th>
<th>nomsol</th>
</tr>
</thead>
<tbody>
<tr>
<td>USM_2017_T1_CI</td>
<td>199</td>
<td>263</td>
<td>USM_T1</td>
</tr>
<tr>
<td>USM_2018_T1</td>
<td>264</td>
<td>570</td>
<td>USM_T1</td>
</tr>
<tr>
<td>BIN_CANPC_05_SEC_220-0-0_34K_CANPC05T3_Q</td>
<td>199</td>
<td>263</td>
<td>LF1</td>
</tr>
<tr>
<td>BIN_AGT_04_IRR_220-0-0_33K_AGT04T2_Q</td>
<td>264</td>
<td>570</td>
<td>LF1</td>
</tr>
<tr>
<td>AGA_ARB_13_IRR_220-0-0_37K_ARB13_C</td>
<td>199</td>
<td>263</td>
<td>F1</td>
</tr>
<tr>
<td>AGA_ARB_13_SEC_220-0-0_37K_ARB13_C</td>
<td>264</td>
<td>570</td>
<td>LF1</td>
</tr>
<tr>
<td>FRA_ARB_11_SEC_220-0-0_38K_E</td>
<td>199</td>
<td>263</td>
<td>LF1</td>
</tr>
<tr>
<td>MAG_ARB_09_SEC_220-0-0_38K_E</td>
<td>264</td>
<td>570</td>
<td>LF1</td>
</tr>
<tr>
<td>MAG_ARV_12_IRR_220-0-0_36K_ARV12_C</td>
<td>199</td>
<td>263</td>
<td>LF1</td>
</tr>
<tr>
<td>MAG_ARV_12_SEC_220-0-0_36K_ARV12_C</td>
<td>264</td>
<td>570</td>
<td>LF1</td>
</tr>
<tr>
<td>FRA_ARB_12_SEC_220-0-0_31K_ARB12_C</td>
<td>199</td>
<td>263</td>
<td>LF1</td>
</tr>
<tr>
<td>FRA_ARB_13_SEC_220-0-0_37K_ARB13_C</td>
<td>264</td>
<td>570</td>
<td>LF1</td>
</tr>
</tbody>
</table>

The first column gives the usm name, all following columns give the parameter values to put in the usms.xml file for each usm row.

The first column name must contain the keyword Usm or usm or USM as a prefix to be detected (as shown in the table extract above).

If not given (the default, NULL), the function returns the template as is.

**Value**

an invisible xml_document object
Examples

```r
xl_path <- download_usm xls(file = "inputs_stics_example.xlsx")
usms_param_df <- read_params_table(file = xl_path, sheet_name = "USMs")
gen_usms_xml(file = file.path(tempdir(), "usms.xml"),
param_df = usms_param_df)
```

---

**gen_usms_xml2txt**  Generating one or several usms directories from a javastics workspace content

**Description**

The function creates sets of input files for one or multiple usms from usms data stored in a JavaSTICS workspace. For multiple usms, sets will be generated into individual folders named with usm names. Observations files will be also copied if they are named `[usm_name].obs` For one usm, files will be generated either in the workspace directory or in a subdirectory.

**Usage**

```r
gen_usms_xml2txt(
  javastics = NULL,
  workspace = NULL,
  out_dir = NULL,
  usm = c(),
  stics_version = "latest",
  verbose = TRUE,
  dir_per_usm_flag = TRUE,
  java_cmd = "java",
  java_converter = FALSE,
  javastics_path = lifecycle::deprecated(),
  workspace_path = lifecycle::deprecated(),
  target_path = lifecycle::deprecated(),
  usms_list = lifecycle::deprecated()
)
```

**Arguments**

- **javastics**  Path of JavaSTICS. Optional (needed if the JavaSTICS converter is used, java_converter set to TRUE in inputs)
- **workspace**  Path of a JavaSTICS workspace (i.e. containing the STICS XML input files). Optional, if not provided the current workspace stored in JavaSTICS preferences will be used.
- **out_dir**  The path of the directory where to create usms directories (Optional), if not provided the JavaSTICS workspace will be used as root
- **usm**  List of usms to generate (Optional). If not provided, all usms contained in workspace/usms.xml file will be generated.
### gen_varmod

Generating a var.mod type file

**Description**

Generating a daily variable list file from variables names

```r
## Not run:
javastics <- "/path/to/JavaSTICS/folder"
workspace <-="/path/to/workspace"

# For all usms
gen_usms_xml2txt(javistics, workspace)

# For an usms list
usm <- c("usm1", "usm2")
gen_usms_xml2txt(javistics, workspace, usm)
```

```r
## End(Not run)
```
Usage

```r
gen_varmod(
  workspace,
  var,
  append = FALSE,
  file_name = "var.mod",
  stics_version = "latest",
  force = FALSE,
  var_names = lifecycle::deprecated(),
  version = lifecycle::deprecated()
)
```

Arguments

- **workspace**: Path of the directory containing the STICS var.mod file to modify
- **var**: vector of variables names (see details)
- **append**: if TRUE, var data are appended to file_name
- **file_name**: file name to generate (without path, default value: "var.mod")
- **stics_version**: Name of the STICS version (used to check variable names)
- **force**: Force variables writing even if they are not a STICS variable (default: FALSE).
- **var_names**: [Deprecated] var_names is no longer supported, use var instead.
- **version**: [Deprecated] version is no longer supported, use stics_version instead.

Details

Variable names can be found using `get_var_info()`. They are checked before writing. If any variable name does not exist, it will not be written by default, but the function will still write the variables that exist. `force` = TRUE may however be used to write variables that do not exist.

Value

None

Examples

```r
gen_varmod(tempdir(), c("lai(n)", "hauteur"))
# Add a variable to the others:
gen_varmod(tempdir(), "masec(n)", append = TRUE)
# NB: var.mod will have "lai(n)","hauteur" and "masec(n)"
```
**get_climate_txt**  
*Read STICS input meteorology file*

**Description**

Read the meteorology input for STICS ("climat.txt")

**Usage**

```r
get_climate_txt(
  workspace,
  file_name = "climat.txt",
  preserve = TRUE,
  dirpath = lifecycle::deprecated(),
  filename = lifecycle::deprecated()
)
```

**Arguments**

- `workspace`  
  Path of the workspace containing the STICS climate file to read

- `file_name`  
  The meteorology file name (default to `climat.txt`).

- `preserve`  
  Logical, `TRUE` for keeping the STICS columns related to date calculation (year, month, day) as `POSIXct`, or only the Date column as a `POSIXct` otherwise. Default to `TRUE`.

- `dirpath`  
  [Deprecated] `dirpath` is no longer supported, use `workspace` instead.

- `filename`  
  [Deprecated] `filename` is no longer supported, use `file_name` instead.

**Value**

A data.frame of the input meteorological variables used as input for the STICS model.

**Note**

The time-related variables are summarised into one `POSIXct` column named `date`.

**Examples**

```r
path <- get_examples_path(file_type = "txt")
Meteo <- get_climate_txt(path)
```
get_cultivars_list

Get the cultivar names for an xml plant file (*_plt.xml)

Description
Extracts the cultivar names from a plant file

Usage
get_cultivars_list(file)

Arguments
file The path of a plant file.

