

Package: surfaceuv (via r-universe)

October 1, 2026

Type Package

Title Solar Ultraviolet at Ground Surface Data Import

Version 0.1.1.9001

Date 2024-11-18

Description Functions for reading files containing data on ``Surface UV" such as daily doses, irradiances and the UV Index (UVI). Currently the Surface UV product from the AC SAF project of EUMETSAT is supported in both geographic grid (HDF5) and time series (text) formats. Additionally the OMI/Aura Surface UV product files in geographic grid NetCDF4 and HDF5 formats is supported.

License GPL (>= 2)

VignetteBuilder knitr

Depends R (>= 4.0.0)

Imports utils, bit64, rhdf5 (>= 2.45.0)

Suggests knitr (>= 1.43), rmarkdown (>= 2.23), spelling, testthat (>= 3.1.10)

LazyLoad yes

ByteCompile true

Encoding UTF-8

URL <https://docs.r4photobiology.info/surfaceuv/>

BugReports <https://github.com/aphalo/surfaceuv/issues/>

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

Language en-US

Config/testthat/edition 3

Config/pak/sysreqs libssl-dev

Repository <https://aphalo.r-universe.dev>

Date/Publication 2024-11-28 12:37:07 UTC

RemoteUrl https://github.com/aphalo/surfaceuv
RemoteRef HEAD
RemoteSha 7393658184e12f96b4a6cfe71c89cac03745aff5

Contents

check_files	2
nearest_corners	3
nearest_point	4
sUV_read_OMUVBd_he5	5
sUV_read_OMUVBd_nc4	8
sUV_read_OUV_hdf5	10
sUV_read_OUV_txt	13
Index	17

check_files	<i>Check files</i>
-------------	--------------------

Description

Internal utility function used to check that file names passed as argument are valid.

Usage

check_files(files, name.pattern = NULL)

Arguments

- files character A vector of file names, no other limitation in length than available memory to hold the data.
- name.pattern character A vector of accepted file name patterns, used as pattern in a call to [grepl\(\)](#) If NULL the test is skipped, if match fails, a warning is issued.

Details

Accepts a character vector or a list of character vectors, returning always a character vector. The character strings are assumed to be paths to files. If the files pointed at cannot be accessed, and error is triggered. If the files exist, but one or more do not match the expected name.pattern a warning is triggered.

Value

A character vector with one or more paths to files as members.

nearest_corners	<i>Nearest 2D grid region</i>
-----------------	-------------------------------

Description

Internal utility functions used to find indexes to rectangular grid regions based on Euclidean distance.

Usage

```
nearest_corners(  
    x,  
    y,  
    x.target,  
    y.target,  
    x.step = Inf,  
    y.step = Inf,  
    envelope = "nearest"  
)
```

```
nearest_region(  
    x,  
    y,  
    x.target,  
    y.target,  
    x.step = Inf,  
    y.step = Inf,  
    envelope = "nearest"  
)
```

```
nearest_rcoord(  
    x,  
    y,  
    x.target,  
    y.target,  
    x.step = Inf,  
    y.step = Inf,  
    envelope = "nearest"  
)
```

```
nearest_ccoord(  
    x,  
    y,  
    x.target,  
    y.target,  
    x.step = Inf,  
    y.step = Inf,
```

```
    envelope = "nearest"  
  )
```

Arguments

- x, y numeric vectors of equal length representing 2D coordinate pairs..
- x.target, y.target numeric of length two giving the coordinates of the corners of the target region.
- x.step, y.step numeric of length one equal or larger than the largest coordinate steps between grid points in x and y.
- envelope character One of "nearest", "outer" or "inner".

Details

The values of x.step and y.step are used to constrain the computation of distances to the neighbourhood of the target. This is to improve performance. The default envelope = "outer" returns the coordinates of the smallest enclosing region.

Value

A vector of indices, assuming that x and y are coordinates variables of data stored in long form.

