Package ‘SticsRFiles’

July 11, 2023

Title  Read and Modify ‘STICS’ Input/Output Files
Version  1.1.3
Date  2023-07-10
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 (>= 3.6.0)
Imports  cli, crayon, curl, data.table, dplyr (>= 1.0.0), lifecycle, lubridate, methods, rlang, rstudioapi, stringr, tibble, tidyr, 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  7.2.1
Collate  'add_node_to_doc.R' 'add_stics_nodes.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'
     'check_usms_files.R' 'col_names_to_var.R' 'compute_doy_cumul.R'
'convert_xml2txt.R' 'convert_xml2txt_int.R' 'download_data.R'
'download_usm_xl.R' 'exist_param_xml.R'
'exists_javastics_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_ini_doc.R' 'gen_ini_xml.R' 'gen_new_travail.R' 'gen_obs.R'
'gen_paramst.R' 'gen_sols_xml.R' 'gen_sta_doc.R'
'gen_sta_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_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_obs_txt.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.R'
'get_param_names.R' 'get_param_names_xml.R'
'get_param_number.R' 'get_param_type.R'
'get_param_value.R' 'get_params_xml.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_javastics_pref.R' 'is_os_name.R' 'is_stics_doc.R'
'is_stics_xml.R' 'javastics_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'
'set_file_executable.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' 'istics_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>),
Thomas Robine [ctb],
Amine Barkouei [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 2023-07-11 15:30:12 UTC

R topics documented:

convert_xml2txt ........................................ 4
download_data .......................................... 5
download_usm_csv ....................................... 6
download_usm_xl ......................................... 7
force_param_values .................................... 8
gen_ini_xml ............................................. 9
gen_obs .................................................. 11
gen_sols_xml ........................................... 12
gen_sta_xml ............................................ 13
gen_tec_xml ............................................ 15
gen_usms_xml ........................................... 16
gen_usms_xml2txt ...................................... 18
gen_varmod ............................................. 20
get_climate_txt ........................................ 21
get_examples_path ..................................... 22
get_lai_forcing ........................................ 23
get_obs ................................................. 23
get_param_info ......................................... 25
get_param_txt .......................................... 27
get_param_xml .......................................... 30
get_plants_nb .......................................... 32
get_report_results .................................... 33
get_sim .................................................. 34
get_soils_list .......................................... 35
get_stics_versions_compat ............................ 36
get_usms_files ......................................... 37
get_usms_list .......................................... 38
get_varmod ............................................. 39
get_var_info ........................................... 40
is_mac ................................................... 41
is_stics_var ............................................ 41
**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,
  javastics,
  plant_id = 1,
  out_dir = NULL,
  save_as = NULL,
  xml_file = lifecycle::deprecated(),
  java_dir = lifecycle::deprecated(),
  plt_num = lifecycle::deprecated(),
  out_file = lifecycle::deprecated()
)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
<td>Path (including name) of the xml file to convert</td>
</tr>
<tr>
<td>javastics</td>
<td>Path of JavaSTICS</td>
</tr>
<tr>
<td>plant_id</td>
<td>The plant identifier (main crop: 1 ; associated crop: 2)</td>
</tr>
<tr>
<td>out_dir</td>
<td>Path of the directory where to generate the file. Optional, set to the path of the input xml file by default</td>
</tr>
</tbody>
</table>
download_data

save_as Name of the output file (optional, default: fixed name for STICS)
xml_file [Deprecated] xml_file is no longer supported, use file instead.
java_dir [Deprecated] java_dir is no longer supported, use javastics 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.

Value
None

Examples

## Not run:
xml_path <- "/path/to/corn_plt.xml"
javastics_path <- "/path/to/JavaSTICS/folder"
convert_xml2txt(file = xml_path, javastics = javastics_path)

## End(Not run)

download_data Download example USMs

Description

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

Usage

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

Arguments

out_dir Path of the directory where to download the data
example_dirs List of use case directories names (optional)
stics_version Name of the STICS version. Optional, by default the latest version returned by get_stics_versions_compat() is used.
dir [Deprecated] dir is no longer supported, use out_dir instead.
version_name [Deprecated] file_path is no longer supported, use file instead.
**Value**

The path to the folder where data have been downloaded

**Examples**

```r
# Getting all data
download_data()

download_data(example_dirs = "study_case_1")

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

### download_usm_csv

**Download a CSV usms data file example into a directory**

**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.
**download_usm_xl**

**Value**

A vector of copied files path.

**Examples**