Value
A vector of cultivar names

Examples
path <- get_examples_path(file_type = "xml")
# Read from a plant file (all cultivars available in a plant file)
cv_list <- get_cultivars_list(file = file.path(path, "file_plt.xml"))

get_cultivars_param

Get the values of cultivar-specific parameters from an xml plant file (*_plt.xml)

Description
Extracts the values of cultivar-specific parameters from a plant file

Usage
get_cultivars_param(file)

Arguments
file The path of a plant file.

Value
A data.frame with one row per cultivar and one column per parameter
get_examples_path

Examples

```r
path <- get_examples_path(file_type = "xml")

# Read from a plant file (all cultivars available in a plant file)
cv_param_df <- get_cultivars_param(file = file.path(path, "file_plt.xml"))
```

---

**Description**

Getting examples files path attached to a STICS version for a given file type

**Usage**

```r
get_examples_path(
  file_type, 
  stics_version = "latest", 
  overwrite = FALSE, 
  version_name = lifecycle::deprecated()
)
```

**Arguments**

- `file_type` A file type string among files types or a vector of ("csv", "obs", "sti", "txt", "xml")
- `stics_version` Name of the STICS version. Optional, by default the latest version returned by `get_stics_versions_compat()` is used.
- `overwrite` TRUE for overwriting directory; FALSE otherwise
- `version_name` [Deprecated] version_name is no longer supported, use stics_version instead.

**Value**

A directory path for examples files for given file type and STICS version or a vector of (for unknown file types "" is returned as path)

**Examples**

```r
get_examples_path(file_type = "csv")
get_examples_path(file_type = c("csv", "sti"))
get_examples_path(file_type = "csv", stics_version = "V8.5")
```
get_lai_forcing  Getting LAI forcing for each usm

Description
Is LAI forced for usms in usms.xml

Usage
get_lai_forcing(usm_file_path, usms_list = c())

Arguments
usm_file_path  Path to usms.xml file
usms_list      Usm(s) name(s) (optional, see details)

Details
Use get_usms_list() to get the list of the usm names for an usms.xml file.

Value
A named numeric vector with a Boolean value (TRUE = forced) for each usm

Examples
# Xml case
xml_usms <- file.path(get_examples_path(file_type = "xml"), "usms.xml")
get_lai_forcing(xml_usms)
get_lai_forcing(xml_usms, "wheat")
get_lai_forcing(xml_usms, c("wheat", "intercrop_pea_barley"))

get_obs  Read STICS observation files (*.obs)

Description
Read STICS observation files from a JavaSTICS workspace and store data into a list per usm
get_obs

Usage

get_obs(
  workspace,
  usm = NULL,
  var = NULL,
  dates = NULL,
  usms_file = NULL,
  javastics = NULL,
  verbose = TRUE,
  usm_name = lifecycle::deprecated(),
  var_list = lifecycle::deprecated(),
  dates_list = lifecycle::deprecated(),
  usms_filepath = lifecycle::deprecated(),
  javastics_path = lifecycle::deprecated()
)

Arguments

workspace  Vector of path(s) of directory(ies) containing the STICS observation files to read (*obs file) or path of a single directory containing one sub-folder per USM (named as the USM names), each of them containing the corresponding files to read. In the second case, the argument usm must also be provided.
usm        Vector of USM names. Optional, if not provided, the function returns the results for all USMs.
var        Vector of variable names for which results have to be provided. Optional, all variables considered by default. See get_var_info() to get the list of STICS variables names.
dates      list of dates to filter (POSIX date)
usms_file  Path of a USM xml file. Optional, if provided, the plant names are added in the Plant column (see details).
javastics  Path of JavaSTICS. Optional, should be provided in addition to usms_file to get the plant codes if the plant files used are not in the workspace but in the JavaSTICS distribution (see Details).
verbose    Logical value for displaying or not information while running
usm_name   [Deprecated] usm_name is no longer supported, use usm instead.
var_list   [Deprecated] var_list is no longer supported, use var instead.
dates_list [Deprecated] dates_list is no longer supported, use dates instead.
usms_filepath [Deprecated] usms_filepath is no longer supported, use usms_file instead.
javastics_path [Deprecated] javastics_path is no longer supported, use javastics instead.

Details

The .obs files names must match USMs names, e.g. for a usm called "banana", the .obs file should be named banana.obs. For intercrops, the name should be suffixed by "p" for the principal and "a" for the associated plant.
If `usm` is not specified (or equal to NULL), the function reads the files from all usms in the workspace(s).

If `usms_file` is provided and if the associated plant file is found, the plant names in the "Plant" column of the generated `data.frame` are either the plant code (as specified in the plant file) or the name of the plant file, if the plant file is not found.

If `usms_file` is not specified, the plants are named "plant_1" by default (+ "plant_2" for intercrops).

**Value**

A list, where each element is a `data.frame` of observations for the given usm. The list is named after the USM name.

Intercrops are returned in a single `data.frame`, and are identified using either the "Plant" or "Dominance" columns.

See Details section for more information about the "Plant" column.

**Examples**

```r
path <- file.path(get_examples_path(file_type = "obs"), "mixed")

# Get observations for all usms, but only banana has observations:
Meas <- get_obs(path)

# Get observations only for banana:
Meas_banana <- get_obs(path, "banana")

## Not run:
# Get observations with real plant names when plant
# folder is not in the workspace:
get_obs(path, "banana", javastics = "/path/to/JavaSTICS/folder")

## End(Not run)
```

---

**Description**

Helper function that returns names and descriptions of STICS input parameters from a partial name and/or descriptive keywords.

**Usage**

```r
get_param_info(param = NULL, keyword = NULL, stics_version = "latest")
```
get_param_txt

Arguments

- **param**: Vector of parameter names (or partial names). Optional, if not provided, the function returns information for all parameters
- **keyword**: Optional, strings or a vector of to be used for searching in parameters names and definition
- **stics_version**: Name of the STICS version. Optional, can be used to search parameters information relative to a specific STICS version. By default the latest version returned by `get_stics_versions_compat()` is used.

Details

The function understand \texttt{regex} as input.

Value

A data.frame with information about parameter(s) with columns \texttt{name}, \texttt{file}, \texttt{min}, \texttt{max}, \texttt{definition}

Examples

```r
# Find by parameter name (fuzzy search):
SticsRFiles::get_param_info("alb")
SticsRFiles::get_param_info("alb[e]?")

# Find by keyword (fuzzy search in parameter name and description):
SticsRFiles::get_param_info(keyword = "bdil")

# Find for a particular version:
SticsRFiles::get_param_info("alb", stics_version = "V9.0")
```

---

**get_param_txt**  \hspace{1cm} \textit{Read STICS input parameters from text files}

Description

Read STICS model input parameters from a usm in text format (STICS input) Generally used after calling building a usm with JavaSTICS.

