traDSSAT Documentation

Release 0.1.5

Julien Malard,

Jun 16, 2020

Contents

TraDSSAT aims to reinvent the wheel, once and for all, for DSSAT file management. Whether for preparing input files for automated DSSAT runs or for analysing outputs, traDSSAT's simple Python interface and modular structure will (we hope!) dramatically simplify your code.

CHAPTER 1

Contents

1.1 Quick start

1.1.1 Installation

traDSSAT requires `numpy` and `chardet` to run. You can install it from pip with: `pip install tradssat`

You can also install the most cutting-edge version directly from GitHub with: `pip install git+git://github.com/julienmalard/tradssat.git@master`

You will need a local installation of DSSAT to use the high-level interface's automatic file managers (since these will need to find soil, weather, and other files in the DSSAT installation directory). TraDSSAT should find your DSSAT installation by itself, but if it needs help you can specify it with:

```
from tradssat import set_dssat_dir
set_dssat_dir('C:/My/odd/path/to/DSSAT47')
```

1.1.2 A quick example

```
from tradssat import SoilFile, WTHFile, ExpFile, set_dssat_dir
from tradssat import GeneticMgr, CULFile, ECOFile

# Read, edit and write soil files
soil = SoilFile('path/to/my/file.SOL')

# Read and write weather files as well
wth = WTHFile('path/to/my/WTHR0001.WTH')

# ...and experiment files!
exp = ExpFile('path/to/my/experiment.EXP')
```

(continues on next page)

(continued from previous page)

```
# Access genetic coefficients by cultivar file or ecotype file
cul = CULFile('path/to/my/MZIXM.CUL')
eco = ECOFile('path/to/my/MZIXM.ECO')

cul.get_value('P1')  # returns array of all varieties' P1
eco.get_value('TOPT')  # returns array of all ecotypes' TOPT

# ...or automagically!
set_dssat_dir('C:/DSSAT47')
gen = GeneticMgr(crop='MZIXM', cult='PC0001')
gen.get_value('P1')  # Returns P1 for MZIXM cultivar PC0001
gen.get_value('TOPT')  # Returns ecotype variable TOPT for cultivar PC001
```

1.2 Input files

TraDSSAT has specific classes to read and edit DSSAT input files. Each class defines its own allowed variables and variable information. All files have the same general structure:

1. Section (named; header line starts with * or $ in DSSAT files)
2. Section header variables (optional)
3. Subsection (numbered; header line starts with @ followed by variable names in DSSAT files)
4. Subsection variables

Specific classes used to read DSSAT input files are:

- *SoilFile* (.SOL)
- *WTHFile* (.WTH, .WTG)
- *ExpFile* (.ccX)
- *CULFile* (.CUL)
- *ECOFile* (.ECO)

As all input files inherit from *InpFile*, the same interface to reading, editing and writing holds for all DSSAT input files.

1.2.1 Reading files

Input files are instantiated with the path to the file to be read.

```
from tradssat import WTHFile
wth = WTHFile('path/to/my/WTHR0001.WTH')
```

Values of all variables can then be read directly.

```
# Get solar radiation data time series
wth.get_value('SRAD')

# Conditions can also be set

# Only get data from 2 Jan, 2013.
```

(continues on next page)

(continued from previous page)

```
wth.get_value('SRAD', cond={'DATE': '13002'})

# Only get data from dry days
wth.get_value('SRAD', cond={'RAIN': 0})
```

Getting a list of all allowed variable names for the file type is also easy.

```
wth.variables()
```

1.2.2 Editing files

Variable values can be changed, either for the whole file or for specific sections and/or subsections, as well as by condition.

```
from tradssat import SoilFile
soil = SoilFile('path/to/my/file.SOL')

# Set all soils' SALB to 0.15
soil.set_value('SALB', 0.15)

# Only set soil IB00000002's SALB to 0.2
soil.set_value('SALB', 0.20, sect='IB00000002')

# Increase clay, but only for the first 5 cm of soil IB00000002
soil.set_value('SLCL', 0.50, sect='IB00000002', cond={'SLB': 5})
```

You can also add rows to specific subsections of a file, or remove existing rows. Subsection variables not included in `vals` will be set to missing (usually -99).

```
# Add new soil layer
soil.add_row(
    sect='IB00000002', subsect=2, vals={'SLB': 180, 'SLLL': 0.260}
)
```

You can save the data to json format, or else write a DSSAT-format file back to disk.

```
# Convert to dict...
json_d = soil.to_dict()

# ...or save to disk
wth.write('path/to/my/new/SOIL.SOL')
```

