# Package 'ProfoundData'

January 20, 2025

**Title** Downloading and Exploring Data from the PROFOUND Database

**Version** 0.2.1

**Date** 2020-03-13

**Description** Provides an R interface for the PROFOUND database <doi:10.5880/PIK.2019.008>. The PROFOUND database contains a wide range of data to evaluate vegetation models and simulate climate
impacts at the forest stand scale. It includes 9 forest sites across Europe, and provides for them a site
description as well as soil, climate, CO2, Nitrogen deposition, tree-level, forest stand-level and remote sensing data. Moreover, for a subset of 5 sites, also time series of carbon fluxes, energy balances and soil water are available.

**Depends** R (>= 3.1.0)

**Imports** methods (>= 3.3.2), sqldf (>= 0.4-10), DBI (>= 0.5-1), RSQLite (>= 1.1-2), RNetCDF (>= 1.9-1), settings (>= 0.2.4), zoo (>= 1.7-14), tcltk

**License** GPL-3

**LazyData** true

**Suggests** knitr (>= 1.15.1), rmarkdown (>= 1.3), R.rsp (>= 0.40.0), testthat (>= 1.0.2), BayesianTools

**VignetteBuilder** R.rsp

**URL** https://cost-fp1304-profound.github.io/ProfoundData/

**BugReports** https://github.com/COST-FP1304-PROFOUND/ProfoundData/issues

**RoxygenNote** 7.0.2.9000

**Encoding** UTF-8

**NeedsCompilation** no

**Author** Ramiro Silveyra Gonzalez [aut],
Christopher Reyer [aut],
Mahnken Mats [aut],
Florian Hartig [aut, cre] (<https://orcid.org/0000-0002-6255-9059>),
Friedrich Bohn [ctb]

**Maintainer** Florian Hartig <florian.hartig@biologie.uni-regensburg.de>

**Repository** CRAN

**Date/Publication** 2020-03-30 16:10:02 UTC

# Contents



---

`ProfoundData-package` *Overview of the functions in the ProfoundData package*

---

## Description

The ProfoundData R package provides functions to explore, visualize and get data from the PROFOUND database. The development of the PROFOUND database and the ProfoundData R package was faciliated by COST Action FP1304 PROFOUND.

A brief description of the PROFOUND database is avalaible in the vignette 'PROFOUNDdatabase' ( you can open it by running this command: `vignette('PROFOUNDdatabase')`).

Note: before you can use the package, you need to download the database via `downloadDatabase` and register the database location via `setDB`

Below is a list of the package's functions grouped by theme. See the package vignette for more information and examples (`vignette('ProfoundData')`).

## I. Browse the database

Functions to explore the database.

- `browseData` To check
  - available sites
  - available datasets
  - available variables for a dataset
  - available datasets for a site

  - database version
  - metadata
  - policy
  - source
  - site description
- summarizeData To obtain
  - data overviews
  - data summaries

### II. Extract data

Functions to extract data. Data can be extracted for one site and one dataset at a time.

- getData To extract data

### III. Visualize data

Functions to plot data from the database.

- plotData To plot data

### IV. Utilities

Miscellanous

- downloadDatabase To download the database
- setDB To set the connection to the database
- getDB To retrieve the filepath to the database
- queryDB To pass self-defined queries
- reportDB To create a site-by-site report of the database
- writeSim2netCDF To write netCDF-files

### Author(s)

Except where indicated otherwise, the functions in this package were written by Ramiro Silveyra Gonzalez, Christopher Reyer, Florian Hartig and Mahnken. Ramiro Silveyra Gonzalez was the main developer of the package. Florian Hartig is currently the maintainer.

`browseData` *A browse database function*

## Description

A function to provide information on available data in the PROFOUND database.

## Usage

```
browseData(
  dataset = NULL,
  site = NULL,
  location = NULL,
  variables = FALSE,
  collapse = TRUE
)
```

## Arguments

| | |
|---|---|
| `dataset` | a character string providing the name of the dataset (optional). |
| `site` | a character string providing the name of the site (optional). |
| `location` | deprecated argument. Please use site instead. |
| `variables` | a boolean indicating whether to return the variables of a dataset, instead of the available sites. |
| `collapse` | a boolean indicating whether to return the compact (TRUE) or the extended (FALSE) data overview, if neither site nor dataset are passed to the function. |

## Details

Besides providing information on available data, this function allows to access the database metadata, policy, data sources and site descriptions.