Read a specific STICS model input parameter file. Users would generally use the wrapper `get_param_txt()` instead.
Usage

get_param_txt(
  workspace,
  param = NULL,
  plant_id = NULL,
  variety = NULL,
  value_id = NULL,
  exact = FALSE,
  stics_version = "latest",
  dirpath = lifecycle::deprecated(),
  ...)

get_ini_txt(
  file = "ficini.txt",
  stics_version,
  filepath = lifecycle::deprecated()
)

general txt(file = "tempopar.sti", filepath = lifecycle::deprecated())

get_tec_txt(file = "tempoparv6.sti", filepath = lifecycle::deprecated())

get_plant_txt(
  file = "ficplt1.txt",
  variety = NULL,
  filepath = lifecycle::deprecated()
)

get_tec_txt(file = "fictep1.txt",
  stics_version = "latest",
  several_fert = NULL,
  several_thin = NULL,
  is_pasture = NULL,
  filepath = lifecycle::deprecated(),
  ...
)

get_soil_txt(file = "param.sol",
  stics_version,
  filepath = lifecycle::deprecated()
)

get_station_txt(file = "station.txt", filepath = lifecycle::deprecated())

get_usm_txt
get_param_txt

```r
file = "new_travail.usm",
plant_id = NULL,
filepath = lifecycle::deprecated()
```

**Arguments**

- `workspace` Path of the workspace containing the STICS (txt) input files.
- `param` Vector of parameter names. Optional, if not provided, the function returns an object with all parameters.
- `plant_id` Plant index (1, 2), default(NULL) calculated from from plant number in STICS initialization file.
- `variety` Integer. The plant variety to get the parameter from.
- `value_id` Index of technical interventions to be used to retrieve parameter values, or layer index for soil parameters.
- `exact` Boolean indicating if the function must return results only for exact match.
- `stics_version` An optional version name as listed in get_stics_versions_compat() return.
- `dirpath` [Deprecated] dirpath is no longer supported, use `workspace` instead.
- `...` Further arguments to pass (for future-proofing only)
- `file` File path.
- `filepath` [Deprecated] filepath is no longer supported, use `file` instead.
- `several_fert` Is there several fertilization in the USM? See details.
- `several_thin` Is there several thinning in the USM? See details.
- `is_pasture` Is the plant a pasture? See details.

**Details**

If the `variety` is not given and a `param` is asked, the function will return the values for the variety that is simulated in the USM by checking the `variete` parameter in the technical file. If `param` is not provided by the user, the values from all varieties will be returned unless the user ask for a given `variety`.

`several_fert`, `several_thin` and `is_pasture` are read from the tmp file (`tempoparv6.sti`). `get_param_txt()` does it automatically. If you absolutely need to use directly `get_tec_txt`, please see example.

**Value**

A list of parameters value(s), or if `param = NULL` a list of all parameters:

<table>
<thead>
<tr>
<th>ini</th>
<th>Initialization parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>general</td>
<td>General parameters</td>
</tr>
<tr>
<td>tec</td>
<td>Technical parameters</td>
</tr>
<tr>
<td>plant</td>
<td>Plant parameters</td>
</tr>
<tr>
<td>soil</td>
<td>Soil parameters</td>
</tr>
</tbody>
</table>
station  Station parameters

A list of parameters, depending on the file/function:

ini  Initialization parameters

general  General parameters

tec  Technical parameters

plant  Plant parameters

soil  Soil parameters

station  Station parameters

tmp  Temporary parameters

Note

Users would generally use `get_param_txt` to identify parameters names and values and pass them to other functions.

The functions are compatible with intercrops. Users generally only use `get_param_txt()`, which is a wrapper for all these functions.

See Also

gen_varmod(),

get_param_txt().

Examples

```r
path <- get_examples_path(file_type = "txt")

# Getting the interrow distance parameter value
get_param_txt(path, param = "interrang")

# Getting varietal parameters values
# Get the leaf lifespan of the variety used in the usm:
get_param_txt(workspace = path, param = "durvieF")

# Get the leaf lifespan of another variety available in the plant file:
get_param_txt(workspace = path, param = "durvieF", variety = "Furio")

# To get the values for several (or all) varieties, either put all varieties:
varieties <- c("Pactol", "Cherif", "Furio", "Dunia", "Volga", "Cecilia")
get_param_txt(workspace = path, param = "durvieF", variety = varieties)

# Or get it from the output of the function returning all parameters:
get_param_txt(workspace = path)$plant$plant1$durvieF

# Get parameters for a specific plant
get_param_txt(workspace = path, plant_id = 1)
get_param_txt(workspace = path, param = "durvieF", plant_id = 1)
get_param_txt(workspace = path, param = "durvieF", plant_id = 1, variety = varieties)

# Get parameters for specific interventions or soil layers
```
get_param_xml

get_param_txt(workspace = path, param = "amount", value_id = c(1,3))
get_param_txt(workspace = path, param = "Hinitf", value_id = c(1,3))
get_param_txt(workspace = path, param = "epc", value_id = c(1,3))

## Not run:
# Read the initialisation file (ficini.txt):
library(SticsRFiles)
path <- file.path(get_examples_path(file_type = "txt"), "ficini.txt")
get_ini_txt(path)

# Read the tec file directly:

# First, get the parameters from the tmp file:
tmp <- get_tmp_txt(file = file.path(get_examples_path(file_type = "txt"), "tempoparv6.sti"))
several_fert <- ifelse(tmp$option_engrais_multiple == 1, TRUE, FALSE)
several_thin <- ifelse(tmp$option_thinning == 1, TRUE, FALSE)
is_pasture <- ifelse(tmp$option_pature == 1, TRUE, FALSE)

# Then, get the technical parameters:
get_tec_txt(
  file = file.path(get_examples_path(file_type = "txt"), "fictec1.txt"),
  several_fert = several_fert, several_thin = several_thin,
  is_pasture = is_pasture
)

## End(Not run)

get_param_xml

Getting parameter values from xml files

Description

Extracting parameter values for a list of xml files and parameters

Usage

get_param_xml(
  file,
  param = NULL,
  select = NULL,
  select_value = NULL,
  value_id = NULL,
  xml_file = lifecycle::deprecated(),
  param_name = lifecycle::deprecated(),
  value = lifecycle::deprecated(),
  ...
)
**get_param_xml**

**Arguments**

- `file`: Vector of the xml file paths from which parameters values must be extracted.
- `param`: Vector of parameter names. Optional, if not provided, the function returns information for all parameters.
- `select`: Node name or attribute name to use for selection (optional, default to no selection).
- `select_value`: Vector of values used for select (see examples). Optional, should be provided only if select is provided.
- `value_id`: Vector of ids of the parameters values to be retrieved from the parameter values vector.
- `xml_file`: [Deprecated] `xml_file` is no longer supported, use `file` instead.
- `param_name`: [Deprecated] `param_name` is no longer supported, use `param` instead.
- `value`: [Deprecated] `value` is no longer supported, use `select_value` instead.

... Pass further arguments to `get_param_value()`.

**Value**

A list of parameter values for each xml_file (a list of list)

**Examples**

```r
# Soil file
file <- file.path(get_examples_path(file_type = "xml"), "sols.xml")

# For all soils
get_param_xml(file)
get_param_xml(file, c("argi", "norg"))

# With soil selection
# scalar parameters per soil
get_param_xml(file, c("argi", "norg"),
  select = "sol", select_value = c("solcanne", "solbanane")
)

# Crop management file
file <- file.path(get_examples_path(file_type = "xml"), "file_tec.xml")

# Getting parameters for irrigation (date and quantity)
get_param_xml(file, c("julapI_or_sum_upvt", "amount"))
```
get_plants_nb

**Description**

Extracting plant number from usms.xml or new_travail.usm file data

**Usage**

```r
get_plants_nb(
  usms_file,
  usms_list = c(),
  usm_file_path = lifecycle::deprecated()
)
```

**Arguments**

- `usms_file`: Path (including name) of a USM xml file or of a new_travail.usm file
- `usms_list`: Usms(s) name(s) (optional, see details)
- `usm_file_path`: [Deprecated] usm_file_path is no longer supported, use usms_file instead.

