Package 'gridclimind'

July 9, 2020

Version 1.0.0

Date 2017-04-14

Title Functions to compute CLIMIND indices over a NetCDF grid

Maintainer ECAD <eca@knmi.nl>

Depends R (>= 3.0),
PCICt (>= 0.5-4)

Imports ncdf4 (>= 1.10),
climind (>= 1.0.0),
ncdf4.helpers (>= 0.3-3),
snow (>= 0.3-13),
udunits2 (>= 0.6),
functional (>= 0.4),
proj4 (>= 1.0-8),
scPDSI,
PETr

Suggests RUnit

Description This package contains functions which can be used to compute CLIMDEX indices using NetCDF input files, writing to NetCDF output files. Code allows for parallel computation of indices using either a SOCK or MPI cluster.

License GPL-3

Encoding UTF-8

URL http://www.r-project.org

RoxygenNote 7.0.2

R topics documented:

`compute.climdex.indices`

Compute Climdex indices using provided data.

Description

Compute Climdex indices using provided data.

Usage

```
compute.climdex.indices(in.dat, cdx.funcs, ts, base.range, metadata.config)
```

Arguments

`in.dat`	The input data to compute indices on.
`cdx.funcs`	The functions to be applied to the data, as created by `get.climdex.functions`.
`ts`	The associated time data, as created by `nc.get.time.series`.
`base.range`	The base range; a vector of two numeric years.
`metadata.config`	object containing the relevant metadata configuration information.

Details

Given the provided data and functions, compute the Climdex indices defined by the functions.

Value

A list of data for each index.

Examples

```
library(climind)

## Prepare input data
in.dat <- list(tmax=ec.1018935.tmax$MAX_TEMP)
cdx.funcs <- get.climdex.functions(get.climdex.variable.list(names(in.dat)))
in.dat$northern.hemisphere <- TRUE
ts <- as.PCICt(do.call(paste, ec.1018935.tmax[,c("year", "jday")]),
               format="%Y %j", cal="gregorian")

## Compute indices
res <- compute.climdex.indices(in.dat, cdx.funcs, ts, c(1981, 1990), FALSE)
```

---

compute.indices.for.stripe

Compute Climdex indices for a subset / stripe

Description

Compute Climdex indices for a subset / stripe

Usage

```
compute.indices.for.stripe(
  subset,
  metadata.config,
  cdx.funcs,
  ts,
  base.range,
  dim.axes,
  v.f.idx,
  variable.name.map,
  src.units,
  t.f.idx,
  thresholds.name.map,
  projection = NULL,
  f,
  thresholds.netcdf
)
```

Arguments

subset	The subset to use.
cdx.funcs	The functions to be applied to the data, as created by get.climdex.functions.
ts	The associated time data, as created by nc.get.time.series.
base.range	The base range; a vector of two numeric years.
dim.axes	The dimension axes for the input data.
v.f.idx	A mapping from variables to files, as created by get.var.file.idx.
variable.name.map	A mapping from standardized names (tmax, tmin, prec) to NetCDF variable names.
src.units	The source units to convert data from.
t.f.idx	A mapping from threshold variables to threshold files, as created by get.var.file.idx.
thresholds.name.map	A mapping from standardized names (tx10thresh, tn90thresh, etc) to NetCDF variable names.
projection	A proj4 string representing the projection the data is in.
f	A list of objects of type ncdf4, consisting of the open input files. If missing, will be pulled from the global namespace.
thresholds.netcdf	A list of objects of type ncdf4, consisting of the open threshold files. If missing, will be pulled from the global namespace.
dest.units	The destination units to convert to.

Details

Given a subset, a set of Climdex functions (as created by get.climdex.functions), and ancillary data, load and convert data, create a climdexInput object for each point, run all of the functions in cdx.funcs on that data, and return the result.

Note

This function relies on an object named 'f' and containing the opened NetCDF files being part of the global namespace.

Examples

```
## Define mappings and filenames.
author.data <- list(institution="Looney Bin", institution_id="LBC")
input.files <- c("pr_NAM44_CanRCM4_ERAINT_r1i1p1_1989-2009.nc")
variable.name.map <- c(tmax="tasmax", tmin="tasmin", prec="pr")

## Open files, etc.
cdx.funcs <- get.climdex.functions(get.climdex.variable.list("tmax"))
f <- lapply(input.files, ncdf4::nc_open)
f.meta <- create.file.metadata(f, variable.name.map)
```

```
climdex.var.list <- get.climdex.variable.list(names(f.meta$v.f.idx), "all", NULL)
cdx.meta <- get.climdex.variable.metadata(climdex.var.list, input.files[1])

## Compute indices for stripe
cdx <- compute.indices.for.stripe(list(Y=1), cdx.funcs, f.meta$ts, c(1981, 1990), f.meta$dim.axes,
                  f.meta$v.f.idx, variable.name.map, f.meta$src.units, f.meta$dest.units,
                           t.f.idx, NULL, f=f, thresholds.netcdf=NULL)
```

`create.climdex.eobs.filenames` *Creates a list of CMIP5-compliant filenames reflecting the input data.*

Description

Creates a list of CMIP5-compliant filenames reflecting the input data.

Usage

```
create.climdex.eobs.filenames(fn.split, vars.list)
```

Arguments

`fn.split`	A vector containing named components, as created by `get.split.filename.cmip5`.
`vars.list`	A vector containing names of variables, as created by `get.climdex.variable.list`.

Details

This function takes a split filename (as created by `get.split.filename.cmip5`) and a list of variables and creates corresponding filenames for the given variables.

Value

A vector containing filenames corresponding to the variables and filename bits supplied.