Note

For "inner" and "outer" envelopes the region is shrunk or expanded by half a step. If the step is Inf, the largest steps found in x and y replace Inf.

nearest_point	<i>Nearest 2D grid point</i>
---------------	------------------------------

Description

Internal utility function used to find the nearest grid point in a 2D coordinate system based on Euclidean distance.

Usage

```
nearest_point(x, y, x.target, y.target, x.step = Inf, y.step = Inf)  
  
nearest_pcoord(x, y, x.target, y.target, x.step = Inf, y.step = Inf)  
  
dist2target(x, y, x.target, y.target, x.step = Inf, y.step = Inf)
```

Arguments

`x, y` numeric vectors of equal length representing 2D coordinate pairs..
`x.target, y.target` numeric of length one giving the coordinates of the nearest neighbor search.
`x.step, y.step` numeric of length one equal or larger than the largest coordinate steps between grid points in `x` and `y`

Details

The values of `x.step` and `y.step` are used to constrain the computation of distances to the neighborhood of the target. This is to improve performance.

Value

A numeric index, assuming that `x` and `y` are coordinates variables of data stored in long form in the case of `nearest_point()` or the values of `x` and `y` at this grid point in the case of `nearest_pcoord()`.

sUV_read_OMUVBd_he5	<i>OMI/Aura Surface UV gridded original</i>
---------------------	---

Description

Import **gridded** "Surface UV" data released by FMI/NASA from **HDF5** files with global coverage downloaded from the NASA EARTHDATA server.

Usage

```

sUV_read_OMUVBd_he5(
  files,
  data.product = NULL,
  group.name = "OMI UVB Product/Data Fields",
  vars.to.read = NULL,
  fill = NA_real_,
  verbose = interactive()
)

sUV_vars_OMUVBd_he5(
  files,
  data.product = NULL,
  group.name = "OMI UVB Product/Data Fields",
  set.oper = "intersect"
)

sUV_grid_OMUVBd_he5(
  files,
  group.name = "OMI UVB Product/Data Fields",

```

```

    expand = FALSE
  )

  sUV_date_OMUVBd_he5(files, use.names = length(files > 1))

```

Arguments

<code>files</code>	character A vector of file names, no other limitation in length than available memory to hold the data.
<code>data.product</code>	character Currently only "Surface UV" supported.
<code>group.name</code>	character The name of the 'group' in the HDF5 files, or a regular expression for matching a single group name with <code>grep()</code> .
<code>vars.to.read</code>	character A vector of variable names. If NULL all the variables present in the first file are read.
<code>fill</code>	numeric The R value used to replace the fill value used in the file, which is retrieved from the file metadata, and also used to fill missing variables.
<code>verbose</code>	logical Flag indicating if progress, and time and size of the returned object should be printed.
<code>set.oper</code>	character One of "intersect", or "union".
<code>expand</code>	logical Flag indicating whether to return ranges or a full grid.
<code>use.names</code>	logical. Should names be added to the returned vector?

Details

Function `sUV_read_OMUVBd_he5()` can be used to read the data stored in a file, either in full or selected variables. Query functions `sUV_vars_OMUVBd_he5()`, `sUV_grid_OMUVBd_he5()` and `sUV_date_OMUVBd_he5()` extract the names of the variables, the range of the grid and the dates of measurements much more efficiently than by using `sUV_read_OMUVBd_he5()`. The dates are decoded from the file names, expecting these to be those set by the data provider. The grid is expected to be identical in all files that are imported in a call to `sUV_read_OMUVBd_he5()`, and grid subsetting is currently not supported. If not all the files named in the argument to `files` are accessible, an error is triggered early. If the files differ in the grid, an error is triggered when reading the first mismatching file. Missing variables named in `vars.to.read` if detected when reading the first file, are filled with the `fill` value, otherwise they trigger an error when an attempt is made to read them.

Value

Function `sUV_read_OMUVBd_he5()` returns a data frame with columns named "Date", "Longitude", "Latitude", and the data variables with their original names. The data variables have their metadata stored as R attributes. `sUV_vars_OMUVBd_he5()` returns a character vector of variable names, `sUV_grid_OMUVBd_he5()` returns a data frame with two numeric variables, Longitude and Latitude, with two rows or an expanded grid depending on the argument passed to `expand`, while `sUV_date_OMUVBd_he5()` returns a vector of class Date, with file names as member names by default.