## Value

- if no arguments, an overview of available data
- if metadata, the requested metadata
- if dataset and variables, available variables for a dataset
- if dataset, available sites for the dataset
- if site, available datasets for a site
- if site and dataset, returns an integer. Availability is coded as 0 = no available and 1 = available. This is the quickest option to check availability.

## Note

To report errors in the package or the data, please use the issue tracker in the GitHub repository of ProfoundData https://github.com/COST-FP1304-PROFOUND/ProfoundData

## Examples

```
# example requires that a sql DB is registered via setDB(dbfile)
# when run without a registered DB, you will get a file query (depending on OS)

## Not run:
# See available data of the database
overview <- browseData()
# Hint: If *collapse* FALSE, full version of the overview table
overview <- browseData(collapse = FALSE)

# Available datasets
tables <- browseData(dataset = "DATASETS")

# Available variables for a given dataset
variables <- browseData(dataset = "CLIMATE_LOCAL", variables = TRUE)

# Available sites for a given dataset
available <- browseData(dataset = "CLIMATE_LOCAL")

# Available datasets for a given site
available <- browseData(site ="le_bray")

# Whether a dataset is available for a specific site
available <- browseData(site ="le_bray", dataset = "CLIMATE_LOCAL")

# See version history
version <- browseData(dataset = "VERSION")

# See metadata
metadata <- browseData(dataset = "METADATA_DATASETS")
metadata <- browseData(dataset = "METADATA_CLIMATE_LOCAL")

# See metadata of a specific site
metadata <- browseData(dataset = "METADATA_TREE", site = "solling_spruce")

# See data source
source <- browseData(dataset = "SOURCE")

# See data source of a specific site
source <- browseData(dataset = "SOURCE", site = "solling_spruce")

# See data policy
source <- browseData(dataset = "POLICY")

# See data policy of a specific site
policy <- browseData(dataset = "POLICY", site = "solling_spruce")

## End(Not run)
```

---

`downloadDatabase` *Downloads the PROFOUND database*

---

## Description

This function downloads the PROFOUND database

## Usage

```
downloadDatabase(location = NULL)
```

## Arguments

| | |
|---|---|
| location | file system location to store the database. If not provide, the function will use the current working directory. |

## Details

This is a convenience function to quickly download the PROFOUND database. The function will query you to ask about the path to store the databse, and will return a string with the location, for use in setDB

## Value

a string with the location of the sql database

## Author(s)

Florian Hartig

## See Also

getDB, setDB

## Examples

```
## Not run:

# For downloading the data, use
dbfile <- downloadDatabase(location = tempdir())

# Set the connection using the setDB function
setDB(dbfile)

# check if database is available
getDB()

## End(Not run)
```

getData *A function to extract data*

## Description

A function to extract datasets for a site from the PROFOUND database.

## Usage

```
getData(
  dataset,
  site = NULL,
  location = NULL,
  forcingDataset = NULL,
  forcingCondition = NULL,
  species = NULL,
  variables = NULL,
  period = NULL,
  collapse = TRUE,
  quality = NULL,
  decreasing = TRUE
)
```

## Arguments

| | |
|---|---|
| dataset | a character string providing the name of a dataset. |
| site | a character string providing the name of a site. |
| location | deprecated argument. Please use site instead. |
| forcingDataset | a character string providing the name of a forcingDataset. Only relevant for ISIMIP datasets. |
| forcingCondition | a character string providing the name of a forcingCondition. Only relevant for ISIMIP datasets. |
| species | a character string providing the species name or species id. |
| variables | a character array holding the variables to be plotted. Default is all variables. |
| period | a character array with either start or start and end of the subset. It must have the format "YYYY-MM-DD", or c("YYYY-MM-DD", "YYYY-MM-DD"). |
| collapse | a boolean indicating whether the returned data should be a single data frame (TRUE) or a list of data frames (FALSE). Relevant when downloading SOIL and ISIMIP datasets. |
| quality | a number indicating the quality threshold to be used. Default is none. |
| decreasing | a boolean indicating whether the quality threshold should be applied up- or downwards. |