Examples

```
## Not run:
library(ncdf4.helpers)
## Split out filename bits for use below...
fn <- "pr_day_BCCAQ+ANUSPLIN300+MRI-CGCM3_historical+rcp85_r1i1p1_19500101-21001231.nc"
fn.split <- get.split.filename.cmip5(fn)

## Create filenames with time data and variable appropriately replaced.
filenames <- create.climdex.cmip5.filenames(fn.split, c("rx5dayETCCDI_mon", "tn90pETCCDI_yr"))

## End(Not run)
```

```
create.climdex.cmip5.filenames <- function(fn.split, vars.list) {
time.res <- c("yr", "mon")[grepl("_mon$", vars.list) + 1]
time.range <- substr(fn.split[c('tstart', 'tend')], 1, 4)

paste(paste(vars.list, fn.split['model'], fn.split['emissions'], fn.split['run'], sapply(time.res, function(x) {
}
```

`create.file.metadata` *Retrieve metadata about NetCDF-format files.*

Description

Retrieve metadata about NetCDF-format files.

Usage

```
create.file.metadata(f, variable.name.map)
```

Arguments

`f`	The list of NetCDF files.
`variable.name.map`	A named character vector mapping standard variable names (tmax, tmin, prec) to NetCDF variable names.

Details

Given a list of NetCDF files and a mapping from standard variable names (tmax, tmin, prec) to NetCDF variable names, retrieve a set of standardized metadata.

Value

A list containing time data (ts), dimension sizes (dim.size), dimension axes (dim.axes), source units (src.units), destination units (dest.units), a mapping from variables to files (v.f.idx), and a projection, if available.

Examples

```
## Not run:
## Get metadata about a single input file.
input.files <- c("pr_NAM44_CanRCM4_ERAINT_r1i1p1_1989-2009.nc")
f <- lapply(input.files, ncdf4::nc_open)
f.meta <- create.file.metadata(f, variable.name.map)

## End(Not run)
```

`create.indices.from.files` *Create Climdex indices from NetCDF input files.*

Description

Create Climdex indices from NetCDF input files.

Usage

```
create.indices.from.files(
  input.files,
  out.dir,
  output.filename.template,
  author.data,
  climdex.vars.subset = NULL,
 climdex.time.resolution = c("all", "annual", "monthly", "seasonal", "halfyear"),
  axis.to.split.on = "Y",
  fclimdex.compatible = TRUE,
  base.range = c(1961, 1990),
  parallel = 4,
  verbose = FALSE,
  thresholds.files = NULL,
  max.vals.millions,
  cluster.type = "SOCK"
)
```

Arguments

`input.files` A list of filenames of NetCDF files to be used as input. A NetCDF file may contain one or more variables.

`out.dir` The directory to put the output files in.

`output.filename.template` The output filename to be used as a template, which must follow the CMIP5 file naming conventions.

`author.data` Data describing the author; a character vector with 0 or more of the following named values:

institution The institution generating the data.

institution_id An abbreviation for the institution generating the data.

indices_archive The URL the data is published at, if applicable.

contact The email address or contact info for the author.

references What to reference when citing this work.

`climdex.vars.subset` A character vector of lower-case names of Climdex indices to calculate (eg: tr, fd, rx5day). See the list of 27 indices in the References section.

`climdex.time.resolution`
: The time resolution to compute indices at; one of "all" (both monthly and annual), "annual" (only annual), or "monthly" (only monthly).

`axis.to.split.on`
: The axis to split up the data on for parallel / incremental processing.

`fclimdex.compatible`
: Whether the thresholds should be created to match fclimdex thresholds; affects padding at the ends of the base period.

`base.range`
: Vector of two numeric years specifying the start and end years.

`parallel`
: The number of parallel processing threads, or FALSE if no parallel processing is desired.

`verbose`
: Whether to be chatty.

`thresholds.files`
: A character vector of files containing thresholds to be used.

`max.vals.millions`
: The number of data values to process at one time (length of time dim * number of values * number of variables).

`cluster.type`
: The cluster type, as used by the `snow` library.

Details

This function computes Climdex indices from NetCDF input files, writing out one file per variable named like the `template.filename`, which must follow the CMIP5 file naming conventions (this is a deficiency which will be corrected in later versions).

The indices to be calculated can be specified; if not, they will be determined by data availability. Thresholds can be supplied (via `thresholds.files`) or, if there is data within the base period, calculated and used as part of the process. Note that in-base thresholds are separate from out-of-base thresholds; this is covered in more detail in the help for the `climind` package.

The metadata is stored in JSON files that are included with the pacakge. Right now, the metadata relevant to EOBS is used by default. To switch to another set of metadata, use the `metadata.id` global option:

```
options(metadata.id = 'eobs')
```

Note that currently only EOBS metadata is available (`metadata.id = 'eobs'`).

Note

NetCDF input files may contain one or more variables, named as per `variable.name.map` in the json config file. The code will search the files for the named variables. The same is true of thresholds files; one file may be supplied, or multiple files may be supplied, via the `thresholds.files` argument; and the name mapping may be supplied via the `thresholds.name.map` argument.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

Examples

```
## Not run:
## Prepare input data and calculate indices for a single file
## with a single thread (no parallelism).
input.files <- c("pr_NAM44_CanRCM4_ERAINT_r1i1p1_1989-2009.nc")
author.data <- list(institution="Looney Bin", institution_id="LBC")
create.indices.from.files(input.files, "out_dir/", input.files[1], author.data,
                          base.range=c(1991, 2000), parallel=FALSE)

## Prepare input data and calculate indices for two files
## in parallel given thresholds.
input.files <- c("pr_NAM44_CanRCM4_ERAINT_r1i1p1_1989-2009.nc",
                 "tasmax_NAM44_CanRCM4_ERAINT_r1i1p1_1989-2009.nc")
author.data <- list(institution="Looney Bin", institution_id="LBC")
create.indices.from.files(input.files, "out_dir/", input.files[1], author.data,
                      base.range=c(1991, 2000), parallel=8, thresholds.files="thresh.nc")

## End(Not run)
```

---

create.ncdf.output.files

Creates output files for Climdex variables.

