Package ‘DSSAT’

February 19, 2024

Type Package

Title A Comprehensive R Interface for the DSSAT Cropping Systems Model

Version 0.0.9

Date 2024-01-19

Author Phillip D. Alderman [aut, cre]

Maintainer Phillip D. Alderman <phillip.alderman@okstate.edu>

Description The purpose of this package is to provide a comprehensive
R interface to the Decision Support System for Agrotechnology
Transfer Cropping Systems Model (DSSAT-CSM; see <https://dssat.net> for more information).
The package provides cross-platform functions to read and
write input files, run DSSAT-CSM, and read output files.

License GPL (>= 3)

Encoding UTF-8

Imports dplyr (>= 1.0.0), glue, lubridate, magrittr, methods, purrr,
readr, rlang, stringr, tibble, tidyr, tidyselect, utils

RoxygenNote 7.2.3

Suggests knitr, rmarkdown

BugReports https://github.com/palderman/DSSAT/issues

Config/testthat/edition 3

NeedsCompilation no

Repository CRAN

Date/Publication 2024-02-19 18:30:02 UTC

R topics documented:

  add_v_fmt .......................................................... 2
  as_DSSAT_tbl ..................................................... 3
  calc_AMP .......................................................... 3
  calc_TAV .......................................................... 4
**add_v_fmt**

Adds variable format information to a tibble

**Description**

Adds variable format information to a tibble

**Usage**

```
add_v_fmt(input_tbl, v_fmt = NULL)
```

**Arguments**

- `input_tbl` a tibble
- `v_fmt` a named vector containing variable format information to be added to ‘input_tbl’

**Value**

a tibble containing the original tibble with an additional attribute that contains variable format information
Examples

# Extract file path for sample ecotype file
sample_eco_file <- system.file('extdata','SAMPLE.ECO',package='DSSAT')

# Read sample ecotype file
eco <- read_eco(sample_eco_file)

# Replace formats for TSEN and CDAY parameters
eco <- add_v_fmt(eco,v_fmt=c(TSEN='%6.1f',CDAY='%6.1f'))

---

**as_DSSAT_tbl**

*Convert tibble to DSSAT_tbl*

**Description**

Convert tibble to DSSAT_tbl

**Usage**

```r
as_DSSAT_tbl(tbl_in, v_fmt = NULL, tier_info = NULL)
```

**Arguments**

- `tbl_in`: a tibble
- `v_fmt`: a character vector specifying the sprintf() format for each column
- `tier_info`: a list of character vectors storing the history of which original table that columns came from for tibbles that are comprised of multiple joined tables

**Value**

a tibble of class DSSAT_tbl

---

**calc_AMP**

*Calculate long-term temperature amplitude (AMP)*

**Description**

Calculate long-term temperature amplitude (AMP)

**Usage**

```r
calc_AMP(wth)
```
Arguments

wth a data frame that contains weather data formatted as would be generated using `read_wth`, namely including columns DATE, TMAX and TMIN.

---

**calc_TAV** Calculate long-term temperature average (TAV)

---

**Description**

Calculate long-term temperature average (TAV)

**Usage**

calc_TAV(wth)

**Arguments**

wth a data frame that contains weather data formatted as would be generated using `read_wth`, namely including columns TMAX and TMIN.

---

**clear_output** Clear output files

---

**Description**

A function to delete DSSAT output files (*.OUT) and intermediate files (*.INH, *.INP and *.LST) from a directory

**Usage**

clear_output(dir_name = getwd(), file_ext = c("OUT", "LST", "INP", "INH"))

**Arguments**

dir_name a length-one character vector indicating the directory in which to delete output files (by default the current working directory)

file_ext a character vector of file extensions to delete
**mutate_cond**

Convenience function that allows mutating a subset of rows

**Description**

Convenience function that allows mutating a subset of rows

**Usage**

```r
mutateCond(.data, condition, ..., envir = parent.frame())
```

**Arguments**

- `.data` a tibble
- `condition` a logical vector for subsetting rows of `.data`
- `...` Name-value pairs of expressions to be evaluated by `mutate()`
- `envir` environment within which expressions should be evaluated

**Details**

Original code taken from [https://stackoverflow.com/questions/34096162/dplyr-mutate-replace-several-columns-on-a-subset-of-rows](https://stackoverflow.com/questions/34096162/dplyr-mutate-replace-several-columns-on-a-subset-of-rows)