**Details**

Use `get_usms_list()` to get the list of the usm names for an usms.xml file.

**Value**

A named numeric vector of plants number per usm

**Examples**

```r
# Xml case
xml_usms <- file.path(get_examples_path(file_type = "xml"), "usms.xml")
get_plants_nb(xml_usms)
get_plants_nb(xml_usms, "wheat")
get_plants_nb(xml_usms, c("wheat", "intercrop_pea_barley"))

# Txt case
txt_usm <- file.path(get_examples_path(file_type = "txt"), "new_travail.usm")
get_plants_nb(txt_usm)
```
get_report_results  

Extracting data from the STICS report file

Description

Extracting data from the STICS report file

Usage

```r
get_report_results(
  workspace,  
  file_name = "mod_rapport.sti",  
  usm = NULL,  
  var_list = NULL,  
  usm_name = lifecycle::deprecated()
)
```

Arguments

- **workspace**: Path of the directory containing the STICS report file to read.
- **file_name**: A report file name among "mod_rapport.sti" (default), "mod_rapportA.sti", "mod_rapportP.sti"
- **usm**: Vector of USM names. Optional, if not provided, the function returns the results for all USMs.
- **var_list**: vector of output variables names to filter (optional, see `get_var_info()` to get the names of the variables)
- **usm_name**: [Deprecated] `usm_name` is no longer supported, use `usm` instead.

Details

The data may be filtered using `usm_name` vector of usm names and and/or `var_list` vector of variables names. In the returned data.frame, variables names respect the same syntax as in the `get_sim` output.

Value

A data.frame

Examples

```r
path <- get_examples_path(file_type = "sti")
get_report_results(workspace = path)

get_report_results(workspace = path, usm = c("DurumWheat", "grass"))

get_report_results(workspace = path, var_list = c("masec(n)", "QNplante"))

get_report_results(workspace = path, usm = c("DurumWheat", "grass"))
```
get_sim

get_report_results(workspace = path)

get_report_results(workspace = path, file_name = "mod_rapportA.sti")

get_sim 

Load and format STICS daily output file(s)

Description

Reads and format daily output file(s) (mod_s*.sti) for usm(s) with possible selection on variable names, cumulative DOY and dates

Usage

get_sim(
    workspace,
    usm = NULL,
    var = NULL,
    dates = NULL,
    usms_file = NULL,
    javastics = NULL,
    verbose = TRUE,
    usm_name = lifecycle::deprecated(),
    var_list = lifecycle::deprecated(),
    dates_list = lifecycle::deprecated(),
    usms_filepath = lifecycle::deprecated(),
    javastics_path = lifecycle::deprecated()
)

Arguments

workspace Vector of path(s) of directory(ies) containing the STICS output files to read (mod_s*.sti file) or path of a single directory containing one sub-folder per USM (named as the USM names), each of them containing the corresponding STICS output file to read. In the second case, the argument usm must also be provided.

usm Vector of USM names. Optional, if not provided, the function returns the results for all USMs.

var Vector of variable names for which results have to be provided. Optional, all variables considered by default. See get_var_info() to get the list of STICS variables names.

dates list of dates to filter (POSIX date)

usms_file Path of a USM xml file. Optional, if provided, the plant names are added in the Plant column (see details).
get_soils_list

Path of JavaSTICS Optional, should be provided in addition to usms_file to get the plant codes if the plant files used are not in the workspace but in the JavaSTICS distribution (see Details).

verbose Logical value for displaying or not information while running

usm_name [Deprecated] usm_name is no longer supported, use usm instead.

var_list [Deprecated] var_list is no longer supported, use var instead.

dates_list [Deprecated] dates_list is no longer supported, use dates instead.

usms_filepath [Deprecated] usms_filepath is no longer supported, use usms_file instead.

javastics_path [Deprecated] javastics_path is no longer supported, use javastics instead.

Details
If usm is not specified (or equal to NULL), the function reads the files from all usms in the workspace(s).
If usms_file is provided and if the associated plant file is found, the plant names in the "Plant" column of the generated data.frame are either the plant code (as specified in the plant file) or the name of the plant file, if the plant file is not found.
If usms_file is not specified, the plants are named "plant_1" by default (+ "plant_2" for intercrops).

Value
A list, where each element is a data.frame of simulation results for the given usm. The list is named after the USM name.
Intercrops are returned in a single data.frame, and are identified using either the "Plant" or "Dominance" columns.
See Details section for more information about the "Plant" column.

Examples
path <- get_examples_path(file_type = "sti")
sim_data <- get_sim(path, "banana")

get_soils_list Get the soil names for an usms.xml file

Description
Extracts the soil names from a "usms.xml" file, or from a soil file

Usage
get_soils_list(
  file,
  soil = NULL,
  file_path = lifecycle::deprecated(),
  name = lifecycle::deprecated()
)
get_stics_versions_compat

Arguments

- **file**: Either the path of an usm file or of a soil file.
- **soil**: Vector of soil names (or partial names). Optional, if not provided, the function returns the names of all the soils included in the given file.
- **file_path**: [Deprecated] file_path is no longer supported, use file instead.
- **name**: [Deprecated] name is no longer supported, use soil instead.

Details

The file given as the file_path is either a "usms" file type to get all the soils used in a particular USM, or a soil file type ("sols") to get all soil types available in a soil file.

Value

A vector of soil names

Examples

```r
path <- get_examples_path(file_type = "xml")

# Read from a usms file (soils used in a USM):
soil_list <- get_soils_list(file = file.path(path, "usms.xml"))

# Read from a soil file (all soil types available in a soil file)
soil_list <- get_soils_list(file = file.path(path, "sols.xml"))

soil_list <- get_soils_list(file = file.path(path, "usms.xml"),
                            soil = c("solcanne", "sole"))
```

get_stics_versions_compat

*Get the compatible STICS versions*

Description

Get the versions of STICS that are fully compatible with this package.

Usage

```r
get_stics_versions_compat(version_index = NULL)
```

Arguments

- **version_index**: Absolute positive index, or negative relative index from latest version
Value

A named list with the STICS versions compatible with this package ($versions_list), and the latest version in use ($latest_version) or an existing version selected using version_index.

Examples

# Getting the complete versions list
get_stics_versions_compat()

# Getting the first version
get_stics_versions_compat(1)

# Getting the previous version of the latest one
get_stics_versions_compat(-1)

get_usms_files

Description

Getting existing xml files path list per usm from an usms.xml file

Usage

get_usms_files(
  workspace,
  usms_list = NULL,
  usms_file = "usms.xml",
  file_type = NULL,
  javastics = NULL,
  df_output = FALSE,
  workspace_path = lifecycle::deprecated(),
  file_name = lifecycle::deprecated(),
  javastics_path = lifecycle::deprecated()
)

Arguments

workspace Path of a JavaSTICS workspace (i.e. containing the STICS XML input files)
usms_list Vector of usms names (Optional)
usms_file Path (including name) of a USM XML file.
file_type Vector of file(s) type to get (if not given, all types are returned, see details)
javastics Path of JavaSTICS Optional, only needed if the plant files are not in the workspace (in this case the plant files used are those included in the JavaSTICS distribution)
**get_usms_list**

get_usms_list is a function that extracts the list of USM (Urban Soil Management) names from an USMS (Urban Soil Management System) file.