Description

Creates output files for Climdex variables.

Usage

```
create.ncdf.output.files(
  cdx.dat,
  f,
  v.f.idx,
  variable.name.map,
  ts,
  time.origin,
  base.range,
  out.dir,
  author.data,
  metadata.config
)
```

Arguments

cdx.dat	The variable description data, as created by get.climdex.variable.metadata.
f	The file(s) being used as input.

v.f.idx	A mapping from variables to files, as created by get.var.file.idx.
variable.name.map	A mapping from standardized names (tmax, tmin, prec) to NetCDF variable names.
ts	The associated time data, as created by nc.get.time.series.
time.origin	The time origin, as specified in the source NetCDF file(s).
base.range	The base range; a vector of two numeric years.
out.dir	The output directory name.
author.data	A vector containing named elements describing the author; see create.indices.from.files.

Details

This function creates a set of output files for the set of variable parameters passed in cdx.dat, as created by get.climdex.variable.metadata. It copies metadata from input files as appropriate and adds new metadata as required.

Value

A list of objects of type ncdf4.

Examples

```
## Establish basic inputs.
author.data <- list(institution="Looney Bin", institution_id="LBC")
input.files <- c("pr_NAM44_CanRCM4_ERAINT_r1i1p1_1989-2009.nc")

## Prepare derived inputs.
f <- lapply(input.files, ncdf4::nc_open)
variable.name.map <- c(tmax="tasmax", tmin="tasmin", prec="pr")
f.meta <- create.file.metadata(f, variable.name.map)
climdex.var.list <- get.climdex.variable.list(names(f.meta$v.f.idx), "all", NULL)
cdx.meta <- get.climdex.variable.metadata(climdex.var.list, input.files[1])

## Create output files
cdx.ncfile <- create.ncdf.output.files(cdx.meta, f, f.meta$v.f.idx, variable.name.map,
                                       f.meta$ts, get.time.origin(f, f.meta$dim.axes),
                                       c(1981,1990), "/foo", author.data)
```

---

create.pet.file *Creates pet output file.*

Description

Creates pet output file.

Usage

```
create.pet.file(
  pet.file,
  f,
  ts,
  v.f.idx,
  variable.name.map,
  dim.size,
  dim.axes,
  pet.dat,
  author.data
)
```

Arguments

pet.file	The filename to be used for the pet file.
f	The file(s) being used as sources for metadata.
ts	The associated time data, as created by nc.get.time.series.
v.f.idx	A mapping from variables to files, as created by get.var.file.idx.
variable.name.map	A mapping from standardized names (tmax, tmin, etc) to NetCDF variable names.
dim.size	Dimension sizes for the input.
dim.axes	Dimension axes for the input.
pet.dat	pet metadata
author.data	A vector containing named elements describing the author; see create.indices.from.files.

Details

This function creates a file suitable for outputting pet to.

Value

An object of class ncdf4.

---

create.pet.makkink.from.files

Create PET

Description

Create PET

Usage

```
create.pet.makkink.from.files(
  input.files,
  output.file,
  author.data,
  climdex.time.resolution = "daily",
  axis.to.split.on = "Y",
  parallel = 4,
  verbose = FALSE,
  cluster.type = "SOCK"
)
```

Arguments

input.files	A list of filenames of NetCDF files to be used as input. A NetCDF file may contain one or more variables.
output.file	The name of the file to be created.
author.data	A vector containing named elements describing the author; see create.indices.from.files.
axis.to.split.on	The axis to split up the data on for parallel / incremental processing.
parallel	The number of parallel processing threads, or FALSE if no parallel processing is desired.
verbose	Whether to be chatty.
cluster.type	The cluster type, as used by the snow library.

Details

The purpose of this function is to compute PET using the Makkink method on the data supplied, saving them to the file specified.

The metadata is stored in JSON files that are included with the pacakge. Right now, the metadata relevant to EOBS is used by default. To switch to another set of metadata, use the metadata.id global option:

```
options(metadata.id = 'eobs')
```

Note that currently only EOBS metadata is available (metadata.id = 'eobs').

Note

NetCDF input files may contain one or more variables, named as per variable.name.map (read from config json file). The code will search the files for the named variables.

Examples

```
## Not run:
## Prepare input data and calculate pet for file.
input.files <- c(paste0(in.dir,"tn_0.50deg_regular_1971-2016.nc"),
paste0(in.dir,"tx_0.50deg_regular_1971-2016.nc"),
```

```
paste0(in.dir,"rs_0.50deg_regular_1971-2016.nc"))
author.data <- list(institution="Looney Bin", institution_id="LBC")
create.pet.from.files(input.files, out.file, input.files[1], author.data,  parallel=3)

## End(Not run)
```

`create.pet.penman.from.files` *Create Climdex PET*

Description

Create Climdex PET

Usage

```
create.pet.penman.from.files(
  input.files,
  output.file,
  author.data,
  climdex.time.resolution = "daily",
  axis.to.split.on = "Y",
  parallel = 4,
  verbose = FALSE,
  cluster.type = "SOCK"
)
```

Arguments

`input.files`	A list of filenames of NetCDF files to be used as input. A NetCDF file may contain one or more variables.
`output.file`	The name of the file to be created.
`author.data`	A vector containing named elements describing the author; see `create.indices.from.files`.
`axis.to.split.on`	The axis to split up the data on for parallel / incremental processing.
`parallel`	The number of parallel processing threads, or FALSE if no parallel processing is desired.
`verbose`	Whether to be chatty.
`cluster.type`	The cluster type, as used by the snow library.

Details

The purpose of this function is to compute PET on the data supplied, saving them to the file specified.