```
download_usm_csv(out_dir = tempdir())
```

---

**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
force_param_values

Value
A vector of copied files path.

Examples

download_usm_x1()

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
force_param_values(
  workspace, 
  values, 
  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

## 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)
**gen_ini_xml**

*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**

an invisible `xml_document` object or a list of

**Examples**

```r
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_table = ini_param_df
)
```
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:

<table>
<thead>
<tr>
<th>usm_name</th>
<th>ian</th>
<th>mo</th>
<th>jo</th>
<th>jul</th>
<th>densite</th>
<th>lai(n)</th>
<th>masec(n)</th>
<th>azomes</th>
</tr>
</thead>
<tbody>
<tr>
<td>USM_2017_T1_CI</td>
<td>2017</td>
<td>9</td>
<td>6</td>
<td>249</td>
<td>NA</td>
<td>NA</td>
<td>0.31</td>
<td>27.07395</td>
</tr>
<tr>
<td>USM_2017_T1_CI</td>
<td>2017</td>
<td>9</td>
<td>20</td>
<td>263</td>
<td>NA</td>
<td>NA</td>
<td>0.60</td>
<td>27.90000</td>
</tr>
<tr>
<td>USM_2018_T1</td>
<td>2017</td>
<td>10</td>
<td>20</td>
<td>293</td>
<td>NA</td>
<td>0.1</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>USM_2018_T1</td>
<td>2018</td>
<td>5</td>
<td>15</td>
<td>482</td>
<td>NA</td>
<td>1.2</td>
<td>NA</td>
<td>NA</td>
</tr>
</tbody>
</table>

- `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

```r
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`, `sols_param`, `sols_out_file`, `sols_nb`: [Deprecated] These arguments are no longer supported, use `param_df` 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>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>20.35000</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>17.00000</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>
<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>Vill09</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>

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

an invisible xml_document object

Examples

```r
tax_path <- download_usm_xl(file = "inputs_stics_example.xlsx")
sols_param_df <- read_params_table(file = tax_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(),
  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 function will use a standard template depending on the STICS version.
- `out_dir`: Path of the directory where to generate the file(s).
- `tics_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.
- `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 `tics_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

an invisible xml_document object or a list of
Examples

```r
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)
```

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>jules</th>
<th>coderes</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>
<tr>
<td>AGA_ARB_13_IRR_220-0-0_37K_ARB13_C_tec.xml</td>
<td>82</td>
<td>1</td>
</tr>
<tr>
<td>AGA_ARB_13_SEC_220-0-0_37K_ARB13_C_tec.xml</td>
<td>82</td>
<td>1</td>
</tr>
<tr>
<td>FRA_ARB_11_SEC_220-0-0_38K_E_tec.xml</td>
<td>70</td>
<td>1</td>
</tr>
<tr>
<td>MAG_ARB_09_SEC_220-0-0_38K_E_tec.xml</td>
<td>81</td>
<td>1</td>
</tr>
<tr>
<td>MAG_ARV_12_IRR_220-0-0_36K_ARV12_C_tec.xml</td>
<td>100</td>
<td>1</td>
</tr>
<tr>
<td>MAG_ARV_12_SEC_220-0-0_36K_ARV12_C_tec.xml</td>
<td>100</td>
<td>1</td>
</tr>
<tr>
<td>FRA_ARB_12_SEC_220-0-0_31K_ARB12_C_tec.xml</td>
<td>92</td>
<td>1</td>
</tr>
<tr>
<td>FRA_ARB_13_SEC_220-0-0_37K_ARB13_C_tec.xml</td>
<td>82</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**

an invisible xml_document object or a list of

**Examples**

```r
xl_path <- download_usm_xl(file = "inputs_stics_example.xlsx")
techn_param_df <- read_params_table(file = xl_path, sheet_name = "Tec")
gen_tec_xml(out_dir = tempdir(), param_df = tech_param_df)
```

---

**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(),
)```

---

**gen_usms_xml**  
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(),
)```
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>nomso1</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_ARB_12_IRR_220-0-0_36K_ARV12_C</td>
<td>199</td>
<td>263</td>
<td>LF1</td>
</tr>
<tr>
<td>MAG_ARB_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_ARV12_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_xl(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)
```