### Arguments

- `workspace_path`: [Deprecated] workspace_path is no longer supported, use workspace instead.
- `file_name`: [Deprecated] file_name is no longer supported, use usms_file instead.
- `javastics_path`: [Deprecated] javastics_path is no longer supported, use javastics instead.
- `df_output`: logical if TRUE returning a data.frame, otherwise returning a named list if FALSE (default)
- `workspace`: path to the workspace
- `javastics`: path to the JavaSTICS folder
- `file_type`: character vector of file types to be returned (default: all)
- `usm_list`: character vector of USM names to be returned

### Value

A named list with existing files path in each usm element

### Examples

```r
## Not run:
get_usms_files(
  workspace = "/path/to/workspace",
  javastics = "/path/to/JavaSTICS/folder"
)
get_usms_files(
  workspace = "/path/to/workspace",
  javastics = "/path/to/JavaSTICS/folder",
  usm_list = c("usm1", "usm3")
)
get_usms_files(
  workspace = "/path/to/workspace",
  file_type = c("finit", "ftec")
)
## End(Not run)
```

### Description

Extracting a usm names list from an usms.xml file
get_usms_list(
    file,
    usm = NULL,
    usm_path = lifecycle::deprecated(),
    name = lifecycle::deprecated()
)

Arguments

file Path (including name) of the USM xml file
usm Vector of USM names (or partial names). Optional, if not provided, the function returns the names of all the USMs included in the given file.
usm_path [Deprecated] usm_path is no longer supported, use file instead.
name [Deprecated] name is no longer supported, use usm instead.

Value

A vector of usm names

Examples

path <- get_examples_path(file_type = "xml")

usms_list <- get_usms_list(file = file.path(path, "usms.xml"))

usms_list <- get_usms_list(file = file.path(path, "usms.xml"),
    usm = c("usm1", "usm2"))

get_varmod

Get desired STICS outputs

Description

Get the STICS output variables (from var.mod file)

Usage

get_varmod(workspace, file_name = "var.mod")

Arguments

workspace Path of the directory containing the STICS var.mod file
file_name file name to read (without path, default value: "var.mod")
get_var_info

Value

The variables that will be returned by STICS

See Also

gen_varmod

Examples

gem_varmod(get_examples_path(file_type = "txt"))

get_var_info  Find STICS output variable names and description

Description

Helper function that returns names and descriptions of STICS output variables from a partial name
and/or descriptive keywords.

Usage

gem_varinfo(var = NULL, keyword = NULL, stics_version = "latest")

Arguments

var  Vector of variable names (or partial names). Optional, if not provided, the function returns information for all variables.

keyword  Search by keyword instead of variable name (search in the name and description field)

stics_version  Name of the STICS version. Optional, can be used to search parameters information relative to a specific STICS version. By default the latest version returned by get_stics_versions_compat() is used.

Details

The function understand regex as input.

Value

A data.frame with information about variable(s) with columns name, definition, unit, type
is_stics_param

Examples

# Find by variable name (fuzzy search):
SticsRFiles::get_var_info("lai")

# Find by keyword (fuzzy search in variable name and description):
SticsRFiles::get_var_info(keyword = "lai")

# Find for a particular version:
SticsRFiles::get_var_info("lai", stics_version = "V9.0")

---

is_mac

Evaluating if the OS is a Mac OS type

Description

Evaluating if the OS is a Mac OS type

Usage

is_mac()

Value

TRUE/FALSE

Examples

is_mac()

---

is_stics_param

Search if a STICS parameter exist

Description

Tells if one or more parameter names are valid STICS input parameters.

Usage

is_stics_param(param, stics_version = "latest")

Arguments

<table>
<thead>
<tr>
<th>param</th>
<th>A vector of parameter names</th>
</tr>
</thead>
<tbody>
<tr>
<td>stics_version</td>
<td>Name of the STICS version. Optional, can be used to search parameters information relative to a specific STICS version. By default the latest version returned by get_stics_versions_compat() is used.</td>
</tr>
</tbody>
</table>
is_stics_var

Value

A boolean vector: TRUE if the parameter exist, FALSE otherwise

See Also

get_param_info() for interactive use.

Examples

is_stics_param(c("adil", "adilmax", "unknown"))

is_stics_var

Search if a STICS variable exist

Description

Tells if one or more variable names are valid STICS output variables.

Usage

is_stics_var(var, stics_version = "latest")

Arguments

var A vector of variable names

stics_version Name of the STICS version. Optional, can be used to search parameters information relative to a specific STICS version. By default the latest version returned by get_stics_versions_compat() is used.

Value

A boolean vector: TRUE if the variable exist, FALSE otherwise

See Also

get_var_info() for interactive use.

Examples

is_stics_var(c("lai(n)", "masec(n)", "unknown"))
is_unix

**Description**
Evaluating if the OS is a unix like type

**Usage**
is_unix()

**Value**
TRUE/FALSE

**Examples**
is_unix()

---

is_windows

**Description**
Evaluating if the OS is a windows type

**Usage**
is_windows()

**Value**
TRUE/FALSE

**Examples**
is_windows()
read_params_table

**Description**

Getting parameters data from tables files (Excel sheet, csv)

**Usage**

```r
read_params_table(
  file,
  sheet_name = NULL,
  num_na = "NA",
  char_na = "NA",
  file_path = lifecycle::deprecated()
)
```

**Arguments**

- `file` Excel or csv file path (including name of the file)
- `sheet_name` Name of an Excel sheet (useless for csv files)
- `num_na` Replacement value for numerical NA values (default: NA)
- `char_na` Replacement value for character NA values (default: "")
- `file_path` [Deprecated] file_path is no longer supported, use file instead.

**Details**

After data are loaded, numerical and string NA values are replaced respectively with num_na or char_na

**Value**

A tibble of parameters

**Examples**

```r
usm_xl_file <- download_usm_xl(
  file = "inputs_stics_example.xlsx",
  verbose = FALSE
)
read_params_table(usm_xl_file, sheet = "USMs")
usm_csv_file <- download_usm_csv(
  file = "inputs_stics_example_USMs.csv",
  verbose = FALSE,
  stics_version = "V9.2"
)
read_params_table(file = usm_csv_file)
```
set_param_txt  

Set (replace) STICS input file parameters

Description

Replace or set an input parameter from a pre-existing STICS input file.

Usage

```r
set_param_txt(
  workspace,
  param,
  value,
  append = FALSE,
  plant_id = 1,
  variety = NULL,
  value_id = NULL,
  stics_version = "latest",
  dirpath = lifecycle::deprecated(),
  add = lifecycle::deprecated(),
  plant = lifecycle::deprecated(),
  layer = lifecycle::deprecated()
)
```

```r
set_usm_txt(
  file = "new_travail.usm",
  param,
  value,
  append = FALSE,
  filepath = lifecycle::deprecated(),
  add = lifecycle::deprecated()
)
```

```r
set_station_txt(
  file = "station.txt",
  param,
  value,
  append = FALSE,
  filepath = lifecycle::deprecated(),
  add = lifecycle::deprecated()
)
```

```r
set_ini_txt(
  file = "ficini.txt",
  param,
  value,
  append = FALSE,
```
plant_id = 1,
layer = NULL,
stics_version = "latest",
filepath = lifecycle::deprecated(),
add = lifecycle::deprecated()
)

set_general_txt(
    file = "tempopar.sti",
    param,
    value,
    append = FALSE,
    filepath = lifecycle::deprecated(),
    add = lifecycle::deprecated()
)

set_tmp_txt(
    file = "tempoparv6.sti",
    param,
    value,
    append = FALSE,
    filepath = lifecycle::deprecated(),
    add = lifecycle::deprecated()
)

set_plant_txt(
    file = "ficplt1.txt",
    param,
    value,
    append = FALSE,
    variety = NULL,
    filepath = lifecycle::deprecated(),
    add = lifecycle::deprecated()
)

set_tec_txt(
    file = "fictec1.txt",
    param,
    value,
    append = FALSE,
    value_id = NULL,
    filepath = lifecycle::deprecated(),
    add = lifecycle::deprecated()
)

set_soil_txt(
    file = "param.sol",
    param,
value,
layer = NULL,
stics_version = "latest",
filepath = lifecycle::deprecated()
)