Note

The constraint on the consistency among all files to be read allows very fast reading into a single data frame. If the files differ in the set of variables, this function can be used to read the files individually into separate data frames. These data frames can later be row-bound together.

This function's performance is fast as long as there is enough RAM available to hold the data frame and the files are read from a reasonably fast SSD. The example data included in the package are global and one day. They are used in examples and automated tests. Function `sUV_read_OMUVBd_he5()` has also been tested by importing multiple files off-line as only one example file is included in the package due to these files' large size.

In files as downloaded the same whole set of variables is included. However, it is possible for users to edit the files. Thus, we do search for variables that are available. The grid could potentially be also subset, but files containing a subset of the grid are not supported (use the NetCDF4 files for such data).

References

Jari Hovila, Antti Arola, and Johanna Tamminen (2013), OMI/Aura Surface UVB Irradiance and Erythmal Dose Daily L3 Global Gridded 1.0 degree x 1.0 degree V3, NASA Goddard Space Flight Center, Goddard Earth Sciences Data and Information Services Center (GES DISC).

See Also

[sUV_read_OMUVBd_nc4\(\)](#) supporting the same Surface UV data, in NetCDF4 files, possibly as a subset of the grid and/or variables.

Examples

```
# find location of one example file
path.to.files <-
  system.file("extdata",
              package = "surfaceuv", mustWork = TRUE)

file.names <-
  list.files(path.to.files, pattern = "*\\\\.he5$", full.names = TRUE)

# available variables
sUV_vars_OMUVBd_he5(file.names)

# available grid
sUV_grid_OMUVBd_he5(file.names)
sUV_grid_OMUVBd_he5(file.names, expand = TRUE)

# decode date from file name
sUV_date_OMUVBd_he5(file.names)
sUV_date_OMUVBd_he5(file.names, use.names = FALSE)

# read all variables
helsinki_3days.tb <-
  sUV_read_OMUVBd_he5(file.names)
dim(helsinki_3days.tb)
```

```
summary(helsinki_3days.tb)

# read some variables
helsinki_UVI_3days.tb <-
  sUV_read_OMUVBd_he5(file.names, vars.to.read = "UVindex")
dim(helsinki_UVI_3days.tb)
summary(helsinki_UVI_3days.tb)
```

sUV_read_OMUVBd_nc4 *OMI/Aura Surface UV gridded subsets*

Description

Import **gridded** "Surface UV" data released by FMI/NASA from **NetCDF4** files downloaded from the NASA EARTHDATA server.

Usage

```
sUV_read_OMUVBd_nc4(
  files,
  vars.to.read = NULL,
  fill = NA_real_,
  verbose = interactive()
)

sUV_vars_OMUVBd_nc4(files, set.oper = "intersect")

sUV_grid_OMUVBd_nc4(files, expand = FALSE)

sUV_date_OMUVBd_nc4(files, use.names = length(files > 1))
```

Arguments

files	character A vector of file names, no other limitation in length than available memory to hold the data.
vars.to.read	character A vector of variable names. If NULL all the variables present in the first file are read.
fill	numeric The R value used to replace the fill value used in the file, which is retrieved from the file metadata, and also used to fill missing variables.
verbose	logical Flag indicating if progress, and time and size of the returned object should be printed.
set.oper	character One of "intersect", or "union".
expand	logical Flag indicating whether to return ranges or a full grid.
use.names	logical. Should names be added to the returned vector?