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,
    workspace = NULL,
    out_dir = NULL,
    usm = c(),
    verbose = TRUE,
    dir_per_usm_flag = TRUE,
    check = TRUE,
    java_cmd = "java",
    javastics_path = lifecycle::deprecated(),
    workspace_path = lifecycle::deprecated(),
    target_path = lifecycle::deprecated(),
    usms_list = lifecycle::deprecated(),
    check_files = lifecycle::deprecated()
)
```

Arguments

- **javastics** 
  - Path of JavaSTICS.
- **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_path/usms.xml file will be generated.

**verbose**  Logical value for displaying information while running

**dir_per_usm_flag**  logical, TRUE if one want to create one directory per USM, FALSE if USM files are generated in the target_path (only useful for usms_list of size one)

**check**  Logical, TRUE to check if usms files exist, FALSE otherwise

**java_cmd**  The java virtual machine command name or executable path

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

**workspace_path**  [Deprecated] workspace_path is no longer supported, use workspace instead.

**target_path**  [Deprecated] target_path is no longer supported, use out_dir instead.

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

**check_files**  [Deprecated] check_files is no longer supported, use check instead.

**Value**

A list with named elements: usms_path : created directories paths (for storing STICS input files), files : generated files list (in JavaSTICS workspace origin), copy_status : logical value vector, indicating if all files have been generated for each usm obs_copy_status : logical value vector, indicating if observation files have been successfully copied in usms directories

**Examples**

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

# For all usms
gen_usms_xml2txt(javastics, workspace)

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

## End(Not run)
Generating a var.mod type file

Description
Generating a daily variable list file from variables names

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).
- `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_examples_path

Description

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

Usage

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

generate_example(file_type = "csv")
generate_example(file_type = c("csv", "sti"))
generate_example(file_type = "csv", stics_version = "V8.5")
get_lai_forcing  

**Description**

Is LAI forced for usms in usms.xml

**Usage**

```r
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**

```r
# 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)
```
get_param_info

stics_version = "latest",
file_path = lifecycle::deprecated(),
parameter = lifecycle::deprecated(),
version = lifecycle::deprecated()
)

Arguments

param Vector of parameter names (or partial names). Optional, if not provided, the
function returns information for all parameters

file Vector of xml file paths. Optional, if not provided, the function searches inform-

ation in all standard XML files.

formalism Optional, formalism name or partial name, or a vector of

keyword Optional, strings or a vector of to be used for searching in parameters data (i.e.: parameters names, formalisms description, file names or part to which parame-
ters are attached to)

stics_version Name of the STICS version. Optional, can be used to search parameters in-

formation relative to a specific STICS version. By default the latest version
returned by get_stics_versions_compat() is used.

file_path [Deprecated] file_path is no longer supported, use file instead.

parameter [Deprecated] parameter is no longer supported, use param instead.

version [Deprecated] version is no longer supported, use stics_version instead.

Details

parameter and formalism may be both set or only one of them, but none of them can be if keyword argument is used.

Value

A data.frame containing parameters names, their file name origin, their bounds and the formalism they belong to. The data.frame has the model version as attribute.

Examples

## Not run:
get_param_info(param = "albedo")
get_param_info(param = "albedo", file = "/path/to/file.xml")

get_param_info(param = "albedo", formalism = "special")

get_param_info(param = "albedo", stics_version = "V9.0")

get_param_info(param = c("alb", "lat"))

get_param_info(keyword = "tec")

# Get the model version afterward:
get_param_txt

params <- get_param_info(param = "albedo")
attr(params, "version")

## End(Not run)

---

**get_param_txt**  
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

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

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

```r
get_general_txt(file = "tempopar.sti", filepath = lifecycle::deprecated())
```

```r
get_tmp_txt(file = "tempoparv6.sti", filepath = lifecycle::deprecated())
```

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

```r
get_tec_txt(  
    file = "fictec1.txt",  
    stics_version = "latest",  
    ...  
)
```
get_param_txt

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(file = "new_travail.usm", 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.
variety Integer. The plant variety to get the parameter from.
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:

- **ini**: Initialization parameters
- **general**: General parameters
- **tec**: Technical parameters
- **plant**: Plant parameters
- **soil**: Soil parameters
- **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

## 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**

```r
get_param_xml(
  file,
  param = NULL,
  select = NULL,
  select_value = NULL,
  xml_file = lifecycle::deprecated(),
  param_name = lifecycle::deprecated(),
  value = lifecycle::deprecated(),
  ...
)
```
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.
- **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, "argi")
get_param_xml(file, c("argi", "norg"))

# For one soil selection
get_param_xml(file, "argi", select = "sol", select_value = "solcanne")

# For soils and parameters vectors
# scalar parameters per soil
get_param_xml(file, c("argi", "norg"),
  select = "sol", select_value = c("solcanne", "solbanane")
)