Arguments

workspace    Path of the workspace containing the STICS (txt) input files.
param        Vector of parameter names.
value        New parameter value
append       Boolean. Append input to existing file
plant_id     The plant identifier (main crop: 1 ; associated crop: 2). Only used for plant, technical or initialisation parameters (default = 1).
variety      The plant variety to set the parameter value, either the variety name (codevar in the plant file) or the index (variete in the technical file).
value_id     The soil layers id or technical interventions id
stics_version An optional version name as listed in get_stics_versions_compat() return
dirpath      [Deprecated] dirpath is no longer supported, use workspace instead.
add          [Deprecated] add is no longer supported, use append instead.
plant        [Deprecated] plant is no longer supported, use plant_id instead.
layer        [Deprecated] layer is no longer supported, use value_id instead.
file         Path (including name) of the file to modify
filepath     [Deprecated] filepath is no longer supported, use file instead.

Details

The plant parameter can be either equal to 1, 2 for the associated plant in the case of intercrop, or c(1,2) for both Principal and associated plants. get_var_info is a helper function that returns all possible output variables. If the variety is not given and if param is a varietal parameter, the function will modify the value of param for the simulated variety, as given in the technical file.

Value

None

Note

gen_varmod is not used by set_param_txt. To replace the output variables required from STICS, please directly call gen_varmod.
set_param_xml

Examples

# Getting example data path
path <- get_examples_path(file_type = "txt")

# Change the value of durvieF for the current variety:
set_param_txt(workspace = path, param = "durvieF", value = 245)

# Change the value of durvieF for another variety:
set_param_txt(workspace = path, param = "durvieF", 
    variety = "Nefer", value = 178)

# Change the value of soil parameter "cailloux" for all layers
# or a specific one
set_param_txt(workspace = path, param = "cailloux", value = 1)
set_param_txt(workspace = path, param = "cailloux", value_id = 2, value = 2)

# Change the value of parameter "amount" for all water supply interventions
# or a specific one
set_param_txt(workspace = path, param = "amount", value = 50)
set_param_txt(workspace = path, param = "amount", value_id = 2, value = 40)

Description

Setting parameter values for a parameter or a vector of and with a parameters values vector

Usage

set_param_xml(
    file, 
    param, 
    values, 
    save_as = NULL, 
    select = NULL, 
    select_value = NULL, 
    value_id = NULL, 
    overwrite = FALSE, 
    xml_file = lifecycle::deprecated(), 
    out_path = lifecycle::deprecated(), 
    param_name = lifecycle::deprecated(), 
    param_value = lifecycle::deprecated(), 
    value = lifecycle::deprecated(), 
    ... 
)
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>file</code></td>
<td>Path (including name) of the xml file to modify</td>
</tr>
<tr>
<td><code>param</code></td>
<td>Vector of parameter names.</td>
</tr>
<tr>
<td><code>values</code></td>
<td>A vector or a list of parameter(s) values (see details).</td>
</tr>
<tr>
<td><code>save_as</code></td>
<td>Path (including name) of the xml file to generate. Optional, if NULL file is overwritten.</td>
</tr>
<tr>
<td><code>select</code></td>
<td>Node name or attribute name to use for selection (optional, default to no selection)</td>
</tr>
<tr>
<td><code>select_value</code></td>
<td>Vector of values used for select (see examples). Optional, should be provided only if select is provided.</td>
</tr>
<tr>
<td><code>value_id</code></td>
<td>Vector of ids of the parameters values to be retrieved from the parameter values vector</td>
</tr>
<tr>
<td><code>overwrite</code></td>
<td>Logical TRUE for overwriting the output file, FALSE otherwise (default)</td>
</tr>
<tr>
<td><code>xml_file</code></td>
<td>[Deprecated] xml_file is no longer supported, use file instead.</td>
</tr>
<tr>
<td><code>out_path</code></td>
<td>[Deprecated] out_path is no longer supported, use save_as instead.</td>
</tr>
<tr>
<td><code>param_name</code></td>
<td>[Deprecated] param_name is no longer supported, use param instead.</td>
</tr>
<tr>
<td><code>param_value</code></td>
<td>[Deprecated] param_value is no longer supported, use values instead.</td>
</tr>
<tr>
<td><code>value</code></td>
<td>[Deprecated] value is no longer supported, use select_value instead.</td>
</tr>
<tr>
<td>...</td>
<td>Pass further arguments to <code>set_param_value()</code></td>
</tr>
</tbody>
</table>

Details

It is possible to give several values for a parameter by passing a vector of values. For example, for two parameters with two values each: value= list(c(1,2), c(2.3,4.5))

Value

A logical value TRUE for operation success, FALSE otherwise

Examples

```r
ex_path <- get_examples_path(file_type = "xml")

# Soil file
sol_path <- file.path(ex_path, "sols.xml")

# For scalar parameters per soil
# Setting all soils "argi" values to 50
set_param_xml(sol_path, "argi", 50, overwrite = TRUE)
# Getting changed values
get_param_xml(sol_path, "argi")

# Setting a specific value to "argi" for "solcanne" soil
set_param_xml(file = sol_path, param = "argi", values = 56, select = "sol", select_value = "solcanne", overwrite = TRUE)
```
# Getting changed values
# get_param_xml(sol_path, "argi",
# select = "sol", select_value = "solcanne"
#
# Setting a specific values to 2 parameters "argi" and
# "norg" for "solcanne" soil
set_param_xml(sol_path, c("argi", "norg"), list(100, 150),
select = "sol", select_value = "solcanne", overwrite = TRUE
)
# Getting changed values
# get_param_xml(sol_path, c("argi", "norg"),
# select = "sol", select_value = "solcanne"
#
# For vector parameters per soil (5 values, one per soil layer)
set_param_xml(sol_path, c("epc", "HCCF"),
select = "sol",
select_value = c("solcanne", "solbanane"),
values = list(c(20:24, 10:14), c(50:54, 40:44)),
overwrite = TRUE
)
# Getting changed values
# get_param_xml(sol_path, c("epc", "HCCF"),
# select = "sol",
# select_value = c("solcanne", "solbanane")
# )

# For specific values of vector parameters
set_param_xml(sol_path, "HCCF",
select = "sol",
select_value = "solcanne",
values = c(46.8, 48.5, 50.1),
value_id = c(1,3,5),
overwrite = TRUE
)
# Getting changed values
# get_param_xml(sol_path, "HCCF",
# select = "sol",
# select_value = "solcanne",
# value_id = c(1,3,5)
# )

# Crop management file

tec_path <- file.path(ex_path, "file_tec.xml")

# Modifying irrigations parameters
set_param_xml(tec_path, c("julapI_or_sum_upvt", "amount"),
    values = list(200:215, 20:35), overwrite = TRUE
)