1.3 Output files

Output files can only be read from, not written to. The following classes are available:

- *PlantGroOut* (PlantGro.OUT)
- *SoilNiOut* (SoilNi.OUT)
- *SoilTempOut* (SoilTemp.OUT)
- *SummaryOut* (Summary.OUT)

These classes can all be used to read the corresponding output file. Note that it is generally more straightforward and pleasant to use a *DSSATResults* object instead of accessing individual output files directly.

```
from tradssat.out import PlantGroOut

plant = PlantGroOut('path/to/my/PlantGro.OUT')

# Get Leaf Area Index time series
plant.get_value('LAID')
```

1.4 High-level interface

The high-level interface simplifies many tasks while working with DSSAT input and output files.

1.4.1 Input files

Each of these managers creates a more friendly interface to DSSAT files than that of the base input file reader.

Experiment file manager

DSSATRun allows you to read, edit and write experiments, with automatic access to referenced external soil, genetic and weather files. TraDSSAT will also automagically manage the link between DSSAT treatments and associated factor levels (if you don't know what this means, now would be a great time to stop reading this and read the DSSAT docs instead).

```
from tradssat import DSSATRun

run = DSSATRun('path/to/my/experiment.EXP')

# Get cultivar for treatment 1
run.get_trt_val('INGENO', trt=1)

# Change level of treatment factor
run.set_trt_factor_level(trt=1, factor='CULTIVARS', level=2)

# Change value of a factor level (in this case cultivar type)
run.set_factor_level_val('INGENO', 'IB0067', level=1)

# Access soil variable SLLL for treatment 2
run.get_trt_val('SLLL', trt=2)
```

Genetic file manager

DSSAT's crop modules generally split coefficients between cultivar, ecotype and species files. TraDSSAT provides a *GeneticMgr* class to automagically manage all genetic coefficients for a particular crop and cultivar type.

```
from tradssat import GeneticMgr

gen = GeneticMgr(crop='MZIXM', cult='PC0001')

# Returns P1 for MZIXM cultivar PC0001
```

(continues on next page)

(continued from previous page)

```
gen.get_value('P1')

# Returns ecotype variable TOPT for cultivar PC001
gen.get_value('TOPT')
```

Note: It is currently not possible to access species coefficients with traDSSAT, because these are in practice model constants and should not be written or changed (and, if they were, would also by default affect **all future** DSSAT simulations run on your DSSAT installation). More practically, they also come in a variety of formats and would be a pain to parse.

Soil file manager

Soil files can be hard to manage, since they can contain data for many different soils in the same file. With *SoilMgr*, simply pass the soil code and traDSSAT will find the correct file and file section.

```
from tradssat.mgrs import SoilMgr

soil_mgr = SoilMgr('IB00000005')
soil_mgr.get_value('SLU1')
```

Weather file manager

Don't feel like finding your weather file yourself? Just give the station code to *WeatherFileMgr* and let it find it for you.

```
from tradssat.mgrs import WeatherFileMgr

wth_mgr = WeatherFileMgr('ACNM')
wth_mgr.get_value('RAIN')
```

1.4.2 Output files

You can access output from a run using a *DSSATResults* object instantiated with the output directory.

```
from tradssat import DSSATResults

out = DSSATResults('path/to/my/output/dir')

# Get FWAD results for treatment 1 (as a time series)
out.get_value('FWAD', trt=1)

# Get results at specific time

 # Get result at 13 days after start
out.get_value('FWAD', trt=1, t=13, at='DAS')

# Get result at 13 days after planting
out.get_value('FWAD', trt=1, t=13, at='DAP')
```

(continues on next page)

(continued from previous page)

```
# Get result at 123th day of year 1989
out.get_value('FWAD', trt=1, t='1989 123', at='DOY')
```

1.5 API Reference

TraDSSAT allows users to access files directly or to manage them through the high-level interface.

1.5.1 File managers

All DSSAT file reading and editing happens through subclasses of these parent classes.

Input files

All input files inherit from *InpFile*.

class tradssat.**SoilFile**(*file*)
File reader for soil (.SOL) DSSAT files.

class tradssat.**WTHFile**(*file*)
Reader for DSSAT weather (.WTH and .WTG) files.

class tradssat.**ExpFile**(*file*)
File reader for all DSSAT (.ccX) experiment files.

class tradssat.**CULFile**(*file*)
Cultivar (.CUL) input file reader.

class tradssat.**ECOFile**(*file*)
Ecotype (.ECO) input file reader.