# vector parameters per soil (5 values, one per soil layer)
get_param_xml(file, c("epc", "HCCF"),
  select = "sol", select_value = c("solcanne", "solbanane")
)

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

get_param_xml(file)

# Getting parameters for irrigation (date and quantity)
get_param_xml(file, c("julapI_or_sum_upvt", "amount"))
```
# Getting all parameters for a given formalism: "irrigation"
get_param_xml(file, select = "formalisme", select_value = "irrigation")

## 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**: Usm(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

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.

ejavastics_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

```r
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

```r
get_soils_list(
  file,
  soil = NULL,
  file_path = lifecycle::deprecated(),
  name = lifecycle::deprecated()
)
```
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
**get_usms_files**

**Value**

A named list with the STICS versions compatible with this package ($\text{versions\_list}$), and the latest version in use ($\text{latest\_version}$) or an existing version selected using $\text{version\_index}$.

**Examples**

```r
# 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**  
*Getting existing xml files path list per usm from an usms.xml file*

**Description**

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

**Usage**

```r
get_usms_files(
  workspace,
  usms_list = NULL,
  usms_file = "usms.xml",
  file_type = NULL,
  javastics = NULL,
  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)
- **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.
get_usms_list

Details

The possible values of file types are: "fplt", "finit", "fclim1", "fclim2", "fstation" and "ftec"

Value

A named list with existing files path in each usm element

See Also

See get_soils_list() to get all soils in a usm file, and get_usms_list() to get the list of usms.

Examples

## Not run:

```r
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)
```

---

get_usms_list

Getting usms names list for an usms.xml file

Description

Extracting a usm names list from an usms.xml file

Usage

```r
get_usms_list(
  file,
  usm = NULL,
  usm_path = lifecycle::deprecated(),
  name = lifecycle::deprecated()
)
```
**get_varmod**

Get desired STICS outputs

**Description**

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

**Usage**

```r
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")

**Value**

The variables that will be returned by STICS

**See Also**

`gen_varmod`

**Examples**

```r
get_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
get_var_info(
  var = NULL,
  keyword = NULL,
  stics_version = "latest",
  version = lifecycle::deprecated()
)

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>var</td>
<td>Vector of variable names (or partial names). Optional, if not provided, the function returns information for all variables.</td>
</tr>
<tr>
<td>keyword</td>
<td>Search by keyword instead of variable name (search in the name and description field)</td>
</tr>
<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>
<tr>
<td>version</td>
<td>[Deprecated] version is no longer supported, use stics_version instead.</td>
</tr>
</tbody>
</table>

Details
The function understand regex as input.

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

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_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", version = lifecycle::deprecated())

**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.
- **version**: [Deprecated] version is no longer supported, use stics_version instead.

**Value**
A boolean vector: TRUE if the variable exist, FALSE otherwise
See Also

get_var_info() for interactive use.

Examples

```r
is_srics_var(c("lai(n)", "masec(n)", "unknown"))
```

---

**is_unix**

* Evaluating if the OS is a unix like type

**Description**

Evaluating if the OS is a unix like type

**Usage**

```r
is_unix()
```

**Value**

TRUE/FALSE

**Examples**

```r
is_unix()
```

---

**is_windows**

* Evaluating if the OS is a windows type

**Description**

Evaluating if the OS is a windows type

**Usage**

```r
is_windows()
```

**Value**

TRUE/FALSE

**Examples**

```r
is_windows()
```


Description

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

Usage

```r
read_params_table(
  file,
  sheet_name = NULL,
  num_na = "NA",
  char_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 (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 = NULL, 
  variety = NULL, 
  layer = NULL, 
  stics_version = "latest", 
  dirpath = lifecycle::deprecated(), 
  add = lifecycle::deprecated(),
  plant = 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,
)```
set_param_txt

layer = NULL,
stats_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,
  filepath = lifecycle::deprecated(),
  add = lifecycle::deprecated()
)

set_soil_txt(
  file = "param.sol",
  param,
  value,
  layer = NULL,
```r
set_param_txt

  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.
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).
layer      The soil layer if any (only concerns soil-related parameters)
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.
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.