## Details

When using quality, please be aware that the threshold value is included in the returned data. Thresholding works by removing data values that are greater (decreasing = TRUE) or smaller (decreasing = FALSE) than the given value. The quality parameter is only relevant for datasets that have quality flags. These are ATMOSPHERICHEATCONDUCTION, SOILTS, FLUX, METEOROLOGICAL, and CLIMATE LOCAL. Please check the metadata of each dataset before using this parameter.

## Value

a data frame or a list of data frames, depending on collapse.

## Note

To report errors in the package or the data, please use the issue tracker in the GitHub repository of ProfoundData https://github.com/COST-FP1304-PROFOUND/ProfoundData

## Examples

```
# example requires that a sql DB is registered via setDB(dbfile)
# when run without a registered DB, you will get a file query (depending on OS)

## Not run:
# Get SITES data
sites <- getData(dataset =  "SITES")

# Get SITES data of a specific site
site <- getData(dataset =  "SITES", site = "soro")

# Get SITEDESCRIPTION
sites <- getData(dataset =  "SITEDESCRIPTION")

# Get SITEDESCRIPTION of a specific site
site <- getData(dataset =  "SITEDESCRIPTION", site = "lebray")

# Get any dataset for a site
data <- getData(dataset =  "CLIMATE_LOCAL", site = "soro")

# Get ISIMIP datasets as a list with data frames
data <- getData(dataset ="CLIMATE_ISIMIP2A", site = "soro", collapse = TRUE)
data <- getData(dataset ="CLIMATE_ISIMIP2B", site = "soro", collapse = TRUE)
data <- getData(dataset ="NDEPOSITION_ISIMIP2B", site = "soro", collapse = TRUE)

# Get ISIMIP datasets as an unique data frame
data <- getData(dataset ="CLIMATE_ISIMIP2A", site = "soro", collapse = FALSE)
data <- getData(dataset ="CLIMATE_ISIMIP2B", site = "soro", collapse = FALSE)
data <- getData(dataset ="NDEPOSITION_ISIMIP2B", site = "soro", collapse = FALSE)

# Get SOIL data. Collapse FALSE is recommended.
data <- getData(dataset ="SOIL", site = "soro", collapse = FALSE)
```

```
# Get specific forcing datasets and/or forcing conditions
data <- getData(dataset ="CLIMATE_ISIMIP2B", site ="soro",
                forcingDataset="GFDLESM2M", forcingCondition ="rcp2p6")
# which is equivalent to
data <- getData(dataset ="CLIMATE_ISIMIP2B_GFDLESM2M_rcp2p6", site ="soro")

# Specify variables
data <- getData(dataset ="CLIMATE_ISIMIP2B", site ="soro",
                forcingDataset="GFDLESM2M", forcingCondition ="rcp2p6",
                variables = "p_mm")
data <- getData(dataset ="CLIMATE_ISIMIP2B", site ="soro",
                forcingDataset="GFDLESM2M", forcingCondition ="rcp2p6",
                variables = c("tmax_degC","p_mm"))

# Specify species
data <- getData(dataset ="TREE", site ="hyytiala", species = "Pinus sylvestris")
data <- getData(dataset ="TREE", site ="hyytiala", species = "pisy")
data <- getData(dataset ="STAND", site ="hyytiala", species = "Picea abies")
data <- getData(dataset ="STAND", site ="hyytiala", species = "piab")

# Specify period
data <- getData(dataset ="CLIMATE_ISIMIP2B", site ="soro",
                forcingDataset="GFDLESM2M", forcingCondition ="rcp2p6",
                period = c("2006-01-01","2006-12-31"))

# Specify quality
data <- getData(dataset = "CLIMATE_LOCAL", site = "soro",
                quality = 1, decreasing = FALSE)
data <- getData(dataset = "FLUX", site = "soro",
                quality = 0, decreasing = TRUE)

## End(Not run)
```

---

`getDB` *A function to return information of the database connection*

---

## Description

A function to return the filepath to the database.