Output files

Most output files inherit from *OutFile*.

class tradssat.out.**PlantGroOut**(*folder*)
File reader for DSSAT plant growth (PLANTGRO.OUT) output files.

class tradssat.out.**SoilNiOut**(*folder*)
Reader for DSSAT soil nitrogen (SOILNI.OUT) files.

class tradssat.out.**SoilTempOut**(*folder*)
Reader for DSSAT soil temperature (SOILTEMP.OUT) output files.

class tradssat.out.**SummaryOut**(*folder*)
Reader for DSSAT run summary (SUMMARY.OUT) output files.

1.5.2 High-level API

The high-level API simplifies work with more complex DSSAT files, espcially ones that reference each other.

Run input managers

class tradssat.mgrs.**DSSATRun**(*file*, *model=None*)
General manager for DSSAT run input files.

add_factor_level(*factor*, *vals*)
Adds a factor level to the experiment.

Parameters

- **factor** (*str*) – Factor code or name.
- **vals** (*dict*) – Dictionnary of variable values for that factor level's variables. Missing variables will be assigned the corresponding missing code (usually -99).

add_treatment(*name*, *ops=None*, *factors=None*)
Adds a treatment level to the experiment.

Parameters

- **name** (*str*) – Name of the new treatment.
- **ops** (*dict*) – Dictionnary of values for the *R*, *O*, and *C* treatment level options (optional).
- **factors** (*dict*) – Dictionnary of values for the treatment's factor levels (optional). Missing factors will be assigned level 0.

get_factor_level_val(*var*, *level*)
Obtain the variable value for a specific factor level.

Parameters

- **var** (*str*) – The variable of interest.
- **level** (*int*) – The factor level corresponding to the specified variable.

Returns The variable value at the factor level.

Return type np.ndarray

get_general_val(*var*)
Obtain a variable value from the *GENERAL* section of the EXP file.

Parameters **var** (*str*) – Variable name

Returns Variable value.

Return type np.ndarray

get_trt_factor_level(*trt*, *factor*)
Obtain the factor level for a specific treatment.

Parameters

- **trt** (*str | int*) – Treatment name or number.
- **factor** (*str*) – Factor name or code.

Returns The factor level corresponding to the given treatment.

Return type int

get_trt_name(*n*)
Returns the treatment name corresponding to a treatment number.

Parameters **n** (*int*) – The treatment number.

Returns The treatment name.

Return type str

get_trt_num(*trt*)
Returns the treatment number corresponding to a specified treatment name.

Parameters **trt** (*str*) – The treatment name.

Returns The corresponding treatment number.

Return type int

get_trt_val(*var*, *trt*)
Returns the value of a treatment's variable.

Parameters

- **var** (*str*) – The variable of interest.
- **trt** (*int | str*) – The treatment name or number.

Returns The variable value for the treatment.

Return type np.ndarray

remove_treatment(*trt*)
Removes a treatment from the experiment.

Parameters **trt** (*str | int*) – Treatment name or number.

set_general_val(*var*, *val*)
Sets a variable value in the *GENERAL* section of the EXP file.

Parameters

- **var** (*str*) – Variable name.
- **val** (*str | float | int | np.ndarray*) – New value for the variable.

set_trt_factor_level(*trt*, *factor*, *level*)
Sets the factor level for a treatment.

Parameters

- **trt** (*str | int*) – The treatment name or number.
- **factor** (*str*) – The factor name or code.
- **level** (*int*) – The new factor level.

treatments(*name=False*)
Returns the treatments in the run.

Parameters **name** (*bool*) – Whether to return treatment names or numbers.

Returns The list of treatments.

Return type np.ndarray

class tradssat.mgrs.**GeneticMgr**(*crop*, *cult*)

class tradssat.mgrs.**SoilMgr**(*code*)

class tradssat.mgrs.**WeatherFileMgr**(*code*)

Run output manager

class tradssat.mgrs.**DSSATResults**(*folder*)

Facilitates the reading of DSSAT results. Instead of having to read each output file individually, you can simply point this class to a DSSAT run output folder containing the output files and it will find the desired variables for you.

get_final_value(*var*, *trt*)

Returns a variable's final value. Faster than *get_value()* if the variable is available in *Summary.OUT*.

Parameters

- **var** (*str*) – The variable name.
- **trt** (*int*) – The treatment number of interest.