The metadata is stored in JSON files that are included with the pacakge. Right now, the metadata relevant to EOBS is used by default. To switch to another set of metadata, use the `metadata.id` global option:

```
options(metadata.id = 'eobs')
```

Note that currently only EOBS metadata is available (`metadata.id = 'eobs'`).

Note

NetCDF input files may contain one or more variables, named as per `variable.name.map` (read from config json file). The code will search the files for the named variables.

Examples

```
## Not run:
## Prepare input data and calculate pet for file.
input.files <- c(paste0(in.dir,"tn_0.50deg_regular_1971-2016.nc"),
paste0(in.dir,"tx_0.50deg_regular_1971-2016.nc"),
paste0(in.dir,"rh_0.50deg_regular_1971-2016.nc"),
paste0(in.dir,"ws_0.50deg_regular_1971-2016.nc"),
paste0(in.dir,"ss_0.50deg_regular_1971-2016.nc"))
author.data <- list(institution="Looney Bin", institution_id="LBC")
create.pet.from.files(input.files, out.file, input.files[1], author.data,  parallel=3)

## End(Not run)
```

`create.pet.priestly.taylor.from.files`
Create PET

Description

Create PET

Usage

```
create.pet.priestly.taylor.from.files(
  input.files,
  output.file,
  author.data,
  climdex.time.resolution = "daily",
  axis.to.split.on = "Y",
  parallel = 4,
  verbose = FALSE,
  cluster.type = "SOCK"
)
```

Arguments

input.files	A list of filenames of NetCDF files to be used as input. A NetCDF file may contain one or more variables.
output.file	The name of the file to be created.
author.data	A vector containing named elements describing the author; see create.indices.from.files.
axis.to.split.on	The axis to split up the data on for parallel / incremental processing.
parallel	The number of parallel processing threads, or FALSE if no parallel processing is desired.
verbose	Whether to be chatty.
cluster.type	The cluster type, as used by the snow library.

Details

The purpose of this function is to compute PET on the data supplied, saving them to the file specified.

The metadata is stored in JSON files that are included with the pacakge. Right now, the metadata relevant to EOBS is used by default. To switch to another set of metadata, use the metadata.id global option:

```
options(metadata.id = 'eobs')
```

Note that currently only EOBS metadata is available (metadata.id = 'eobs').

Note

NetCDF input files may contain one or more variables, named as per variable.name.map (read from config json file). The code will search the files for the named variables.

Examples

```
## Not run:
## Prepare input data and calculate pet for file.
input.files <- c(paste0(in.dir,"tn_0.50deg_regular_1971-2016.nc"),
paste0(in.dir,"tx_0.50deg_regular_1971-2016.nc"),
paste0(in.dir,"rh_0.50deg_regular_1971-2016.nc"),
paste0(in.dir,"rs_0.50deg_regular_1971-2016.nc"))
author.data <- list(institution="Looney Bin", institution_id="LBC")
create.pet.from.files(input.files, out.file, input.files[1], author.data,  parallel=3)

## End(Not run)
```

`create.scPDSI.file` *Creates scPDSI output file.*

Description

Creates scPDSI output file.

Usage

```
create.scPDSI.file(
  scPDSI.file,
  f,
  ts,
  v.f.idx,
  variable.name.map,
  dim.size,
  dim.axes,
  scPDSI.dat,
  author.data
)
```

Arguments

`scPDSI.file`	The filename to be used for the scPDSI file.
`f`	The file(s) being used as sources for metadata.
`ts`	The associated time data, as created by `nc.get.time.series`.
`v.f.idx`	A mapping from variables to files, as created by `get.var.file.idx`.
`variable.name.map`	A mapping from standardized names (tmax, tmin, etc) to NetCDF variable names.
`dim.size`	Dimension sizes for the input.
`dim.axes`	Dimension axes for the input.
`scPDSI.dat`	scPDSI metadata
`author.data`	A vector containing named elements describing the author; see `create.indices.from.files`.

Details

This function creates a file suitable for outputting scPDSI to.

Value

An object of class `ncdf4`.

`create.scPDSI.from.files` *Create scPDSI*

Description

Create scPDSI

Usage

```
create.scPDSI.from.files(
  input.files,
  output.file,
  author.data,
  climdex.time.resolution = "daily",
  axis.to.split.on = "Y",
  parallel = 4,
  verbose = FALSE,
  cluster.type = "SOCK",
  start = NULL,
  end = NULL,
  cal_start = NULL,
  cal_end = NULL
)
```

Arguments

`input.files`	A list of filenames of NetCDF files to be used as input. A NetCDF file may contain one or more variables.
`output.file`	The name of the file to be created.
`author.data`	A vector containing named elements describing the author; see `create.indices.from.files`.
`axis.to.split.on`	The axis to split up the data on for parallel / incremental processing.
`parallel`	The number of parallel processing threads, or FALSE if no parallel processing is desired.
`verbose`	Whether to be chatty.
`cluster.type`	The cluster type, as used by the `snow` library.

Details

The purpose of this function is to compute scPDSI on the data supplied, saving them to the file specified.

The metadata is stored in JSON files that are included with the pacakge. Right now, the metadata relevant to EOBS is used by default. To switch to another set of metadata, use the `metadata.id` global option:

```
options(metadata.id = 'eobs')
```

Note that currently only EOBS metadata is available (`metadata.id = 'eobs'`).

Note

NetCDF input files may contain one or more variables, named as per `variable.name.map` (read from config json file). The code will search the files for the named variables.

Examples

```
## Not run:
## Prepare input data and calculate scPDSI for file.
input.files <- c(paste0(in.dir,"rr_0.50deg_regular_1971-2016.nc"),paste0(in.dir,"pet_0.50deg_regular_1971-2016.
author.data <- list(institution="Looney Bin", institution_id="LBC")
create.scPDSI.from.files(input.files, out.file, input.files[1], author.data,  parallel=3)

## End(Not run)
```

---

create.thresholds.file

Creates Climdex thresholds output file.

