Package 'argoFloats'

October 23, 2026

Type Package

Title Analysis of Oceanographic Argo Floats

Version 1.0.10

Maintainer Dan Kelley <dan.kelley@dal.ca>

Depends R (>= 4.1.0)

Suggests colourpicker, curl, knitr, lubridate, markdown, ncdf4, ocedata, rmarkdown, shiny, shinyBS, s2, sf, testthat

Imports oce (>= 1.3.0), methods

BugReports https://github.com/ArgoCanada/argoFloats/issues

Description Supports the analysis of oceanographic data recorded by Argo autonomous drifting profiling floats. Functions are provided to (a) download and cache data files, (b) subset data in various ways, (c) handle quality-control flags and (d) plot the results according to oceanographic conventions. A shiny app is provided for easy exploration of datasets. The package is designed to work well with the 'oce' package, providing a wide range of processing capabilities that are particular to oceanographic analysis. See Kelley, Harbin, and Richards (2021) <doi:10.3389/fmars.2021.635922> for more on the scientific context and applications.

License GPL (>= 2)

Encoding UTF-8

URL https://github.com/ArgoCanada/argoFloats

LazyData false

BuildVignettes true

VignetteBuilder knitr

Config/roxygen2/version 8.1.1

NeedsCompilation no

Author Dan Kelley [aut, cre, cph] (ORCID: <https://orcid.org/0000-0001-7808-5911>), Jaimie Harbin [aut] (ORCID: <https://orcid.org/0000-0003-3774-3732>), Dewey Dunnington [ctb] (ORCID: <https://orcid.org/0000-0002-9415-4582>), Clark Richards [ctb] (ORCID: <https://orcid.org/0000-0002-7833-206X>)

Repository CRAN

Date/Publication 2026-10-23 13:10:02 UTC

Contents

applyQC . . . 3
argoDefaultDestdir . . . 5
argoFloats-class . . . 6
argoFloatsClearCache . . . 7
argoFloatsDebug . . . 7
argoFloatsGetFromCache . . . 9
argoFloatsIsCached . . . 9
argoFloatsListCache . . . 10
argoFloatsStoreInCache . . . 11
argoFloatsWhenCached . . . 11
D4900785_048.nc . . . 12
downloadWithRetries . . . 12
getIndex . . . 13
getProfileFromUrl . . . 16
getProfiles . . . 18
hexToBits . . . 19
index . . . 20
indexBgc . . . 21
indexDeep . . . 22
indexSynthetic . . . 22
mapApp . . . 23
merge,argoFloats-method . . . 25
plot,argoFloats-method . . . 26
plotArgoTS . . . 32
R3901602_163.nc . . . 34
readProfiles . . . 34
SD5903586_001.nc . . . 36
show,argoFloats-method . . . 37
showQCTests . . . 38
SR2902204_131.nc . . . 40
subset,argoFloats-method . . . 40
summary,argoFloats-method . . . 46
useAdjusted . . . 47
[[,argoFloats-method . . . 49

Index **52**

applyQC *Apply Quality Control Flags*

Description

This function examines the quality-control ("QC") flags within an argoFloats object that was created by readProfiles(). By default, it replaces all suspicious data with NA values, so they will not appear in plots or be considered in calculations. This is an important early step in processing, because suspicious Argo floats commonly report data that are suspicious based on statistical and physical measures, as is illustrated in the "Examples" section. See section 3.3 of Kelley et al. (2021) for more information about this function.

Usage

```
applyQC(x, flags = NULL, actions = NULL, debug = 0)
```

Arguments

x	an argoFloats object of type "argos", as created by readProfiles().
flags	A list specifying flag values upon which actions will be taken. This can take two forms. In the first form, the list has named elements each containing a vector of integers. For example, salinities flagged with values of 1 or 3:9 would be specified by flags=list(salinity=c(1,3:9)). Several data items can be specified, e.g. flags=list(salinity=c(1,3:9), temperature=c(1,3:9)) indicates that the actions are to take place for both salinity and temperature. In the second form, flags is a list holding a single unnamed vector, and this means to apply the actions to all the data entries. For example, flags=list(c(1,3:9)) means to apply not just to salinity and temperature, but to everything within the data slot. If flags is NULL then it is set to list(c(0,3,4,6,7,9)), which means to eliminate data that are considered bad (in some degree) or for which QC was not performed. Following Section 3.2.1 of reference 1, the flag meanings are as follows. • 0: No QC was performed • 1: Good data • 2: Probably good data • 3: Bad data that are potentially correctable • 4: Bad data • 5: Value changed • 6: Not used • 7: Not used • 8: Estimated value • 9: Missing data

actions	the actions to perform. The default, NULL, means to use the actions set up by readProfiles(), which, by default. causes any data flagged as suspicious to be set to NA.
debug	an integer passed to oce::handleFlags,argo-method(). If this is set to a positive value, then some debugging information will be printed as the processing is done.