Returns The variable's final value.

Return type str | float | int

get_value(*var*, *trt*, *t=None*, *at='YEAR DOY'*, *run=None*)

Returns the value (point or time-series) of a variable from a DSSAT run.

Parameters

- **var** (*str*) – The variable name.
- **trt** (*int*) – The treatment number of interest.
- **t** (*str | int*) – The time at which one wants the results. If `None`, results will be given for all time steps.
- **at** (*str*) – Must be one of `DAS` (days after start), `DAP` (days after planting), or `YEAR DOY` (year, day of year; default). Only used if `t` is not `None`.

Returns The value of the variable.

Return type np.ndarray

reload()

Reload data (useful if a new DSSAT simulation has been run).

1.5.3 traDSSAT internals

This references traDSSAT's (mostly) internal API and is really only useful if you are planning on contributing to or changing a few things in the source code itself.

Variables

Variable types used to specify DSSAT file variables.

class tradssat.tmpl.var.**CharacterVar**(*name*, *size*, *spc=1*, *sect=None*, *header_fill=' '*, *miss='-99'*, *info=''*)

Character (string) variable.

type_

alias of `builtins.str`

class tradssat.tmpl.var.**FloatVar**(*name*, *size*, *dec*, *lims=None*, *spc=1*, *sect=None*, *header_fill=' '*, *miss='-99'*, *info=''*)

Floating point (decimal) variable.

type_
alias of builtins.float

class tradssat.tmpl.var.**HeaderVariableSet**(*d_vars*)
Organiser for the allowed header variables of a DSSAT file type.

class tradssat.tmpl.var.**IntegerVar**(*name*, *size*, *lims=None*, *spc=1*, *sect=None*, *header_fill=' '*, *miss='-99'*, *info=''*)
Integer (whole number) variable.

type_
alias of builtins.int

class tradssat.tmpl.var.**NumericVar**(*name*, *size*, *lims*, *spc*, *header_fill*, *miss*, *sect*, *info*)
Parent class for numeric variable types.

class tradssat.tmpl.var.**Variable**(*name*, *size*, *spc*, *header_fill*, *float_r*, *miss*, *sect*, *info*)
Parent class for all variable types.

class tradssat.tmpl.var.**VariableSet**(*l_vars*)
Organiser for the allowed variables of a DSSAT file type.

Values

Internal structure to organise DSSAT file data and variable values.

class tradssat.tmpl.vals.**FileValueSet**
Represents the set of values in a DSSAT file.

add_row(*sect*, *subsect=None*, *vals=None*)
Adds a row to the file.

Parameters

- **sect** (*str*) – Name of section.
- **subsect** (*int*) – Subsection number. If None, will add row to all subsections.
- **vals** (*dict*) – Dictionnary of new row variable values.

add_section(*name*)
Adds a section to the file.

Parameters **name** (*str*) – Name of the new section.

changed()
Detects if any variable values have been changed.

Returns

Return type bool

find_var_sect(*var*)
Finds the section in which a variable appears.

Parameters **var** (*str*) – The name of the variable

Returns The file section name.

Return type str

to_dict()
Converts the file to a dictionnary.

Returns

Return type dict

write(*lines*)
Writes the file.

Parameters **lines** (*list[str]*) – List to which to write output lines.

Returns The modified list.

Return type list

class tradssat.tmpl.vals.**HeaderValues**
Represents DSSAT file header variables and their values.

changed()
Checks if the header variables values have been changed.

Returns

Return type bool

get_value(*var*)
Obtain the value of a header variable.

Parameters **var** (*str*) – The variable of interest.

Returns The value of the variable.

Return type np.ndarray

set_vars(*subsect*)
Sets the variables of the header.

Parameters **subsect** (ValueSubSection) – The subsection with variables and their values already specified.

to_dict()
Convert to dictionnary (json) format.

Returns The (mostly) jsonified version of the header's variables.

Return type list

write()
Writes the header values to a string.

Returns

Return type str

class tradssat.tmpl.vals.**ValueSection**(*name*)
Represents the structure and variable values in a DSSAT file section.

add_subsection(*subsect*)
Add a subsection to the section.

Parameters **subsect** (ValueSubSection) – The new subsection.

changed()
Checks whether any value has changed in the section.

Returns

Return type bool

get_header_var(*var*)
Obtain the value of a header variable.

Parameters **var** (*str*) – The name of the variable

Returns The value of the header variable.

Return type np.ndarray

set_header_vars(*h_vars*)
Sets the section's header variables.