Description

Creates Climdex thresholds output file.

Usage

```
create.thresholds.file(
  thresholds.file,
  f,
  ts,
  v.f.idx,
  variable.name.map,
  base.range,
  dim.size,
  dim.axes,
  threshold.dat,
  author.data
)
```

Arguments

`thresholds.file`	The filename to be used for the thresholds file.
`f`	The file(s) being used as sources for metadata.

ts	The associated time data, as created by nc.get.time.series.
v.f.idx	A mapping from variables to files, as created by get.var.file.idx.
variable.name.map	A mapping from standardized names (tmax, tmin, prec) to NetCDF variable names.
base.range	The base range; a vector of two numeric years.
dim.size	Dimension sizes for the input.
dim.axes	Dimension axes for the input.
threshold.dat	Threshold metadata, as provided by get.thresholds.metadata.
author.data	A vector containing named elements describing the author; see create.indices.from.files.

Details

This function creates a file suitable for outputting thresholds to, with all variables that can be created with the input data present in the file.

Value

An object of class ncdf4.

Examples

```
## Establish basic inputs.
author.data <- list(institution="Looney Bin", institution_id="LBC")
input.files <- c("pr_NAM44_CanRCM4_ERAINT_r1i1p1_1989-2009.nc")

## Prepare derived inputs.
f <- lapply(input.files, ncdf4::nc_open)
variable.name.map <- c(tmax="tasmax", tmin="tasmin", prec="pr")
f.meta <- create.file.metadata(f, variable.name.map)
threshold.dat <- get.thresholds.metadata(names(f.meta$v.f.idx))

## Create output file
thresh.file <- create.thresholds.file("thresh.nc", f, f.meta$ts, f.meta$v.f.idx, variable.name.map,
                                      c(1981,1990), f.meta$dim.size, f.meta$dim.axes,
                                      threshold.dat, author.data)
```

---

create.thresholds.from.file

Create Climdex thresholds used for computing threshold-based indices

---

Description

Create Climdex thresholds used for computing threshold-based indices

Usage

```
create.thresholds.from.file(
  input.files,
  output.file,
  author.data,
  axis.to.split.on = "Y",
  base.range = c(1961, 1990),
  parallel = 4,
  verbose = FALSE,
  max.vals.millions,
  cluster.type = "SOCK"
)
```

Arguments

input.files	A list of filenames of NetCDF files to be used as input. A NetCDF file may contain one or more variables.
output.file	The name of the file to be created.
author.data	A vector containing named elements describing the author; see create.indices.from.files.
axis.to.split.on	The axis to split up the data on for parallel / incremental processing.
base.range	Vector of two numeric years specifying the start and end years.
parallel	The number of parallel processing threads, or FALSE if no parallel processing is desired.
verbose	Whether to be chatty.
max.vals.millions	The number of data values to process at one time (length of time dim * number of values * number of variables).
cluster.type	The cluster type, as used by the snow library.

Details

For many applications, one may want to compute thresholds on one data set, then apply them to another. This is usually the case when comparing GCM (Global Climate Model) results for future time periods to either historical reanalysis data or historical / pre-industrial control runs from models. The purpose of this function is to compute these thresholds on the data supplied, saving them to the file specified. Then these thresholds can be used with create.indices.from.files to compute indices using the thresholds computed using this code.

The metadata is stored in JSON files that are included with the pacakge. Right now, the metadata relevant to EOBS is used by default. To switch to another set of metadata, use the metadata.id global option:

```
options(metadata.id = 'eobs')
```

Note that currently only EOBS metadata is available (metadata.id = 'eobs').

Note

NetCDF input files may contain one or more variables, named as per `variable.name.map` (read from config json file). The code will search the files for the named variables.

Examples

```
## Not run:
## Prepare input data and calculate thresholds for file.
input.files <- c("pr_NAM44_CanRCM4_ERAINT_r1i1p1_1989-2009.nc")
author.data <- list(institution="Looney Bin", institution_id="LBC")
create.thresholds.from.file(input.files, "thresh.nc", author.data,
                            base.range=c(1991, 2000), parallel=FALSE)

## End(Not run)
```

`curry_in_subset_for_huglin` *A curry function used only for the Huglin Index This function curries the cdx.funcs so that the current subset (cur_sub) is retrieved with the cdx function It is placed inside compute.indices.for.stripe*

Description

A curry function used only for the Huglin Index This function curries the cdx.funcs so that the current subset (cur_sub) is retrieved with the cdx function It is placed inside compute.indices.for.stripe

Usage

```
curry_in_subset_for_huglin(cdx.funcs, cur_sub)
```

`flatten.dims` *Flatten the X and Y dimensions down to a space dimension.*

Description

Flatten the X and Y dimensions down to a space dimension.

Usage

```
flatten.dims(dat, reduce.dims, names.subset)
```

Arguments

`dat`	The data to operate on.
`reduce.dims`	The names or indices of the dimensions to reduce to 1 dimension.
`names.subset`	Optionally, a subset of dimension names to copy.

Details

This function takes input data, a vector of dimensions to reduce to 1 dimension, and optionally a subset of dimnames to copy. It returns the data with the specified dimensions shrunk down to 1 dimension.

Value

The data with the specified dimensions reduced to 1 dimension.

Note

The dimensions to reduce must be adjoining dimensions.

Examples

```
## Take example data and flatten the last two dims down to one.
dat <- structure(1:8, .Dim=c(2, 2, 2))
dat.flat <- flatten.dims(dat, 2:3)
```

---

get.climdex.functions *Returns a list of Climdex functions, with parameters curried in.*

Description

Returns a list of Climdex functions, with parameters curried in.