**Value**

a tibble with specified rows modified

---

**read_cul**

Reads parameters from a single DSSAT cultivar parameter file (*.CUL)

**Description**

Reads parameters from a single DSSAT cultivar parameter file (*.CUL)

**Usage**

```r
read_cul(
  file_name,
  col_types = NULL,
  col_names = NULL,
  left_justified = c("VAR#", "VARNAME\.*", "VAR-NAME\.*", "VRNAME\.*"),
  use_std_fmt = TRUE
)
```
read_dssat

Arguments

- **file_name**: a character vector of length one that contains the name of a single DSSAT output file.
- **col_types**: One of NULL, a cols() specification, or a string. See `read_fwf` or vignette("readr") for more details.
- **col_names**: A character vector of column names; primarily helpful for cases where there are no white spaces between column names within the header line.
- **left_justified**: A character vector of column names that should be left justified.
- **use_std_fmt**: logical value indicating whether to read FileX using the standard column formats. If FALSE, column formats will be inferred from tier headers.

Value

a tibble containing the data from the raw DSSAT output

Examples

```r
# Extract file path for sample cultivar file path
sample_cul_file <- system.file("extdata", "SAMPLE.CUL", package="DSSAT")

# Read sample cultivar file
cul <- read_cul(sample_cul_file)
```

Description

Reads data from a single DSSAT file

Usage

```r
read_dssat(
  file_name,
  col_types = NULL,
  col_names = NULL,
  na_strings = NULL,
  left_justified = "EXCODE",
  guess_max = 10
)
```
read_dssbatch

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file_name</td>
<td>a character vector of length one that contains the name of a single DSSAT output file</td>
</tr>
<tr>
<td>col_types</td>
<td>One of NULL, a cols() specification, or a string. See read_fwf or vignette(&quot;readr&quot;) for more details.</td>
</tr>
<tr>
<td>col_names</td>
<td>A character vector of column names; primarily helpful for cases where there are no white spaces between column names within the header line</td>
</tr>
<tr>
<td>na_strings</td>
<td>A character vector containing strings that should be interpreted as missing values</td>
</tr>
<tr>
<td>left_justified</td>
<td>A character vector containing strings that should be interpreted as missing values</td>
</tr>
<tr>
<td>guess_max</td>
<td>An integer indicating the maximum number of lines that should be used to guess the type of a column</td>
</tr>
</tbody>
</table>

Value

a tibble containing the data from the raw DSSAT output

Examples

```r
# Extract file path for sample output file path
sample_output <- system.file('extdata','SAMPLE.OUT',package='DSSAT')
read_dssat(sample_output)
```

---

**Description**

Reads data from a single DSSAT batch file

**Usage**

```r
read_dssbatch(file_name = "DSSBatch.V47")
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file_name</td>
<td>a character vector of length one that contains the name of a single DSSAT batch file</td>
</tr>
</tbody>
</table>

**Value**

a tibble containing the data from the DSSAT batch file
Examples

```r
# Create example batch file path
batch_file_path <- paste0(tempdir(), '/quotesingle.Var/DSSBatch.V47/quotesingle.Var')

# Write example batch file
write_dssbatch(x='UFGA0601.BMX', trtno=1:4, file_name = batch_file_path)

# Read example batch file
dssbatch <- read_dssbatch(batch_file_path)
```

---

**read_eco**  
*Reads parameters from a single DSSAT ecotype parameter file (*.ECO)*

**Description**

Reads parameters from a single DSSAT ecotype parameter file (*.ECO)*

**Usage**