Examples

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

# Replace the interrow distance parameter to 0.01:
set_param_txt(workspace = path, param = "interrang", value = 0.01)
```
set_param_xml

# 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 infil for a given layer:
set_param_txt(workspace = path, param = "infil", layer = 2, value = 60)

# If the parameter is found in several files, use the set_*
# functions directly, e.g. cailloux is found in the general file
# ("codecailloux") and the soil file. If we want to change its value
# in the soil file, we use set_soil_txt():
set_soil_txt(file = file.path(path, "param.sol"),
param = "cailloux", layer = 2, value = 1)

---

set_param_xml  Setting parameter values into xml files

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,
  overwrite = FALSE,
  xml_file = lifecycle::deprecated(),
  out_path = lifecycle::deprecated(),
  param_name = lifecycle::deprecated(),
  param_value = lifecycle::deprecated(),
  value = lifecycle::deprecated(),
  ...
)

Arguments

file Path (including name) of the xml file to modify
param Vector of parameter names.
values: A vector or a list of parameter(s) values (see details).

save_as: Path (including name) of the xml file to generate. Optional, if NULL file is overwritten.

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.

overwrite: Logical TRUE for overwriting the output file, FALSE otherwise (default)

xml_file: [Deprecated] xml_file is no longer supported, use file instead.

out_path: [Deprecated] out_path is no longer supported, use save_as instead.

param_name: [Deprecated] param_name is no longer supported, use param instead.

param_value: [Deprecated] param_value is no longer supported, use values instead.

value: [Deprecated] value is no longer supported, use select_value instead.

... Pass further arguments to set_param_value().

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)

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)

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
)

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"),
param_value = list(c(20:24, 10:14), c(50:54, 40:44)), overwrite = TRUE
)

get_param_xml(sol_path, c("epc", "HCCF"),
select = "sol",
select_value = c("solcanne", "solbanane")
)

# 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"),
param_value = list(200:215, 20:35), overwrite = TRUE
)

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**

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

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")
)
upgrade_param_gen_xml  Upgrading a param_gen.xml file to a newer version

Description
Upgrading a param_gen.xml file to a newer version

Usage
up 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

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()
)
Description

Upgrading a param_newform.xml file to a newer version

Usage

```r
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

Details

See SticsRFiles::get_stics_versions_compat() for listing versions

Value

None
Examples

```r
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")
)
```

**Description**

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

**Usage**

```r
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
- **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
upgrade_sols_xml

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_plt_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")
)
```

upgrade_sols_xml Upgrading a sols.xml file to a newer version

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

```r
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")
)
```

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

Usage

```r
upgrade_sta_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 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

```r
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")
)
upgrade_usms_xml  Upgrading a usms.xml file to a newer version

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
- **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)
```

## S3 method for class 'cropr_simulation'

```r
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

[.cropr_simulation, 60
convert_xml2txt, 4
download_data, 5
download_usm_csv, 6
download_usm_xml, 7
force_param_values, 8
gen_ini_xml, 7, 9
gen_obs, 11
gen_sols_xml, 7, 12
gen_sta_xml, 7, 13
gen_tec_xml, 7, 15
gen_usms_xml, 6, 7, 16
gen_usms_xml2txt, 18
gen_var_info, 11, 40, 46
get_climate_txt, 21
get_examples_path, 22
get_general_txt (get_param_txt), 27
gen_ini_txt (get_param_txt), 27
get_lai_forcing, 23
gen_obs, 23
get_param_info, 25
get_param_txt, 27
get_param_xml, 30
get_plant_txt (get_param_txt), 27
get_plants_nb, 32
get_report_results, 33
get_sim, 34
get_soil_txt (get_param_txt), 27
get_soils_list, 35
get_station_txt (get_param_txt), 27
get_stics_versions_compat, 36
get_tec_txt (get_param_txt), 27
get_tmp_txt (get_param_txt), 27
get_usm_txt (get_param_txt), 27
get_usms_files, 37
get_usms_list, 38
get_var_info, 11, 40, 46
get_varmod, 39
is_mac, 41
is_stics_var, 41
is_unix, 42
is_windows, 42
read_params_table, 43
regex, 40
set_general_txt (set_param_txt), 44
set_ini_txt (set_param_txt), 44
set_param_txt, 44
set_param_xml, 47
set_plant_txt (set_param_txt), 44
set_soil_txt (set_param_txt), 44
set_station_txt (set_param_txt), 44
set_tec_txt (set_param_txt), 44
set_tmp_txt (set_param_txt), 44
set_usm_txt (set_param_txt), 44
upgrade_ini_xml, 49
upgrade_param_gen_xml, 51
upgrade_param_newform_xml, 52
upgrade_plt_xml, 53
upgrade_sols_xml, 54
upgrade_stas_xml, 55
upgrade_tec_xml, 56
upgrade_usms_xml, 58
upgrade_workspace_xml, 59