Details

Function `sUV_read_OMUVBd_nc4()` can be used to read the data stored in a file, either in full or selected variables. Query functions `sUV_vars_OMUVBd_nc4()`, `sUV_grid_OMUVBd_nc4()` and `sUV_date_OMUVBd_nc4()` extract the names of the variables, the range of the grid and the dates of measurements much more efficiently than by using `sUV_read_OMUVBd_nc4()`. The dates are decoded from the file names, expecting these to be those set by the data provider. The grid is expected to be identical in all files that are imported in a call to `sUV_read_OMUVBd_nc4()`, and grid subsetting is currently not supported. If not all the files named in the argument to `files` are accessible, an error is triggered early. If the files differ in the grid, an error is triggered when reading the first mismatching file. Missing variables named in `vars.to.read` if detected when reading the first file, are filled with the `fill` value, otherwise they trigger an error when an attempt is made to read them.

Value

Function `sUV_read_OMUVBd_nc4()` returns a data frame with columns named "Date", "Longitude", "Latitude", and the data variables with their original names. The data variables have their meta-data stored as R attributes. `sUV_vars_OMUVBd_nc4()` returns a character vector of variable names, `sUV_grid_OMUVBd_nc4()` returns a data frame with two numeric variables, Longitude and Latitude, with two rows or an expanded grid depending on the argument passed to `expand`, while `sUV_date_OMUVBd_nc4()` returns a, by default named, vector of class Date, with file names as names.

Note

The constraint on the consistency among all files to be read allows very fast reading into a single data frame. If the files differ in the grid or set of variables, this function can be used to read the files individually into separate data frames. These data frames can later be row-bound together.

This function's performance is fast as long as there is enough RAM available to hold the data frame and the files are read from a reasonably fast SSD. The example data included in the package are only for Spain and five summer days. They are used in examples and automated tests.

References

Jari Hovila, Antti Arola, and Johanna Tamminen (2013), OMI/Aura Surface UVB Irradiance and Erythral Dose Daily L3 Global Gridded 1.0 degree x 1.0 degree V3, NASA Goddard Space Flight Center, Goddard Earth Sciences Data and Information Services Center (GES DISC).

See Also

[sUV_read_OMUVBd_he5\(\)](#) supporting the same Surface UV data as stored in the original HDF5 files with a global geographic scope.

Examples

```
# find location of one example file
path.to.files <-
  system.file("extdata",
              package = "surfaceuv", mustWork = TRUE)
```

```

file.names <- list.files(path.to.files, pattern = "*.nc4$", full.names = TRUE)
one.file.name <- file.names[1]

# available variables
sUV_vars_OMUVBd_nc4(one.file.name)

# available grid
sUV_grid_OMUVBd_nc4(one.file.name)
sUV_grid_OMUVBd_nc4(one.file.name, expand = TRUE)

# decode date from file name
sUV_date_OMUVBd_nc4(one.file.name)
sUV_date_OMUVBd_nc4(one.file.name, use.names = FALSE)

# read all variables
midsummer_spain.tb <- sUV_read_OMUVBd_nc4(one.file.name)
dim(midsummer_spain.tb)
summary(midsummer_spain.tb)

# read only UVindex
midsummer_spain_daily.tb <-
  sUV_read_OMUVBd_nc4(one.file.name,
                      vars.to.read = "UVindex")
dim(midsummer_spain_daily.tb)
summary(midsummer_spain_daily.tb)

# read multiple files

sUV_vars_OMUVBd_nc4(file.names)

sUV_grid_OMUVBd_nc4(file.names)
sUV_grid_OMUVBd_nc4(file.names, expand = TRUE)

sUV_date_OMUVBd_nc4(file.names)
sUV_date_OMUVBd_nc4(file.names, use.names = FALSE)

summer_3days_spain.tb <- sUV_read_OMUVBd_nc4(file.names)
dim(summer_3days_spain.tb)
summary(summer_3days_spain.tb)

```

sUV_read_OUV_hdf5

Offline AC SAF gridded Surface UV

Description

Import **gridded** "Surface UV" data released by EUMETSAT AC SAF (Atmospheric Composition Monitoring) project from **HDF5** files downloaded from the FMI server.