```r
read_eco(
  file_name,
  col_types = NULL,
  col_names = NULL,
  left_justified = c("ECO ", "ECO#", "ECONAME\.\.*", "ECO-NAME\.\.*")
)
```

**Arguments**

- `file_name`: A character vector of length one that contains the name of a single DSSAT output file
- `col_types`: One of NULL, a cols() specification, or a string. See `read_fwf` or vignette("readr") for more details.
- `col_names`: A character vector of column names; primarily helpful for cases where there are no white spaces between column names within the header line
- `left_justified`: A character vector of column names that should be left justified

**Value**

A tibble containing the data from the raw DSSAT file
Examples

# Extract file path for sample ecotype file path
sample_eco <- system.file('extdata', 'SAMPLE.ECO', package='DSSAT')

eco <- read_eco(sample_eco)

---

read_filea  

*Reads data from a single DSSAT file* A

Description

Reads data from a single DSSAT file A

Usage

`read_filea(file_name, col_types = NULL, col_names = NULL, na_strings = NULL)`

Arguments

- `file_name` 
  a character vector of length one that contains the name of a single DSSAT output file
- `col_types` 
  One of NULL, a cols() specification, or a string. See `read_fwf` or vignette("readr") for more details.
- `col_names` 
  A character vector of column names; primarily helpful for cases where there are no white spaces between column names within the header line
- `na_strings` 
  a character vector of string to represent missing values

Value

a tibble containing the data from the raw DSSAT file

Examples

# Extract FileA path for sample file
sample_filea <- system.file('extdata', 'SAMPLE.CRA', package='DSSAT')

filea <- read_filea(sample_filea)
### read_filex

**Reads input data from a single DSSAT experiment file (*.x)**

**Description**

Reads input data from a single DSSAT experiment file (*.x)

**Usage**

```r
read_filex(file_name, col_types = NULL, col_names = NULL, na_strings = NULL)
```

**Arguments**

- **file_name**: a character vector of length one that contains the name of a single DSSAT output file
- **col_types**: One of NULL, a cols() specification, or a string. See `read_fwf` or `vignette("readr")` for more details.
- **col_names**: A character vector of column names; primarily helpful for cases where there are no white spaces between column names within the header line
- **na_strings**: a character vector of string to represent missing values

**Value**

a tibble containing the data from the raw DSSAT file

**Examples**

```r
# Extract FileT path for sample file
sample_file <- system.file('extdata', 'SAMPLE.CRT', package='DSSAT')

read_filex(sample_file)
```
**Usage**

```r
def read_filex(  
  file_name,  
  col_types = NULL,  
  col_names = NULL,  
  na_strings = NULL,  
  store_v_fmt = FALSE,  
  use_std_fmt = FALSE  
)
```

**Arguments**

- `file_name`: a character vector of length one that contains the name of a single DSSAT output file
- `col_types`: One of NULL, a cols() specification, or a string. See `read_fwf` or vignette("readr") for more details.
- `col_names`: A character vector of column names; primarily helpful for cases where there are no white spaces between column names within the header line
- `na_strings`: A character vector containing strings that should be interpreted as missing values
- `store_v_fmt`: a logical value indicating whether or not to store the format for variables being read
- `use_std_fmt`: logical value indicating whether to read FileX using the standard column formats. If FALSE, column formats will be inferred from tier headers

**Value**

a tibble containing the data from the raw DSSAT file

---

**read_filex_multiple**

**Read multiple File X**

**Description**

Read multiple File X

**Usage**

```r
def read_filex_multiple(  
  file_name,  
  col_types = NULL,  
  col_names = NULL,  
  na_strings = NULL,  
  store_v_fmt = FALSE,  
  use_std_fmt = TRUE  
)
```
Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file_name</td>
<td>a character vector of length one that contains the name of a single DSSAT output file</td>
</tr>
<tr>
<td>col_types</td>
<td>One of <code>NULL</code>, a <code>cols()</code> specification, or a string. See <code>read_fwf</code> or <code>vignette(&quot;readr&quot;)</code> for more details.</td>
</tr>
<tr>
<td>col_names</td>
<td>A character vector of column names; primarily helpful for cases where there are no white spaces between column names within the header line</td>
</tr>
<tr>
<td>na_strings</td>
<td>A character vector containing strings that should be interpreted as missing values</td>
</tr>
<tr>
<td>store_vFmt</td>
<td>a logical value indicating whether or not to store the format for variables being read</td>
</tr>
<tr>
<td>use_std_fmt</td>
<td>logical value indicating whether to read FileX using the standard column formats. If <code>FALSE</code>, column formats will be inferred from tier headers</td>
</tr>
</tbody>
</table>

**read_output**

*Reads data from a single DSSAT output file*

**Description**

Reads data from a single DSSAT output file

**Usage**