Usage

```
get.climdex.functions(vars.list, metadata.config, fclimdex.compatible = TRUE)
```

Arguments

`vars.list`	The variable list, as created by `get.climdex.variable.list`.
`fclimdex.compatible`	Whether to create fclimdex compatible functions.

Details

This function takes a variable list (as created by `get.climdex.variable.list`) and creates a list of functions corresponding to the specified indices, with parameters such as time resolution curried in. This allows for these functions to be called with just the `t` object as an argument, easing the automation of computing indices.

Value

A list of functions, named by the variable they compute.

Examples

```
## Get Climdex functions for a variable list with all appropriate params
## curried in, so that all they take is a t object.
cdx.funcs <- get.climdex.functions(get.climdex.variable.list(c("tmax", "tmin")))
```

---

get.climdex.variable.list

Returns a list of Climdex variables given constraints

Description

Returns a list of Climdex variables given constraints.

Usage

```
get.climdex.variable.list(
  source.data.present,
  metadata.config,
  time.resolution = c("all", "annual", "monthly", "seasonal", "halfyear"),
  climdex.vars.subset = NULL
)
```

Arguments

`source.data.present`
: A vector of strings naming the data that's present; at least one of (tmin, tmax, prec, tavg).

`metadata.config`
: config object read using `read_json_metadata_config_file`. This contains all the metadata such as the output long names of the indices in the output NCDF files.

`time.resolution`
: The time resolutions to compute indices at. See `create.indices.from.files`.

`climdex.vars.subset`
: A character vector of lower-case names of Climdex indices to calculate (eg: tr, fd, rx5day). See `create.indices.from.files`.

Details

This function takes a character vector which specifies what source data is present and a time resolution, and generates a list of names consisting of the variable and the time resolution, separated by an underscore.

Value

A character vector containing variable names with time resolutions appended.

See Also

create.indices.from.files

Examples

```
## Get all variables which require tmin and/or tmax, for all time resolutions.
var.list1 <- get.climdex.variable.list(c("tmax", "tmin"))

## Get all variables which require prec with an annual time resolution.
var.list2 <- get.climdex.variable.list("prec", time.resolution="annual")

## Get the intersection of a set list of vars and available data.
sub.vars <- c("su", "id", "tr", "fd", "gsl", "csdi", "wsdi", "r10mm")
var.list3 <- get.climdex.variable.list("tmax", climdex.vars.subset=sub.vars)
```

---

get.climdex.variable.metadata

Returns metadata for specified Climdex variables

Description

Returns metadata for specified Climdex variables.

Usage

```
get.climdex.variable.metadata(vars.list, template.filename, metadata.config)
```

Arguments

vars.list	The list of variables, as returned by get.climdex.variable.list.
template.filename	The filename template to be used when generating filenames.
metadata.config	config object read using read_json_metadata_config_file. This contains all the metadata such as the output long names of the indices in the output NCDF files.

Details

This function returns metadata suitable for use in NetCDF files for the specified variables.

Value

A data frame containing the following:

- long.nameLong names for the variable
- var.nameVariable name for use in the file
- unitsUnits for the variable
- time.resThe time resolution of the variable.
- base.period.attrWhether to include a base period attribute
- standard.nameStandard name to use for the variable
- filenameFilename to be written out

Examples

```
## Get metadata (including filenames) for specified variables.
fn <- "pr_day_BCCAQ+ANUSPLIN300+MRI-CGCM3_historical+rcp85_r1i1p1_19500101-21001231.nc"
var.list2 <- get.climdex.variable.list("prec", time.resolution="annual")
md <- get.climdex.variable.metadata(var.list2, fn)
```

---

get.data *Retrieve and convert data to correct units and dimensions.*

Description

Retrieve and convert data to correct units and dimensions.

Usage

```
get.data(f, v, subset, src.units, dim.axes)
```

Arguments

f	The NetCDF file to retrieve data from; an object of class ncdf4.
v	The variable to retrieve data from.
subset	The subset to retrieve.
src.units	The source units to convert data from.
dim.axes	The dimension axes to be used.
dest.units	The destination units to convert to.

Details

This function retrieves NetCDF data for the specified subset from the specified file and variable; converts from src.units to dest.units, transposes the data to (T, S) dimensionality, and returns the result.

Value

The retrieved and massaged data.

Examples

```
get.data(f, "pr", list(Y=3), "kg m-2 s-1", "kg m-2 s-1", c(X="lon",Y="lat",T="time"))
```

`get.northern.hemisphere.booleans`

Determine what portions of a subset are within the northern hemisphere.

Description

Determine what portions of a subset are within the northern hemisphere.

Usage

```
get.northern.hemisphere.booleans(subset, f, v, projection)
```

Arguments

`subset`	The subset to use.
`f`	The NetCDF file to use; an object of class `ncdf4`.
`v`	The variable in question.
`projection`	The proj4 string to use; NULL if the data is not in a projected coordinate space.

Details

Given a subset, a file, a variable, and a projection, determine what positions are within the northern hemisphere, returning the result as an array of booleans.

Value

An array of booleans corresponding to the subset containing TRUE if the point is within the northern hemisphere, and FALSE otherwise.

Examples

```
## Open files, etc.
input.files <- c("tasmax_NAM44_CanRCM4_ERAINT_r1i1p1_1989-2009.nc")
f <- list(nc_open(input.files))
f.v <- lapply(f, ncdf4.helpers::nc.get.variable.list, min.dims=2)
bools <- get.northern.hemisphere.booleans(list(X=1:2, Y=1:2), f[[1]], f.v[[1]], NULL)
```

`get.quantiles.for.stripe` *Compute Climdex thresholds for a subset / stripe*

Description

Compute Climdex thresholds for a subset / stripe

Usage

```
get.quantiles.for.stripe(
  subset,
  ts,
  base.range,
  dim.axes,
  v.f.idx,
  variable.name.map,
  src.units,
  f
)
```

Arguments

`subset`	The subset to use.
`ts`	The associated time data, as created by `nc.get.time.series`.
`base.range`	The base range; a vector of two numeric years.
`dim.axes`	The dimension axes for the input data.
`v.f.idx`	A mapping from variables to files, as created by `get.var.file.idx`.
`variable.name.map`	A mapping from standardized names (tmax, tmin, prec) to NetCDF variable names.
`src.units`	The source units to convert data from.
`f`	A list of objects of type `ncdf4`, consisting of the open input files. If missing, will be pulled from the global namespace.
`dest.units`	The destination units to convert to.