Usage

```

sUV_read_OUV_hdf5(
  files,
  data.product = NULL,
  group.name = "GRID_PRODUCT",
  vars.to.read = NULL,
  fill = NA_real_,
  keep.QC = TRUE,
  verbose = interactive()
)

sUV_vars_OUV_hdf5(
  files,
  data.product = NULL,
  group.name = "GRID_PRODUCT",
  keep.QC = TRUE,
  set.oper = "intersect"
)

sUV_grid_OUV_hdf5(files, expand = FALSE)

sUV_date_OUV_hdf5(files, use.names = length(files > 1))

```

Arguments

files	character A vector of file names, no other limitation in length than available memory to hold the data.
data.product	character Currently only "Surface UV" supported.
group.name	character The name of the 'group' in the HDF5 files, or a regular expression for matching a single group name with grep() .
vars.to.read	character A vector of variable names. If NULL all the variables present in the first file are read.
fill	numeric The R value used to replace the fill value used in the file, which is retrieved from the file metadata, and also used to fill missing variables.
keep.QC	logical Add to the returned data frame or vector the quality control variable, always present in the files.
verbose	logical Flag indicating if progress, and time and size of the returned object should be printed.
set.oper	character One of "intersect", or "union".
expand	logical Flag indicating whether to return ranges or a full grid.
use.names	logical. Should names be added to the returned vector?

Details

Function `sUV_read_OUV_hdf5()` can be used to read the data stored in a file, either in full or selected variables. Query functions `sUV_vars_OUV_hdf5()`, `sUV_grid_OUV_hdf5()` and `sUV_date_OUV_hdf5()`

extract the names of the variables, the range of the grid and the dates of measurements much more efficiently than by using `sUV_read_OUV_hdf5()`. The dates are decoded from the file names, expecting these to be those set by the data provider. The grid is expected to be identical in all files that are imported in a call to `sUV_read_OUV_hdf5()`, and grid subsetting is currently not supported. If not all the files named in the argument to `files` are accessible, an error is triggered early. If the files differ in the grid, an error is triggered when reading the first mismatching file. Missing variables named in `vars.to.read` if detected when reading the first file, are filled with the `fill` value, otherwise they trigger an error when an attempt is made to read them.

Value

Function `sUV_read_OUV_hdf5()` returns a data frame with columns named "Date", "Longitude", "Latitude", the data variables with their original names, and "QualityFlags". The data variables have their metadata stored as R attributes. `sUV_vars_OUV_hdf5()` returns a character vector of variable names, `sUV_grid_OUV_hdf5()` returns a data frame with two numeric variables, Longitude and Latitude, with two rows or an expanded grid depending on the argument passed to `expand`, while `sUV_date_OUV_hdf5()` returns a named vector of class Date, with file names as names.

Note

The constraint on the consistency among all files to be read allows very fast reading into a single data frame. If the files differ in the grid or set of variables, this function can be used to read the files individually into separate data frames. These data frames can later be row-bound together.

Variable `QualityFlags` is encoded as 64 bit integers in the HDF5 file and read as a double. R package 'bit64' can be used to print these values as 64 bit integers.

When requesting the data from the EUMETSAT AC SAF FMI server at <https://acsaf.org/> it is possible to select the range of latitudes and longitudes and the variables to be included in the file. This is more efficient than doing the selection after importing the data into R. The data are returned as a .zip compressed file containing one .HDF5 file for each day in the range of dates selected. For world coverage each of these files can be as large as 10 MB in size depending on how many variables they contain. These files in HDF5 format are binary files so the size in RAM of a data.frame object containing one-year of data can be a few 10's of GB.

This function's performance is fast as long as there is enough RAM available to hold the data frame and the files are read from a reasonably fast SSD. The example data included in the package are only for Spain and five summer days. They are used in examples and automated tests. Function `sUV_read_OUV_hdf5()` has been also tested by importing one-year's worth of data with worldwide coverage on a PC with 64GB RAM.

References

Kujanpää, J. (2019) *PRODUCT USER MANUAL Offline UV Products v2 (IDs: O3M-450 - O3M-464) and Data Record R1 (IDs: O3M-138 - O3M-152)*. Ref. SAF/AC/FMI/PUM/001. 18 pp. EUMETSAT AC SAF.