```r
read_output(
  file_name, 
  col_types = NULL, 
  col_names = NULL, 
  left_justified = NULL, 
  read_only = NULL, 
  store_vFmt = FALSE
)
```

**Arguments**

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>file_name</td>
<td>a character vector of length one that contains the name of a single DSSAT output file</td>
</tr>
<tr>
<td>col_types</td>
<td>One of <code>NULL</code>, a <code>cols()</code> specification, or a string. See <code>read_fwf</code> or <code>vignette(&quot;readr&quot;)</code> for more details.</td>
</tr>
<tr>
<td>col_names</td>
<td>A character vector of column names; primarily helpful for cases where there are no white spaces between column names within the header line</td>
</tr>
<tr>
<td>left_justified</td>
<td>A character vector of column names that should be left justified</td>
</tr>
<tr>
<td>read_only</td>
<td>NULL or a character vector of column names that should be read in; If non-NULL only the columns listed will be read in.</td>
</tr>
<tr>
<td>store_vFmt</td>
<td>a logical value indicating whether or not to store the format for variables being read</td>
</tr>
</tbody>
</table>
**read_pest**

Value

a tibble containing the data from the raw DSSAT output

Examples

```r
# Extract file path for sample output file path
sample_output <- system.file('extdata', 'SAMPLE.OUT', package='DSSAT')

out <- read_output(sample_output)
```

**Description**

Reads input data from a single DSSAT pest file (*.PST)

**Usage**

```r
read_pest(file_name, col_types = NULL, col_names = NULL)
```

**Arguments**

- `file_name`: a character vector of length one that contains the name of a single DSSAT output file
- `col_types`: One of NULL, a cols() specification, or a string. See `read_fwf` or vignette("readr") for more details.
- `col_names`: A character vector of column names; primarily helpful for cases where there are no white spaces between column names within the header line

**Value**

a tibble containing the data from the raw DSSAT file
**read_soil_profile** *Reads a single DSSAT formatted soil profile from a raw character vector*

**Description**
Reads a single DSSAT formatted soil profile from a raw character vector.

**Usage**

```r
read_soil_profile(
  raw_lines,
  left_justified = NULL,
  col_types = NULL,
  col_names = NULL
)
```

**Arguments**

- `raw_lines` a character vector that includes the contents of a single tier of data (including headline, but excluding version stamp and other header information) from a DSSAT output file
- `left_justified` A character vector of column names that should be left justified
- `col_types` One of NULL, a cols() specification, or a string. See `read_fwf` or vignette("readr") for more details.
- `col_names` A character vector of column names; primarily helpful for cases where there are no white spaces between column names within the header line

**Value**
a list containing tibbles for each tier of a DSSAT formatted soil profile

**Examples**

```r
g <- c("*IB00000001 IBSNAT SIC 210 DEFAULT - DEEP SILTY CLAY",
      "$SITE COUNTRY LAT LONG SCS FAMILY",
      "$ Generic Generic -99 -99 Generic",
      "@ SCOM SALB SLU1 SLDR SLRO SLNF SLPF SMHB SMPX SMKE",
      " -99 0.11 6.0 0.30 85.0 1.00 1.00 IB001 IB001 IB001",
      "$ SLB SLMH SLLL SDUL SSAT SRGF SSKS SBDM SLOC SLCL SLSI SLNF SLHW SLHB",
      "$ 5 -99 0.228 0.385 0.481 1.000 -99 1.30 1.75 50.0 45.0 0.0 0.170 6.5 -99",
      "$ 15 -99 0.228 0.385 0.481 1.000 -99 1.30 1.75 50.0 45.0 0.0 0.170 6.5 -99",
      "$ 30 -99 0.249 0.406 0.482 0.638 -99 1.30 1.60 50.0 45.0 0.0 0.170 6.5 -99",
      "$ 45 -99 0.249 0.406 0.465 0.472 -99 1.35 1.45 50.0 45.0 0.0 0.140 6.5 -99",
      "$ 60 -99 0.249 0.406 0.465 0.350 -99 1.35 1.45 50.0 45.0 0.0 0.140 6.5 -99",
      "$ 90 -99 0.308 0.456 0.468 0.223 -99 1.35 1.10 50.0 45.0 0.0 0.110 6.5 -99",
```
```r
read_sol

# Extract file path for sample soil file
sample_sol <- system.file("extdata", "SAMPLE.SOL", package = "DSSAT")

sol <- read_sol(sample_sol)
```

#### Description

Reads soil parameters from a single DSSAT soil parameter file (*.SOL)

#### Usage