## Usage

```
getDB()
```

## Value

a character string with the filepath to the database

## Author(s)

Ramiro Silveyra Gonzalez

## See Also

setDB, downloadDatabase

## Examples

```
## Not run:

# For downloading the data, use
dbfile <- downloadDatabase(location = tempdir())

# Set the connection using the setDB function
setDB(dbfile)

# check if database is available
getDB()

## End(Not run)
```

---

getPanels *getPanels*

---

## Description

Calculates the argument x for par(mfrow = x) for a desired number of panels

## Usage

```
getPanels(x)
```

## Arguments

| | |
|---|---|
| x | the desired number of panels |

## Author(s)

Florian Hartig

plotData *A function to plot data*

## Description

A function to plot data of a site from the PROFOUND database.

## Usage

```
plotData(
  dataset,
  site,
  location = NULL,
  forcingDataset = NULL,
  forcingCondition = NULL,
  species = NULL,
  variables = NULL,
  period = NULL,
  aggregated = NULL,
  FUN = mean,
  automaticPanels = T,
  quality = NULL,
  decreasing = TRUE
)
```

## Arguments

| | |
|---|---|
| dataset | a character string providing the name of a dataset |
| site | a character string providing the name of a site. |
| location | deprecated argument. Please use site instead. |
| forcingDataset | a character string providing the name of a forcingDataset. Only relevant for ISIMIP datasets. |
| forcingCondition | a character string providing the name of a forcingCondition. Only relevant for ISIMIP datasets. |
| species | a character string providing the species name or species id. |
| variables | a character array holding the variables to be plotted. Default is all variables. |
| period | a character array with either start or start and end of the subset. It must have the format "YYYY-MM-DD", or c("YYYY-MM-DD", "YYYY-MM-DD"). |
| aggregated | a boolean indicating whether data should be aggregated for display. Possible values are date, day, month and year. |
| FUN | a function to use for aggregating the data |
| automaticPanels | a boolean indicating whether the function automatically creates panels |

| | |
|---|---|
| quality | a number indicating the quality threshold to be used. Default is none. |
| decreasing | a boolean indicating whether the quality threshold should be applied up- or downwards |

## Details

Plotting is not supported for the following datasets: OVERVIEW, SITES, SOIL(more). The aggregation of data relies on the function aggregate for zoo objects. The FUN parameter is passed to FUN from aggregate. Please check the help files of aggregate for further information. For handling NAs we recommend to pass self-defined functions (see examples).

## Value

plots for the specified dataset, site and variables

## Note

To report errors in the package or the data, please use the issue tracker in the GitHub repository of ProfoundData https://github.com/COST-FP1304-PROFOUND/ProfoundData