Parameters **h_vars** (ValueSubSection) – The subsection representing the header variables.

to_dict()
Converts the section to a dictionnary.

Returns

Return type dict

write(*lines*)
Writes the section to a list of lines.

Parameters **lines** (*list[str]*) – The list of lines to which to append this section.

class tradssat.tmpl.vals.**ValueSubSection**(*l_vars*, *l_vals*)
Represents the variables and values in a DSSAT file subsection.

add_row(*vals=None*)
Adds a row to the subsection.

Parameters **vals** (*dict*) – The values for the new row. Any missing value will be assigned the corresponding missing code for that variable (usually -99).

changed()
Checks whether any variable in the subsection has been changed.

Returns

Return type bool

check_dims()
Checks that all variables in the subsection have the same size. (If not, the subsection cannot write to disk.)

Raises ValueError – If not all variables have the same size.

n_data()
Obtain the size of variables. Will fail if not all variables have the same size.

Returns The size of all variables in the subsection.

Return type int

Raises ValueError – If not all variables have the same size.

to_dict()
Converts the subsection to a dictionnary.

Returns

Return type dict

write(*lines*)
Writes the subsection.

Parameters **lines** (*list[str]*) – The list of lines to which to append this subsection.

class tradssat.tmpl.vals.**VariableValue**(*var*, *val*)
Represents a DSSAT file variable.

add_value(*val*)
Adds a value to the variable's matrix.

Parameters **val** (*int | float*) – The new value.

remove_value(*i*)
Removes a value from the variable's matrix.

Parameters **i** (*np.ndarray*) – The indices of the value(s) to remove. May be a list of indices, or else a boolean mask of the same size as the variable.

set_value(*val*, *i=None*)
Changes the value of the variable.

Parameters

- **val** (*float | int | np.ndarray*) – The new value.
- **i** (*int | np.ndarray*) – Índices at which to change the value. If `None`, all values of the variable will be changed.

size()
Returns the size of the variable.

Returns

Return type int

write(*i=None*)
Converts the variable to a string.

Parameters **i** (*int*) – The index of the value to write. If `None`, the name of the variable will be written instead.

Returns The properly written and formatted variable.

Return type str

Input and output files

All file objects, whether input or output, inherit from the abstract class *File* and all of its methods.

class tradssat.tmpl.**InpFile**(*file*)
Parent class for all input files, as well as for *Summary.OUT*.

changed()
Checks whether the file has been edited and needs to be rewritten.

Returns Whether the file has been edited or not.

Return type bool

classmethod matches_file(*file*)
Checks whether a given file can be read by this class.

Parameters **file** (*str*) – The filename or full path to be read.

Returns `True` if the file matches; `False` otherwise.

Return type bool

class tradssat.tmpl.**OutFile**(*folder*)
Parent class for (nearly all) DSSAT output files.

classmethod matches_file(*file*)
Checks whether a given file can be read by this class.

Parameters **file** (*str*) – The filename or full path to be read.

Returns `True` if the file matches; `False` otherwise.

Return type bool

File template

All file objects, whether input or output, inherit from the abstract class *File* and all of its methods.

class tradssat.tmpl.**File**(*file*)
Parent class for all file objects.

get_value(*var*, *sect=None*, *subsect=None*, *cond=None*)

Parameters

- **var** –
- **sect** –
- **subsect** –
- **cond** –

Returns

Return type np.ndarray

get_var_size(*var*, *sect=None*)
Returns the size of a variable.

Parameters

- **var** (*str*) – The name of the variable.
- **sect** (*str*) – The name of the section in which this variable appears (optional; for ambiguous cases where a file has several variables with the same code).

Returns The size of the variable.

Return type int

classmethod matches_file(*file*)
Checks whether a given file can be read by this class. Must be implemented in subclasses.

Parameters **file** (*str*) – The file to be read.

Returns `True` if the file matches; `False` otherwise.

Return type bool

1.6 Contributing

There are a few ways to contribute to traDSSAT...

1.6.1 Error reporting

Something looks fishy? Please report it on GitHub.

1.6.2 Improvements

Think you can fix it yourself? Great! Fork traDSSAT on GitHub and submit a pull request with your changes.

1.7 Acknowledgements

1.7.1 Authors

TraDSSAT was written by:

- Julien Jean Malard
- (Shreya Yadav)

CHAPTER 2

Indices and tables

- genindex
- search

Python Module Index

t

Index

M

N

O

P

R

S

T

V

W