```r
read_sol(
  file_name,  
  id_soil = NULL,  
  left_justified = NULL,  
  col_types = NULL,  
  col_names = NULL
)
```

#### Arguments

- **file_name**: a character vector of length one that contains the name of a single DSSAT output file
- **id_soil**: a length-one character vector containing the soil ID code for a single soil profile
- **left_justified**: A character vector of column names that should be left justified
- **col_types**: One of NULL, a cols() specification, or a string. See `read_fwf` or vignette("readr") for more details.
- **col_names**: A character vector of column names; primarily helpful for cases where there are no white spaces between column names within the header line

#### Value

A tibble containing the data from the raw DSSAT file

#### Examples

```r
# Extract file path for sample soil file
sample_sol <- system.file("extdata", "SAMPLE.SOL", package = "DSSAT")

sol <- read_sol(sample_sol)
```
Description

Reads and combines data and header information from a single tier of a DSSAT output file

Usage

```r
read_tier(
  raw_lines,
  col_types = NULL,
  col_names = NULL,
  na_strings = NULL,
  left_justified = "EXCODE",
  guess_max = 1000,
  store_v_fmt = TRUE,
  read_only = NULL
)
```

Arguments

- `raw_lines`: a character vector that includes the contents of a single tier of data (including headline, but excluding version stamp and other header information) from a DSSAT output file
- `col_types`: One of NULL, a cols() specification, or a string. See `read_fwf` or `vignette("readr")` for more details.
- `col_names`: A character vector of column names; primarily helpful for cases where there are no white spaces between column names within the header line
- `na_strings`: a character vector of string to represent missing values
- `left_justified`: A character vector of column names that should be left justified
- `guess_max`: An integer indicating the maximum number of lines that should be used to guess the type of a column
- `store_v_fmt`: a logical value indicating whether or not to store the format for variables being read
- `read_only`: NULL or a character vector of column names that should be read in; If non-NULL only the columns listed will be read in.

Value

a tibble containing the data from the raw DSSAT output
Examples

```r
c <- read_tier(sample_data_tier)
```

**Description**

Reads weather input data from a single DSSAT weather file (*.WTH)

**Usage**

```r
read_wth(file_name, col_types = NULL, col_names = NULL)
```

**Arguments**

- `file_name`: a character vector of length one that contains the name of a single DSSAT output file
- `col_types`: One of NULL, a cols() specification, or a string. See `read_fwf` or vignette("readr") for more details.
- `col_names`: A character vector of column names; primarily helpful for cases where there are no white spaces between column names within the header line

**Value**

a tibble containing the data from the raw DSSAT file
Examples

# Extract file path for sample weather file
sample_wth <- system.file("extdata", "SAMPLE.WTH", package = "DSSAT")

read_wth(sample_wth)

run_dssat W\text{rites data from a single DSSAT data tier}

Description

Writes data from a single DSSAT data tier

Usage

run_dssat(run_mode = "B", file_name = NULL, suppress_output = FALSE, wd = NULL)

Arguments

Run_mode a length-one character vector that specifies the run mode that should be used for the DSSAT simulation. One of: A - Run all treatments. User specifies fileX on the command line and the model runs all treatments B - Batch mode. User defines fileX and treatment numbers in Batch file C - Command line mode. Use input from the command line. D - Debug mode. Model skips input module and reads temp file from the command line E - Sensitivity analysis. User defines fileX and treatment number in Batch file F - Farm model. Use Batch file to define experiment G - Gencalc. Use Command line to define experiment and treatment I - Interactive mode. Use model interface for exp. & trtno. L - Gene based model (Locus). Use Batch file to define experiment N - Seasonal analysis. Use Batch file to define experiment and treatments Q - Sequence analysis. Use Batch file to define experiment S - Spatial. Use Batch file to define experiment T - Gencalc. Use Batch file to define experiments and treatment

file_name a length-one character vector that specifies the file name to be used for simulation. Usually the name of a batch file or file X.

suppress_output a logical value indicating whether to suppress DSSAT-CSM output from being printed to the console

wd an optional character string that specifies the working directory within which to run DSSAT-CSM. If left NULL, DSSAT-CSM will be run in the current working directory

Value

Invisibly returns the console output from running DSSAT-CSM
Examples