Details

Given a subset and ancillary data, load and convert data, get the out-of-base quantiles for the data for each point, and return the result.

Note

This function relies on an object named 'f' and containing the opened NetCDF files being part of the global namespace.

Examples

```
## Establish basic inputs.
author.data <- list(institution="Looney Bin", institution_id="LBC")
input.files <- c("pr_NAM44_CanRCM4_ERAINT_r1i1p1_1989-2009.nc")

## Prepare derived inputs.
f <- lapply(input.files, ncdf4::nc_open)
variable.name.map <- c(tmax="tasmax", tmin="tasmin", prec="pr")
f.meta <- create.file.metadata(f, variable.name.map)
threshold.dat <- get.thresholds.metadata(names(f.meta$v.f.idx))

## Create output file
thresh.file <- create.thresholds.file("thresh.nc", f, f.meta$ts, f.meta$v.f.idx, variable.name.map,
                                      c(1991, 2000), f.meta$dim.size, f.meta$dim.axes,
                                      threshold.dat, author.data)

## Compute threshold quantiles for stripe
q <- get.quantiles.for.stripe(list(Y=1), f.meta$ts, c(1991, 2000), f.meta$dim.axes,
                              f.meta$v.f.idx, variable.name.map, f.meta$src.units,
                              f.meta$dest.units, f)
```

`get.quantiles.object` *Extract a single quantiles object from a set of thresholds.*

Description

Extract a single quantiles object from a set of thresholds.

Usage

```
get.quantiles.object(thresholds, idx, metadata.config)
```

Arguments

`thresholds`	The thresholds, as extracted by `get.thresholds.chunk`.
`idx`	The index to extract.

Details

From a set of thresholds as retrieved from one or more NetCDF files containing thresholds, this function extracts a single point and converts the format to one suitable for passing to `climdexInput.raw`.

Value

A quantiles object suitable for passing to `climdexInput.raw` as the `quantiles` argument.

Examples

```
## Define mappings and filenames.
thresholds.name.map <- c(tx10thresh="tx10thresh", tn10thresh="tn10thresh", tx90thresh="tx90thresh",
                  tn90thresh="tn90thresh", r95thresh="r95thresh", r99thresh="r99thresh")
thresh.files <- "thresholds.nc"

## Open files, etc.
cdx.funcs <- get.climdex.functions(get.climdex.variable.list("tmax"))
thresholds.netcdf <- lapply(thresh.files, nc_open)
t.f.idx <- get.var.file.idx(thresholds.name.map, lapply(thresholds.netcdf,
                            ncdf4.helpers::nc.get.variable.list, min.dims=2))

## Get thresholds chunk.
dat <- get.thresholds.chunk(list(Y=1), cdx.funcs, thresholds.netcdf, t.f.idx, thresholds.name.map)

## Get quantiles object for index 2
q <- get.quantiles.object(dat, 2)
```

`get.thresholds.chunk` *Retrieve thresholds for a subset*

Description

Retrieve thresholds for a subset

Usage

```
get.thresholds.chunk(
  subset,
  cdx.funcs,
  thresholds.netcdf,
  t.f.idx,
  thresholds.name.map
)
```

Arguments

`subset`	The subset to use.
`cdx.funcs`	The functions to be applied to the data, as created by `get.climdex.functions`.
`thresholds.netcdf`	One or more NetCDF files containing thresholds.
`t.f.idx`	A mapping from threshold variables to threshold files, as created by `get.var.file.idx`.
`thresholds.name.map`	A mapping from standardized names (tx10thresh, tn90thresh, etc) to NetCDF variable names.

Details

Given a subset, a set of Climdex functions (as created by `get.climdex.functions`), and ancillary data, load the thresholds required for the functions being called and return them.

Examples

```
## Define mappings and filenames.
thresholds.name.map <- c(tx10thresh="tx10thresh", tn10thresh="tn10thresh", tx90thresh="tx90thresh",
                  tn90thresh="tn90thresh", r95thresh="r95thresh", r99thresh="r99thresh")
thresh.files <- "thresholds.nc"

## Open files, etc.
cdx.funcs <- get.climdex.functions(get.climdex.variable.list("tmax"))
thresholds.netcdf <- lapply(thresh.files, nc_open)
t.f.idx <- get.var.file.idx(thresholds.name.map, lapply(thresholds.netcdf,
                            ncdf4.helpers::nc.get.variable.list, min.dims=2))

## Get thresholds chunk.
dat <- get.thresholds.chunk(list(Y=1), cdx.funcs, thresholds.netcdf, t.f.idx, thresholds.name.map)
```

`get.thresholds.metadata` *Retrieve threshold metadata*

Description

Retrieve threshold metadata

Usage

```
get.thresholds.metadata(var.names, metadata.config)
```

Arguments

`var.names`	A vector containing names of available variables (tmax, tmin, prec).

Details

Returns units, long names, locations within the t data structure, and whether time data should be included given the variable information available.

Value

A list containing metadata for each of the six thresholds.

Examples

```
thresholds.meta <- get.thresholds.metadata("prec")
```

`get.var.file.idx` *Create mapping from variables to files.*

Description

Create mapping from variables to files.

Usage

```
get.var.file.idx(variable.name.map, v.list)
```

Arguments

`variable.name.map`	A mapping from standardized names (tmax, tmin, prec) to NetCDF variable names.
`v.list`	A list containing a vector of variables in each file.