## Examples

```
# example requires that a sql DB is registered via setDB(dbfile)
# when run without a registered DB, you will get a file query (depending on OS)

## Not run:
# Normal plotting
plotData(dataset = "CLIMATE_LOCAL", site = "le_bray", automaticPanels = TRUE)
plotData(dataset = "TREE", site = "solling_beech",   automaticPanels = TRUE)
plotData(dataset = "CLIMATE_LOCAL", site = "le_bray", automaticPanels = FALSE)
plotData(dataset = "TREE", site = "solling_beech",   automaticPanels = FALSE)

# Plot specific forcing datasets and conditions
plotData(dataset ="CLIMATE_ISIMIP2B", site ="soro", forcingDataset="GFDLESM2M",
         forcingCondition ="rcp2p6", automaticPanels = TRUE)

# Plot specific variables
plotData(dataset ="CLIMATE_ISIMIP2B",site ="soro", forcingDataset="GFDLESM2M",
         forcingCondition ="rcp2p6", variables = "p_mm")
plotData(dataset ="TREE",site ="solling_beech", variables = "dbh1_cm")

# Plot specific species
plotData(dataset ="TREE", site ="hyytiala", species = "Pinus sylvestris",
         automaticPanels = TRUE)

# Plot specific period
plotData(dataset = "CLIMATE_LOCAL", site =  "soro",
         period = c("2011-01-01","2012-12-31"))

# Set quality threshold
plotData(dataset = "CLIMATE_LOCAL", site = "soro",
         period = c("2011-01-01","2012-12-31"), quality = 1, decreasing = FALSE)
```

```
plotData(dataset = "FLUX", site = "soro", period = c("2011-01-01","2012-12-31"),
         quality = 0, decreasing = TRUE)

# Plot aggregated data
plotData(dataset = "CLIMATE_ISIMIP2B", site ="soro",  forcingDataset= "GFDLESM2M",
         forcingCondition="rcp2p6", variables = "tmax_degC",
         period = c("2020-01-01", "2022-01-01"),
         aggregate = "month", FUN =median)
plotData(dataset = "CLIMATE_ISIMIP2B", site ="soro",  forcingDataset= "GFDLESM2M",
         forcingCondition="rcp2p6", variables = "p_mm",
         period = c("2020-01-01", "2022-01-01"),
         aggregate = "month", FUN =sum)

# Plot aggregated data with self-defined function. In this case, compare month
# mean temperature of one year to mean of all period
data <- getData(dataset = "CLIMATE_ISIMIP2B", site ="soro",  forcingDataset= "GFDLESM2M",
                forcingCondition="rcp2p6", variables = "tmax_degC")
meanTemperature <- mean(data$tmax_degC, na.rm = TRUE)
difference <- function(x) {mean(x, na.rm = TRUE) -meanTemperature}

plotData(dataset = "CLIMATE_ISIMIP2B", site ="soro",  forcingDataset= "GFDLESM2M",
         forcingCondition="rcp2p6", variables = "tmax_degC",
         period = c("2020-01-01", "2021-01-01"),
         aggregate = "month", FUN =difference)
abline(h = 0, col = "red")

## End(Not run)
```

---

queryDB *A function to query the database*

---

## Description

A queryDB function to perform self-defined queries on the PROFOUND database.

## Usage

```
queryDB(queryItem)
```

## Arguments

| | |
|---|---|
| queryItem | a character string providing the query |

## Note

To report errors in the package or the data, please use the issue tracker in the GitHub repository of ProfoundData https://github.com/COST-FP1304-PROFOUND/ProfoundData

## Examples

```
# example requires that a sql DB is registered via setDB(dbfile)
# when run without a registered DB, you will get a file query (depending on OS)

## Not run:
# Basic query
overview <- queryDB("SELECT * FROM OVERVIEW")
tree <- queryDB("SELECT * FROM TREE")

# Site name or site_id
myQuery <- queryDB("SELECT date, tmax_degC FROM CLIMATE_LOCAL WHERE site == 'hyytiala'")
myQuery <- queryDB("SELECT date, tmax_degC FROM CLIMATE_LOCAL_12")

# Tree species
myQuery <- queryDB("SELECT  * FROM TREE WHERE species == 'Picea abies'")
myQuery <- queryDB("SELECT  * FROM TREE_piab")

# Specify variables
myQuery <- queryDB("SELECT date, tmax_degC FROM CLIMATE_LOCAL WHERE
                   tmax_degC > 20 AND site == 'hyytiala' AND year == 2010")

## End(Not run)
```

---

# reportDB *A function report data*

---

## Description

This functions creates a site-by-site report of all avalaible data in the PROFOUND database. The summary is created with a rmarkdown document, which is rendered and saved as a html document.

## Usage

```
reportDB(outDir = NULL)
```

## Arguments

| | |
|---|---|
| outDir | a character string indicating the output directory in which the html file will be saved. If no value is provided, the working directory will be used as output directory. |

## Details

Please note that creating the report it might take several minutes.