# Getting changed values
# get_param_xml(tec_path, c("julapI_or_sum_upvt", "amount"))

---

**upgrade_ini_xml**  

Upgrading \_ini.xml file(s) to a newer version

**Description**

Upgrading \_ini.xml file(s) to a newer version

**Usage**

```r
upgrade_ini_xml(
    file,
    out_dir,
    param_gen_file,
    stics_version = "V9.2",
    target_version = "V10.0",
    check_version = TRUE,
    overwrite = FALSE,
    ...
)
```

**Arguments**

- `file`  
  Path of an initialisation (*\_ini.xml) file or a vector of

- `out_dir`  
  Output directory path of the generated files

- `param_gen_file`  
  Path of the param_gen.xml file corresponding to the file version

- `stics_version`  
  Name of the STICS version (VX.Y format)

- `target_version`  
  Name of the STICS version to upgrade files to (VX.Y format)

- `check_version`  
  Perform version consistency with in stics_version input with the file version
  and finally checking if the upgrade is possible allowed to the target_version. If
  TRUE, param_gen_file is mandatory.

- `overwrite`  
  logical (optional), TRUE for overwriting file if it exists, FALSE otherwise

- `...`  
  Additional input arguments

**Details**

See SticsRFiles::get_stics_versions_compat() for listing versions
upgrade_param_gen_xml

Value
None

Examples

dir_path <- get_examples_path(file_type = "xml", stics_version = "V9.2")

upgrade_ini_xml(
  file = file.path(dir_path,"file_ini.xml"),
  out_dir = tempdir(),
  param_gen_file = file.path(dir_path, "param_gen.xml")
)

Description
Upgrading a param_gen.xml file to a newer version

Usage

upgrade_param_gen_xml(
  file,
  out_dir,
  stics_version = "V9.2",
  target_version = "V10.0",
  check_version = TRUE,
  overwrite = FALSE
)

Arguments

  file          Path of a param_gen.xml file
  out_dir       Output directory path of the generated file
  stics_version Name of the STICS version (VX.Y format)
  target_version Name of the STICS version to upgrade files to (VX.Y format)
  check_version Perform version consistency with in stics_version input with the file version and
                  finally checking if the upgrade is possible allowed to the target_version
  overwrite     logical (optional), TRUE for overwriting file if it exists, FALSE otherwise

Details
See get_stics_versions_compat() for listing versions
upgrade_param_newform_xml

Value
None

Examples

dir_path <- get_examples_path(file_type = "xml", stics_version = "V9.2")

upgrade_param_gen_xml(
  file = file.path(dir_path, "param_gen.xml"),
  out_dir = tempdir()
)

upgrade_param_newform_xml

Upgrading a param_newform.xml file to a newer version

Description
Upgrading a param_newform.xml file to a newer version

Usage

upgrade_param_newform_xml(
  file,
  out_dir,
  param_gen_file,
  stics_version = "V9.2",
  target_version = "V10.0",
  check_version = TRUE,
  overwrite = FALSE
)

Arguments

file Path of a param_newform.xml file
out_dir Output directory path of the generated file
param_gen_file Path of the param_gen.xml file corresponding to the file version
stics_version Name of the STICS version (VX.Y format)
target_version Name of the STICS version to upgrade files to (VX.Y format)
check_version Perform version consistency with in stics_version input with the file version and finally checking if the upgrade is possible allowed to the target_version. If TRUE, param_gen_file is mandatory.
overwrite logical (optional), TRUE for overwriting file if it exists, FALSE otherwise
upgrade_plt_xml

Details
See SticsRFiles::get_stics_versions_compat() for listing versions

Value
None

Examples

dir_path <- get_examples_path(file_type = "xml", stics_version = "V9.2")

upgrade_param_newform_xml(
  file = file.path(dir_path,"param_newform.xml"),
  out_dir = tempdir(),
  param_gen_file = file.path(dir_path, "param_gen.xml")
)

---

upgrade_plt_xml  Upgrading _plt.xml file(s) to a newer version

Description
Upgrading _plt.xml file(s) to a newer version

Usage
upgrade_plt_xml(
  file,
  out_dir,
  param_newform_file,
  param_gen_file,
  stics_version = "V9.2",
  target_version = "V10.0",
  check_version = TRUE,
  overwrite = FALSE,
  ...
)

Arguments
- file: Path of an plant (*.plt.xml) file or a vector of
- out_dir: Output directory path of the generated files
- param_newform_file: Path of the param_newform.xml file corresponding to the file version
- param_gen_file: Path of the param_gen.xml file corresponding to the file version
**upgrade_sols_xml**

**stics_version**  
Name of the STICS version (VX.Y format)

**target_version**  
Name of the STICS version to upgrade files to (VX.Y format)

**check_version**  
Perform version consistency between stics_version and the file version, for finally checking if an upgrade is possible allowed to the target_version. If TRUE, param_gen_file is mandatory.

**overwrite**  
Logical (optional), TRUE for overwriting file if it exists, FALSE otherwise

**...**  
Additional input arguments

**Details**

See get_stics_versions_comp() for listing versions

**Value**

None

**Examples**

```r
dir_path <- get_examples_path(file_type = "xml", stics_version = "V9.2")

upgrade_sols_xml(
  file = file.path(dir_path,"file_plt.xml"),
  out_dir = tempdir(),
  param_newform_file = file.path(dir_path, "param_newform.xml"),
  param_gen_file = file.path(dir_path, "param_gen.xml")
)
```

---

**Description**

Upgrading a sols.xml file to a newer version

**Usage**

```r
upgrade_sols_xml(
  file,
  out_dir,
  param_gen_file,
  stics_version = "V9.2",
  target_version = "V10.0",
  check_version = TRUE,
  overwrite = FALSE
)
```
Arguments

- **file**: Path of a sols.xml file
- **out_dir**: Output directory path of the generated file
- **param_gen_file**: Path of the param_gen.xml file corresponding to the file version
- **stics_version**: Name of the STICS version (VX.Y format)
- **target_version**: Name of the STICS version to upgrade files to (VX.Y format)
- **check_version**: Perform version consistency with in stics_version input with the file version and finally checking if the upgrade is possible allowed to the target_version. If TRUE, param_gen_file is mandatory.
- **overwrite**: logical (optional), TRUE for overwriting file if it exists, FALSE otherwise

Details

See SticsRFiles::get_stics_versions_compat() for listing versions

Value

None

Examples

dir_path <- get_examples_path(file_type = "xml", stics_version = "V9.2")

upgrade_sols_xml(
  file = file.path(dir_path,"sols.xml" ),
  out_dir = tempdir(),
  param_gen_file = file.path(dir_path, "param_gen.xml")
)

upgrade_sta_xml  Upgrading _sta.xml file(s) to a newer version

Description

Upgrading _sta.xml file(s) to a newer version

Usage

upgrade_sta_xml(
  file,
  out_dir,
  param_gen_file,
  stics_version = "V9.2",
  target_version = "V10.0",
  check_version = TRUE,
upgrade_tec_xml

overwrite = FALSE,
...
)

Arguments

file Path of a station (*.sta.xml) file or a vector of
out_dir Output directory path of the generated files
param_gen_file Path of the param_gen.xml file corresponding to the file version
stics_version Name of the STICS version (VX.Y format)
target_version Name of the STICS version to upgrade files to (VX.Y format)
check_version Perform version consistency with in stics_version input with the file version
and finally checking if the upgrade is possible allowed to the target_version. If
TRUE, param_gen_file is mandatory.
overwrite logical (optional), TRUE for overwriting file if it exists, FALSE otherwise
... Additional input arguments