```r
## Not run:
run_dssat()
## End(Not run)
```

### write_cul

**Description**

Reads parameters from a single DSSAT cultivar parameter file (*.CUL)

**Usage**

```r
write_cul(cul, file_name)
```

**Arguments**

- `cul` a DSSAT_tbl containing the contents of a DSSAT cultivar parameter file
- `file_name` a character vector of length one that contains the name of a single DSSAT output file

**Value**

a tibble containing the data from the raw DSSAT output

**Examples**

```r
# Extract file path for sample cultivar file path
sample_cul_file <- system.file('extdata', 'SAMPLE.CUL', package='DSSAT')

# Read sample cultivar file
cul <- read_cul(sample_cul_file)

# Create example cultivar file path
sample_cul_file2 <- paste0(tempdir(), '/SAMPLE.CUL')

# Write out sample cultivar file
write_cul(cul, sample_cul_file2)
```
write_dssbatch

Constructs and writes a DSSAT simulation batch file

Description

Constructs and writes a DSSAT simulation batch file

Usage

write_dssbatch(x, trtno = 1, rp = 1, sq = 0, op = 0, co = 0, file_name = NULL)

Arguments

x

a tibble/data frame or character vector; if a tibble, it should contain all required columns of a DSSAT batch file (FILEX, TRTNO, RP, SQ, OP, CO); if a character vector, it should contain FileX file names

trtno, rp, sq, op, co

a numeric vector

file_name

an optional character vector of the intended batch file name

Value

invisibly returns a character vector containing the content of a DSSAT batch file

Examples

# Create example batch file path
batch_file_path <- paste0(tempdir(), '\DSSBatch.V47')

# Write example batch file
write_dssbatch(x='UFGA0601.BMX', trtno=1:4, file_name = batch_file_path)

# Construct tibble for DSSAT batch file
batch_tbl <- data.frame(FILEX='UFGA0601.BMX',
                        TRTNO=1:4,
                        RP=1,
                        SQ=0,
                        OP=0,
                        CO=0)

# Write example batch file
write_dssbatch(batch_tbl, file_name = batch_file_path)
write_eco

**Description**

Reads parameters from a single DSSAT ecotype parameter file (*.ECO)

**Usage**

```r
write_eco(eco, file_name)
```

**Arguments**

- `eco`: a DSSAT_tbl containing the contents of a DSSAT ecotype parameter file
- `file_name`: a character vector of length one that contains the name of a single DSSAT output file

**Value**

a tibble containing the data from the raw DSSAT output

**Examples**

```r
# Extract file path for sample ecotype file path
sample_eco_file <- system.file('extdata', 'SAMPLE.ECO', package='DSSAT')

# Read sample ecotype file
eco <- read_eco(sample_eco_file)

# Create example ecotype file path
sample_eco_file2 <- paste0(tempdir(), '/SAMPLE.ECO')

# Write out sample ecotype file
write_eco(eco, sample_eco_file2)
```

write_filea

**Description**

Writes data to a single DSSAT file A
Usage

write_filea(filea, file_name, drop_duplicate_rows = TRUE)

Arguments

filea  
a tibble containing the data to write to a DSSAT file A

file_name  
a character vector of length one that contains the name of a single DSSAT file
into which 'filea' will be written

drop_duplicate_rows  
a logical value indicating whether duplicate rows should be dropped from tier_data

Examples

# Extract FileA path for sample file
sample_filea <- system.file('extdata','SAMPLE.CRA',package='DSSAT')

filea <- read_filea(sample_filea)

# Create example FileA file path
sample_filea2 <- paste0(tempdir(),'SAMPLE.CRA')

# Write out sample FileA
write_filea(filea,sample_filea2)

write_filet

Writes data to a single DSSAT file T

Description

Writes data to a single DSSAT file T

Usage

write_filet(filet, file_name, drop_duplicate_rows = TRUE)

Arguments

filet  
a tibble containing the data to write to a DSSAT file T

file_name  
a character vector of length one that contains the name of a single DSSAT file
into which 'filet' will be written

drop_duplicate_rows  
a logical value indicating whether duplicate rows should be dropped from tier_data
write_filex

Examples

# Extract FileT path for sample file
sample_filet <- system.file('extdata','SAMPLE.CRT',package='DSSAT')

filet <- read_filet(sample_filet)

# Create example FileT file path
sample_filet2 <- paste0(tempdir(),'SAMPLE.CRT')

# Write out sample FileA
write_filet(filet,sample_filet2)

write_filex                 Writes data to a single DSSAT FileX

Description

Writes data to a single DSSAT FileX

Usage

write_filex(filex, file_name, drop_duplicate_rows = TRUE, force_std_fmt = TRUE)

Arguments

filex  a list of tibbles containing the data to write to a DSSAT file X
file_name  a character vector of length one that contains the name of a single DSSAT file
into which `filet` will be written
drop_duplicate_rows  a logical value indicating whether duplicate rows should be dropped from tier_data
force_std_fmt  a logical value indicating whether to override the variable format stored within
the FileX object with standard DSSAT formatting

write_sol                 Writes soil parameters to a single DSSAT soil parameter file (*.SOL)

Description

Writes soil parameters to a single DSSAT soil parameter file (*.SOL)

Usage

write_sol(sol, file_name, title = NULL, append = TRUE, force_std_fmt = TRUE)
write_tier

 Writes data from a single DSSAT data tier

**Description**

Writes data from a single DSSAT data tier

**Usage**