## Value

a html file with the database report

## Note

To report errors in the package or the data, please use the issue tracker in the GitHub repository of ProfoundData https://github.com/COST-FP1304-PROFOUND/ProfoundData

## Examples

```
# example requires that a sql DB is registered via setDB(dbfile)
# when run without a registered DB, you will get a file query (depending on OS)

## Not run:
reportDB(outDir = tempdir())

## End(Not run)
```

---

# setDB *A function to set the connection to the database*

---

## Description

A setDB funtion to create a database connection object

## Usage

```
setDB(db_name = NULL)
```

## Arguments

| | |
|---|---|
| db_name | a character string providing the absolute path to the PROFOUND database. |

## Note

To report errors in the package or the data, please use the issue tracker in the GitHub repository of ProfoundData https://github.com/COST-FP1304-PROFOUND/ProfoundData

## See Also

getDB, downloadDatabase

## Examples

```
## Not run:

# For downloading the data, use
dbfile <- downloadDatabase(location = tempdir())

# Set the connection using the setDB function
setDB(dbfile)

# check if database is available
```

```
getDB()

## End(Not run)
```

---

summarizeData *A function to summarize data from the database*

---

## Description

This function allows to summarize datasets for a site from the PROFOUND database.

## Usage

```
summarizeData(
  dataset,
  site,
  location = NULL,
  forcingDataset = NULL,
  forcingCondition = NULL,
  by = "year",
  period = NULL,
  mode = "data"
)
```

## Arguments

| | |
|---|---|
| dataset | a character string providing the name of a climatic dataset (CLIMATE_LOCAL, CLIMATE_ISIMIP, ...) or the tree dataset.(change this you ) |
| site | a character string providing the name of a site. |
| location | deprecated argument. Please use site instead. |
| forcingDataset | a character string providing the name of a forcingDataset. Only relevant for ISIMIP datasets. |
| forcingCondition | a character string providing the name of a forcingCondition. Only relevant for ISIMIP datasets. |
| by | a character string indicating how to summarize the data. Currently supports by year and total. The latter refers to the entire available period. |
| period | a character array with either start or start and end of the subset. It must have the format "YYYY-MM-DD", or c("YYYY-MM-DD", "YYYY-MM-DD"). |
| mode | a character string indicating whether to display the data summary (data) or an overview (overview). |

## Details

This function is under development and has limited functionality. At the moment, it is possible to summarize daily climate datasets and tree data.

Data are summarized by years. Radiation and precipitation are provided as total yearly values, while the rest of climatic values are year mean values. For ISIMIP datasets a summary for whole period will be returned if the dataset comprises more than one forcing dataset and one forcing condition.

## Value

a data frame with the summary values

## Summary values

Summary is calculated by year

- Climatic datasets
  - p_mm and rad_Jcm2day are total yearly values
  - tmax_degC, tmean_degC, tmin_degC, relhum_percent, airpress_hPa and wind_ms are mean yearly values
- TREE dataset
  - density_treeha is the number of tree per ha
  - dbhArit_cm is the arithmetic mean diameter
  - dbhBa_cm is the average diameter weighted by basal area calculated as dbhBA = (ba1*dbh1 + ba2*dbh2 + ... + bak*dbhk) / (ba1 + ba2+ ... + bak), where bai and dbhi are the basal area and dbh, respectively, of the tree i, and i = 1, 2, . . , k
  - dbhDQ_cm is the mean squared diameter or quadratic mean diameter calculated as dbhDQ = sqrt( (dbh1^2 + dbh2^2+ ... + dbhk^2) / N), where dbhi is the diameter at breast height of tree i, i = 1, 2, . . , k, N is the total number of trees, and sqrt is the square root
  - heightArith_m is the arithmetic mean height
  - heightArith_m is the average height weighted by basal area or Loreys height calculated as heightBA = (ba1*h1 + ba2*h2 + ... + bak*hk) / (ba1 + ba2+ ... + bak), where bai and hi are the basal area and height, respectively, of the tree i, and i = 1, 2, . . , k
  - heightBA_m
  - ba_m2 is the basal area per hectare

## Note

To report errors in the package or the data, please use the issue tracker in the GitHub repository of ProfoundData https://github.com/COST-FP1304-PROFOUND/ProfoundData