Details

See SticsRFiles::get_stics_versions_compat() for listing versions

Value

None

Examples

dir_path <- get_examples_path(file_type = "xml", stics_version = "V9.2")

upgrade_sta_xml(
  file = file.path(dir_path,"file_sta.xml" ),
  out_dir = tempdir(),
  param_gen_file = file.path(dir_path, "param_gen.xml")
)

upgrade_tec_xml Upgrading _tec.xml file(s) to a newer version

Description

Upgrading _tec.xml file(s) to a newer version
Usage

upgrade_tec_xml(
  file,
  out_dir,
  param_newform_file,
  param_gen_file,
  stics_version = "V9.2",
  target_version = "V10.0",
  check_version = TRUE,
  overwrite = FALSE,
  ...
)

Arguments

file          Path of a crop management (*.tec.xml) file or a vector of
out_dir       Output directory path of the generated files
param_newform_file  Path of the param_newform.xml file corresponding to the file version
param_gen_file  Path of the param_gen.xml file corresponding to the file version
stics_version  Name of the STICS version (VX.Y format)
target_version  Name of the STICS version to upgrade files to (VX.Y format)
check_version  Perform version consistency with in stics_version input with the file version
                and finally checking if the upgrade is possible allowed to the target_version. If
                TRUE, param_gen_file is mandatory.
overwrite      logical (optional), TRUE for overwriting file if it exists, FALSE otherwise
...            Additional input arguments

Details

See get_stics_versions_compat() for listing versions

Value

None

Examples

dir_path <- get_examples_path(file_type = "xml", stics_version = "V9.2")

upgrade_tec_xml(
  file = file.path(dir_path,"file_tec.xml"),
  out_dir = tempdir(),
  param_newform_file = file.path(dir_path, "param_newform.xml"),
  param_gen_file = file.path(dir_path, "param_gen.xml")
)
Description

Upgrading a usms.xml file to a newer version

Usage

upgrade_usms_xml(
  file,
  out_dir,
  param_gen_file,
  obs_dir = NULL,
  stics_version = "V9.2",
  target_version = "V10.0",
  check_version = TRUE,
  overwrite = FALSE
)

Arguments

file Path of a usms.xml file
out_dir Output directory path of the generated file
param_gen_file Path of the param_gen.xml file corresponding to the file version
obs_dir Directory path of the observation data files
stics_version Name of the STICS version (VX.Y format)
target_version Name of the STICS version to upgrade files to (VX.Y format)
check_version Perform version consistency with in stics_version input with the file version and finally checking if the upgrade is possible allowed to the target_version. If TRUE, param_gen_file is mandatory.
overwrite logical (optional), TRUE for overwriting file if it exists, FALSE otherwise

Details

See get_stics_versions_compat() for listing versions

Value

None
Examples

```r
dir_path <- get_examples_path(file_type = "xml", stics_version = "V9.2")

upgrade_usms_xml(
  file = file.path(dir_path,"usms.xml"),
  out_dir = tempdir(),
  param_gen_file = file.path(dir_path, "param_gen.xml")
)
```

Description

Upgrading XML files of a JavaSTICS workspace directory to a newer STICS version format

Usage

```r
upgrade_workspace_xml(
  workspace,
  javastics,
  out_dir,
  stics_version = "V9.2",
  target_version = "V10.0",
  plant = FALSE,
  overwrite = FALSE,
  ...
)
```

Arguments

- `workspace`: Path of a JavaSTICS workspace
- `javastics`: Path of JavaSTICS containing the STICS version corresponding to the version of the files to be converted
- `out_dir`: Output directory of the generated files
- `stics_version`: Name of the STICS version (VX.Y format)
- `target_version`: Name of the STICS version to upgrade files to (VX.Y format)
- `plant`: logical (optional), TRUE for upgrading plant files if a "plant" sub-directory of workspace exists, FALSE otherwise
- `overwrite`: logical (optional), TRUE for overwriting files if they exist, FALSE otherwise
- `...`: Additional input arguments
Details

- See SticsRFiles::get_stics_versions_compat() for listing versions
- If general parameters files exist in workspace, they are also upgraded. In that case, residues parameters values are kept and might not be adapted to the target model version.
- Weather data and observations files are fully copied to out_dir

Value

None

Examples

```r
## Not run:
upgrade_workspace_xml(
  workspace = "/path/to/JavaSTICS/workspace",
  javastics = "/path/to/JavaSTICS/folder",
  out_dir = "/path/to/an/output/directory"
)
## End(Not run)
```

---

Description

This method ensure keeping the cropr_simulation attribute when subsetting a cropr_simulation list.

Usage

```r
## S3 method for class 'cropr_simulation'

x[...]
```

Arguments

- `x` A cropr_simulation list
- `...` An index

Value

A subset of a cropr_simulation, keeping its attribute

Examples

```r
path <- file.path(get_examples_path("sti"), "workspace1")
sim <- SticsRFiles::get_sim(workspace = path)
# sim returns a `cropr_simulation` list
```
Index

compute_date_from_day, 4
compute_day_from_date, 5
convert_xml2txt, 6

date, 5
download_data, 7
download_usm_csv, 8
download_usm_xl, 9

force_param_values, 10

gen_ini_xml, 9, 11
gen_obs, 12
gen_sols_xml, 9, 14
gen_sta_xml, 9, 15
gen_tec_xml, 9, 17
gen_usms_xml, 8, 9, 18
gen_usms_xml2txt, 20
gen_varmod, 21
get_climate_txt, 23
get_cultivars_list, 24
get_cultivars_param, 24
get_examples_path, 25
get_general_txt (get_param_txt), 29
gen_ini_txt (get_param_txt), 29
get_lai_forcing, 26
get_obs, 26
get_param_info, 28
get_param_txt, 29
get_param_xml, 33
get_plant_txt (get_param_txt), 29
get_plants_nb, 35
get_report_results, 36
get_sim, 37
get_soil_txt (get_param_txt), 29
get_soils_list, 38
get_station_txt (get_param_txt), 29
get_stics_versions_compat, 39
get_tec_txt (get_param_txt), 29
get_tmp_txt (get_param_txt), 29
get_usm_txt (get_param_txt), 29
get_usms_files, 40
get_usms_list, 41
get_var_info, 13, 43, 50
get_varmod, 42
is_mac, 44
is_stics_param, 44
is_stics_var, 45
is_unix, 46
is_windows, 46
read_params_table, 47
regex, 29, 43
set_general_txt (set_param_txt), 48
set_ini_txt (set_param_txt), 48
set_param_txt, 48
set_param_xml, 51
set_plant_txt (set_param_txt), 48
set_soil_txt (set_param_txt), 48
set_station_txt (set_param_txt), 48
set_tec_txt (set_param_txt), 48
set_tmp_txt (set_param_txt), 48
set_usm_txt (set_param_txt), 48
upgrade_ini_xml, 54
upgrade_param_gen_xml, 55
upgrade_param_newform_xml, 56
upgrade_plt_xml, 57
upgrade_sols_xml, 58
upgrade_sta_xml, 59
upgrade_tec_xml, 60
upgrade_usms_xml, 62
upgrade_workspace_xml, 63