```r
write_tier(
  tier_data,
  pad_name = NULL,
  drop_duplicate_rows = FALSE,
  drop_na_rows = TRUE
)
```
write_wth

Arguments

- tier_data: a tibble containing the data to write out
- pad_name: a character vector of column names for which to add leading spaces/trailing periods
- drop_duplicate_rows: a logical value indicating whether duplicate rows should be dropped from tier_data
- drop_na_rows: a logical value indicating whether rows containing all NA values should be dropped from tier_data

Value

a character vector

Examples

tier_data <- data.frame(TRNO=1:4,HWAM=rnorm(4,2000,250))
tier_data <- add_v_fmt(tier_data,v_fmt=c(TRNO='%6.0f', HWAM='%6.0f'))
output <- write_tier(tier_data)

write_wth(Writes data to a single DSSAT weather file)

Description

Writes data to a single DSSAT weather file

Usage

write_wth(
  wth,
  file_name,
  force_std_fmt = TRUE,
  location = NULL,
  comments = NULL,
  INSI = NULL,
  LAT = NULL,
  LONG = NULL,
  ELEV = NULL,
  TAV = NULL,
  AMP = NULL,
  REFHT = NULL,
  WNDHT = NULL,
  CO2 = NULL
)

Arguments

- **wth**
  - A tibble containing the data to write to a DSSAT weather file.

- **file_name**
  - A character vector of length one that contains the name of a single DSSAT file
    into which ‘wth’ will be written.

- **force_std_fmt**
  - A logical value indicating whether to override the variable format stored within
    the ‘wth’ object with standard DSSAT formatting.

- **location**
  - A character value that gives the location for the weather file header line.

- **comments**
  - A character vector containing any comments to be included in the weather file.

- **INSI**
  - Institute and site code (four-digit character code).

- **LAT**
  - Latitude in decimal degrees.

- **LONG**
  - Longitude in decimal degrees.

- **ELEV**
  - Elevation in meters.

- **TAV**
  - Long-term average air temperature at reference height (typically 2 meters).

- **AMP**
  - Long-term monthly air temperature amplitude at reference height (typically 2
    meters).

- **REFHT**
  - Reference height for air temperature measurements.

- **WNDHT**
  - Reference height for wind speed measurements.

- **CO2**
  - Carbon dioxide concentration in parts per million.
Index

add_v_fmt, 2
as_DSSAT_tbl, 3

calc_Amp, 3
calc_TAV, 4
clear_output, 4

mutate_cond, 5

read_cul, 5
read_dssat, 6
read_dssbatch, 7
read_eco, 8
read_filea, 9
read_filet, 10
read_filex, 10
read_filex_multiple, 11
read_fwf, 6–17
read_output, 12
read_pest, 13
read_soil_profile, 14
read_sol, 15
read_tier, 16
read_wth, 4, 17
run_dssat, 18

write_cul, 19
write_dssbatch, 20
write_eco, 21
write_filea, 21
write_filet, 22
write_filex, 23
write_sol, 23
write_tier, 24
write_wth, 25