See Also

`sUV_read_OUV_txt()` supporting the same Surface UV data stored in text files as single-location time series.

Examples

```
# find location of one example file
one.file.name <-
  system.file("extdata", "03MOUV_L3_20240621_v02p02.HDF5",
             package = "surfaceuv", mustWork = TRUE)

# available variables
sUV_vars_OUV_hdf5(one.file.name)

# available grid
sUV_grid_OUV_hdf5(one.file.name)

# decode date from file name
sUV_date_OUV_hdf5(one.file.name)
sUV_date_OUV_hdf5(one.file.name, use.names = FALSE)

# read all variables
midsummer_spain.tb <- sUV_read_OUV_hdf5(one.file.name)
dim(midsummer_spain.tb)
summary(midsummer_spain.tb)

# read two variables
midsummer_spain_daily.tb <-
  sUV_read_OUV_hdf5(one.file.name,
                   vars.to.read = c("DailyDoseUva", "DailyDoseUvb"))
dim(midsummer_spain_daily.tb)
summary(midsummer_spain_daily.tb)

# find location of three example files
three.file.names <-
  system.file("extdata",
             c("03MOUV_L3_20240621_v02p02.HDF5",
               "03MOUV_L3_20240622_v02p02.HDF5",
               "03MOUV_L3_20240623_v02p02.HDF5"),
             package = "surfaceuv", mustWork = TRUE)

sUV_date_OUV_hdf5(three.file.names)

summer_3days_spain.tb <- sUV_read_OUV_hdf5(three.file.names)
dim(summer_3days_spain.tb)
summary(summer_3days_spain.tb)
```

sUV_read_OUV_txt

Offline AC SAF Surface UV time series

Description

Import **time series** "Surface UV" data released by EUMETSAT AC SAF (Atmospheric Composition Monitoring) project from **text** files downloaded from the FMI server.

Usage

```

sUV_read_OUV_txt(
  files,
  vars.to.read = NULL,
  add.geo = length(files) > 1,
  keep.QC = TRUE,
  verbose = interactive()
)

sUV_vars_OUV_txt(files, keep.QC = TRUE, set.oper = "intersect")

sUV_grid_OUV_txt(files, use.names = length(files) > 1)

```

Arguments

<code>files</code>	character A vector of file names, no other limitation in length than available memory to hold the data.
<code>vars.to.read</code>	character A vector of variable names. If NULL all the variables present in the first file are read.
<code>add.geo</code>	logical Add columns Longitude and Latitude to returned data frame.
<code>keep.QC</code>	logical Add to the returned data frame or vector the quality control variables, always present in the files.
<code>verbose</code>	logical Flag indicating if progress, and time and size of the returned object should be printed.
<code>set.oper</code>	character One of "intersect", or "union".
<code>use.names</code>	logical. Should row names be added to the returned data frame?

Details

All information is in the files, including dates, and no information is decoded from file names, that users will most likely want to rename. Each file corresponds to a single geographic location. If not all the files named in the argument to `files` are accessible, an error is triggered early. If the files differ in the coordinates, an error is triggered when reading the first mismatching file if coordinates are not being added to the data frame. Missing variables named in `vars.to.read` are currently ignored.

Data from multiple files are concatenated. By default, the geographic coordinates are added in such a case.

Value

`sUV_read_OUV_txt()` returns a data frame with columns named "Date", "Longitude", "Latitude", and the data variables with their original names (with no units). The data variables have no metadata stored as R attributes. When reading multiple files, by default the format is similar to that from function `sUV_read_OUV_hdf5()`. Column names are the same but column order can differ. File headers are saved as a list in R attribute `file.headers`. `sUV_vars_OUV_txt()` returns a character vector of variable names, and `sUV_grid_OUV_txt()` a dataframe with two numeric variables, Longitude and Latitude, and a single row.

Note

When requesting the data from the EUMETSAT AC SAF FMI server at <https://acsaf.org/> it is possible to select the variables to be included in the file, the period and the geographic coordinates of a single location. The data are returned as a .zip compressed file containing one text file with one row for each day in the range of dates selected. These files are fairly small.