## Examples

```
# example requires that a sql DB is registered via setDB(dbfile)
# when run without a registered DB, you will get a file query (depending on OS)

## Not run:
# Summarize by years
```

```
data <-summarizeData(dataset =  "TREE", site = "bily_kriz", by = "year",
                     mode = "data")
data <-summarizeData(dataset =  "CLIMATE_LOCAL", site = "bily_kriz", by = "year",
                     mode = "data")
data <-summarizeData(dataset =  "CLIMATE_ISIMIP2A", site = "bily_kriz", by = "year",
                     mode = "data")

# Specify forcing dataset or condition
data <-summarizeData(dataset ="CLIMATE_ISIMIP2B", site ="soro",
                     forcingDataset="GFDLESM2M", forcingCondition ="rcp2p6")


# Summarize total period
data <-summarizeData(dataset =  "TREE", site = "bily_kriz", by = "total",
                     mode = "data")
data <-summarizeData(dataset =  "CLIMATE_LOCAL", site = "bily_kriz", by = "total",
                     mode = "data")

# Summarize overview
data <-summarizeData(dataset =  "CLIMATE_LOCAL", site = "bily_kriz", mode = "overview")
data <-summarizeData(dataset =  "FLUX", site = "bily_kriz", mode = "overview")
data <-summarizeData(dataset =  "TREE", site = "bily_kriz", mode = "overview")
data <-summarizeData(dataset =  "STAND", site = "bily_kriz", mode = "overview")

## End(Not run)
```

---

`writeSim2netCDF` *A function to write netCDF-files*

---

## Description

This function transforms simulation results into netCDF files following the ISIMIP2 protocol

## Usage

```
writeSim2netCDF(
  df,
  comment1 = NA,
  comment2 = NA,
  institution = "PIK",
  contact = "isi-mip@pik-potsdam.de",
  modelname = "formind",
  GCM = "hadgem",
  RCP = "rcp85",
  ses = "nat",
  ss = "co2const",
  region = "Kroof",
  start = "1980",
  folder = "ISI-MIP"
)
```

## Arguments

| | |
|---|---|
| df | A data.frame containing in the first three columns longitude latitude and time. These columns are followed by columns containing the output variables. The columns have to be named with the output variable name as required by the 2B protocol. See table 21. |
| comment1 | Optional comment regarding your simulation |
| comment2 | Optional comment regarding your simulation |
| institution | Your institution |
| contact | Your mail address |
| modelname | The name of the used forest model |
| GCM | The climate model which created the used climate time series |
| RCP | The RCP scenario |
| ses | The scenario describing forest management. UMsoc equals the "nat" settings and histsoc and 2005soc equal the "man" settings in the ISIMIP2a protocol. Default value: "nat". |
| ss | "co2" for all experiments other than the sensitivity experiments for which 2005co2 is explicitly written. Note: even models in which CO2 has no effect should use the co2 identifier relevant to the experiment. Default value: "co2const". |
| region | the region or site of the simulation |
| start | the start year of the simulation. Default value: 1980. |
| folder | The folder in which all netCDF files will be written |

## Details

The function transforms your simulation output data frame into several netCDF -files and writes them into the indicated folder using the naming convention of the ISIMIP2(B)-protocol (https://www.isimip.org/protocol/). Units and long names of variables (table 21) will be created automatically.

## Author(s)

Friedrich J. Bohn

## Examples

```
# Produce sample data
df <- data.frame(lat = rep(20, 10),
                 lon = rep(30, 10),
                 time = seq(1920, 2010, 10),
                 dbh_total = c(10:19),
                 nee_total = rnorm(10, -0.5, 0.25),
                 evap = rnorm(10, 0.001, 0.0001),
                 cwood_fasy = seq(40, 85, 5))

# Convert multi-variable data.frame into single-variable netCDFs using ISIMIP naming conventions
writeSim2netCDF(df = df,
                comment1 = NA,
```

```
comment2 = NA,
institution = 'PIK',
contact = 'isi-mip@pik-potsdam.de',
modelname = "formind",
GCM = "hadgem",
RCP = "rcp85",
ses = "nat",
ss = "co2const",
region = "Kroof",
start = '1920',
folder = tempdir())
```

# Index