Details

Given a variable name map and list of variables in each file, determine a mapping from variables to files.

Value

A vector mapping standardized variable names (tmax, tmin, prec) to indices in the file list.

Examples

```
## Not run:
## Get mapping for a single file.
input.files <- c("pr_NAM44_CanRCM4_ERAINT_r1i1p1_1989-2009.nc")
f <- lapply(input.files, ncdf4::nc_open)
v.list <- lapply(f, ncdf4.helpers::nc.get.variable.list, min.dims=2)
v.f.idx <- get.var.file.idx(variable.name.map, v.list)

## End(Not run)
```

gridclimind — *gridclimind, a package to calculate Climdex indices from NetCDF files.*

Description

This package implements code to facilitate computation of Climdex indices from NetCDF input files.

Details

The Climdex climate extremes indices have historically been calculated using Fortran code. This has a number of problems:

- Difficult to test
- Difficult to modify (for instance, to add NetCDF file I/O)
- Difficult to parallelize

The `climind` package provides an easy interface to efficient computation of Climdex indices. This package is complementary to it, providing easy access to functions to compute indices in parallel, using NetCDF files as input and output. It implements chunked processing of input files to keep memory usage reasonable; it implements parallel computation using the `snow` library; and it includes a test suite to verify correctness of the implementation. Furthermore, the package has a modular design, allowing for easy extension to allow for adaptation to changing or custom requirements. For example, the metadata is stored in json files that are included with the package. This allows the core of the code to separate from the metadata details, for example allowing one to switch on-the-fly.

Users of this package should pay particular attention to the `create.indices.from.files` function, which computes Climdex indices given NetCDF input files; and `create.thresholds.from.file`, which computes thresholds for use with threshold-based indices given NetCDF input files. Many of the other functions exposed by the package are intended to provide for extensibility, but are unlikely to be routinely used by users of this package.

References

http://etccdi.pacificclimate.org/list_27_indices.shtml

Karl, T.R., N. Nicholls, and A. Ghazi, 1999: CLIVAR/GCOS/WMO workshop on indices and indicators for climate extremes: Workshop summary. Climatic Change, 42, 3-7.

Peterson, T.C., and Coauthors: Report on the Activities of the Working Group on Climate Change Detection and Related Rapporteurs 1998-2001. WMO, Rep. WCDMP-47, WMO-TD 1071, Geneve, Switzerland, 143pp.

Zhang, X., 2005: Avoiding inhomogeneity in percentile-based indices of temperature extremes. Journal of Climate 18.11 (2005):1641-.

See Also

`create.indices.from.files`, `create.thresholds.from.file`

thresholds.close *Close thresholds file(s)*

Description

Close thresholds file(s)

Usage

```
thresholds.close(thresholds.nc)
```

Arguments

thresholds.nc	A list of objects of class ncdf4, or NULL

Details

This function closes one or more thresholds files.

Examples

```
## Not run:
## Open a single thresholds file, then close it.
thresholds.files <- c("thresh.nc")
thresh <- thresholds.open(thresholds.files)
thresholds.close(thresh)

## End(Not run)
```

thresholds.open *Open thresholds file(s)*

Description

Open thresholds file(s)

Usage

```
thresholds.open(thresholds.files)
```

Arguments

thresholds.files	A character vector containing the names of thresholds files.

Details

This function opens one or more thresholds files and returns the ncdf4 objects as a list.

Value

A list of objects of class ncdf4, or NULL if thresholds.files is NULL.

Examples

```
## Not run:
## Open a single thresholds file
thresholds.files <- c("thresh.nc")
thresh <- thresholds.open(thresholds.files)

## End(Not run)
```

---

write.climdex.results *Write out computed climdex results*

Description

Write out computed climdex results

Usage

```
write.climdex.results(
  climdex.results,
  chunk.subset,
  cdx.ncfile,
  dim.size,
  cdx.varname
)
```

Arguments

climdex.results	The results to write out.
chunk.subset	The corresponding subset.
cdx.ncfile	The list of NetCDF files to write the results out to.
dim.size	The overall size of the input data.
cdx.varname	The list of NetCDF variable names for the files in cdx.ncfile.

Details

Given a set of Climdex results, a subset, a set of files, and dimension sizes, write out the data to the appropriate files.

Examples

```
## Define mappings and filenames.
author.data <- list(institution="Looney Bin", institution_id="LBC")
input.files <- c("pr_NAM44_CanRCM4_ERAINT_r1i1p1_1989-2009.nc")
variable.name.map <- c(tmax="tasmax", tmin="tasmin", prec="pr")

## Open files, etc.
cdx.funcs <- get.climdex.functions("tmax")
f <- lapply(input.files, ncdf4::nc_open)
f.meta <- create.file.metadata(f, variable.name.map)
climdex.var.list <- get.climdex.variable.list(names(f.meta$v.f.idx), "all", NULL)
cdx.meta <- get.climdex.variable.metadata(climdex.var.list, input.files[1])

## Create output files
cdx.ncfile <- create.ncdf.output.files(cdx.meta, f, f.meta$v.f.idx, variable.name.map,
                                       f.meta$ts, get.time.origin(f, f.meta$dim.axes),
                                       c(1981,1990), "/foo", author.data)

## Compute indices for stripe
cdx <- compute.indices.for.stripe(list(Y=1), cdx.funcs, f.meta$ts, c(1991, 2000), f.meta$dim.axes,
                  f.meta$v.f.idx, variable.name.map, f.meta$src.units, f.meta$dest.units,
                           t.f.idx, NULL, f=f, thresholds.netcdf=NULL)

## Write out indices
write.climdex.results(cdx, list(Y=1), cdx.ncfile, f.meta$dim.size, cdx.meta$varname)
```

Index