This function's performance is not optimized for speed as these single location files are rather small. The example time series data included in the package are for one summer in Helsinki, Finland.

References

Kujanpää, J. (2019) *PRODUCT USER MANUAL Offline UV Products v2 (IDs: O3M-450 - O3M-464) and Data Record R1 (IDs: O3M-138 - O3M-152)*. Ref. SAF/AC/FMI/PUM/001. 18 pp. EUMETSAT AC SAF.

See Also

[sUV_read_OUV_hdf5\(\)](#) supporting the same Surface UV data stored in a gridded format.

Examples

```
# find location of one example file
one.file.name <-
  system.file("extdata", "AC_SAF-Viikki-FI-6mas1.txt",
    package = "surfaceuv", mustWork = TRUE)

# Available variables
sUV_vars_OUV_txt(one.file.name)
sUV_vars_OUV_txt(one.file.name, keep.QC = FALSE)

# Grid point coordinates
sUV_grid_OUV_txt(one.file.name)

# read all variables
summer_viikki.tb <-
  sUV_read_OUV_txt(one.file.name)
dim(summer_viikki.tb)
colnames(summer_viikki.tb)
str(sapply(summer_viikki.tb, class))
summary(summer_viikki.tb)
attr(summer_viikki.tb, "file.headers")

# read all data variables
summer_viikki_QCf.tb <-
  sUV_read_OUV_txt(one.file.name, keep.QC = FALSE)
dim(summer_viikki_QCf.tb)
summary(summer_viikki_QCf.tb)

# read all data variables including geographic coordinates
summer_viikki_geo.tb <-
  sUV_read_OUV_txt(one.file.name, keep.QC = FALSE, add.geo = TRUE)
dim(summer_viikki_geo.tb)
```

```
summary(summer_viikki_geo.tb)

# read two variables
summer_viikki_2.tb <-
  sUV_read_OUV_txt(one.file.name,
                    vars.to.read = c("DailyDoseUva", "DailyDoseUvb"))
dim(summer_viikki_2.tb)
summary(summer_viikki_2.tb)
```

Index

check_files, [2](#)

dist2target (nearest_point), [4](#)

grep(), [6](#), [11](#)

grepl(), [2](#)

nearest_ccoord (nearest_corners), [3](#)

nearest_corners, [3](#)

nearest_pcoord (nearest_point), [4](#)

nearest_point, [4](#)

nearest_rcoord (nearest_corners), [3](#)

nearest_region (nearest_corners), [3](#)

sUV_date_OMUVBd_he5
 (sUV_read_OMUVBd_he5), [5](#)

sUV_date_OMUVBd_nc4
 (sUV_read_OMUVBd_nc4), [8](#)

sUV_date_OUV_hdf5 (sUV_read_OUV_hdf5),
 [10](#)

sUV_grid_OMUVBd_he5
 (sUV_read_OMUVBd_he5), [5](#)

sUV_grid_OMUVBd_nc4
 (sUV_read_OMUVBd_nc4), [8](#)

sUV_grid_OUV_hdf5 (sUV_read_OUV_hdf5),
 [10](#)

sUV_grid_OUV_txt (sUV_read_OUV_txt), [13](#)

sUV_read_OMUVBd_he5, [5](#)

sUV_read_OMUVBd_he5(), [9](#)

sUV_read_OMUVBd_nc4, [8](#)

sUV_read_OMUVBd_nc4(), [7](#)

sUV_read_OUV_hdf5, [10](#)

sUV_read_OUV_hdf5(), [15](#)

sUV_read_OUV_txt, [13](#)

sUV_read_OUV_txt(), [12](#)

sUV_vars_OMUVBd_he5
 (sUV_read_OMUVBd_he5), [5](#)

sUV_vars_OMUVBd_nc4
 (sUV_read_OMUVBd_nc4), [8](#)

sUV_vars_OUV_hdf5 (sUV_read_OUV_hdf5),
 [10](#)

sUV_vars_OUV_txt (sUV_read_OUV_txt), [13](#)