Details

The work is done by using oce::handleFlags,argo-method() on each of the profiles stored within the object. In most cases, only the object needs to be specified, for the default actions coincide with common conventions for flags in Argo data.

Value

A copy of x but with each of the objects within its data slot having been passed through oce::handleFlags,argo-method()

Author(s)

Dan Kelley

References

1. Argo Data Management. "Argo User's Manual." Ifremer, July 5, 2022. https://doi.org/10.13155/29825.
2. Kelley, D. E., Harbin, J., & Richards, C. (2021). argoFloats: An R package for analyzing Argo data. Frontiers in Marine Science, (8), 636922. doi:10.3389/fmars.2021.635922

Examples

```
# Demonstrate applyQC to a built-in file
library(argoFloats)
f <- system.file("extdata", "SR2902204_131.nc", package = "argoFloats")
raw <- readProfiles(f)
clean <- applyQC(raw)
oldpar <- par(no.readonly = TRUE)
par(mar = c(3.3, 3.3, 1, 1), mgp = c(2, 0.7, 0))
plot(raw, col = "red", which = "TS")
points(clean[[1]][["SA"]], clean[[1]][["CT"]], pch = 20)
legend("topleft",
    pch = 20, cex = 1,
    col = c("black", "red"), legend = c("OK", "Flagged"), bg = "white"
)
par(oldpar)
```

argoDefaultDestdir *Get Default Values*

Description

These are helper functions that permit customization of various aspects of functions within the argoFloats package. The idea is that values can be set using `options()` or by using system 'environment variables', freeing the user from the necessity of altering the parameters provided to various argoFloats functions. See "Details" for more on the individual functions, noting that the entry for `argoDefaultServer()` is written with the most detail, with other entries relying on the background established there.

Usage

```
argoDefaultDestdir()

argoDefaultServer()

argoDefaultIndexAge()

argoDefaultProfileAge()

argoDefaultBathymetry()

hasArgoTestCache()
```

Details

- `argoDefaultServer()` The `getIndex()` and `getProfiles()` functions download data from a remote machine with URL specified by an argument named `server`. A user may prefer one server over another, perhaps due to speed of downloads to a particular research laboratory. However, that choice might not be best for another user, or even the same user at another time. Code reusability would be enhanced if the user had a way to alter the value of the `server` argument across all code, thereby eliminating the need to work in a text editor to find all instances of the function call. This is where `argoDefaultServer()` is useful. It lets the user specify a value for `server` either in R, using a call like `options(argoFloats.server="ifremer-https")` within R code (perhaps in the user's `.Rprofile` file), or by defining an environment variable named `R_ARGOFLOATS_SERVER` at the operating-system level. If the `argoFloats.server` option has not been set in R, and `R_ARGOFLOATS_SERVER` has not been set in the OS, then `argoDefaultServer()` defaults to `c("ifremer-https","usgodae")`.
- `argoDefaultDestdir()` returns the name of the local directory into which to store indices and other argo data. The option is named `argoFloats.destdir`, the environment variable is named `R_ARGOFLOATS_DESTDIR`, and the default is '"~/data/argo".
- `argoDefaultIndexAge()` returns the number of days beyond which an index is regarded as stale (and thus in need of a new download). The option is named `argoFloats.indexAge`, the environment variable is named `R_ARGOFLOATS_INDEX_AGE`, and the default is 1.0, for 1 day.

- `argoDefaultProfileAge()` returns the number of days beyond which an individual profile netCDF file is regarded as stale (and thus in need of a new download). The option is named `argoFloats.profileAge`, the environment variable is named `R_ARGOFLOATS_PROFILE_AGE`, and the default is 365.0 days. (Note that this is much higher than the value for `argoDefaultIndexAge()`, on the assumption that users will prefer recent indices, to get new data, but will prefer to update profile-specific datasets infrequently.)
- `argoDefaultBathymetry()` returns a value for the `bathymetry` argument used by `plot,argoFloats-method()`. The option is named `argoFloats.bathymetry`, the environment variable is named `R_ARGOFLOATS_BATHYMETRY`, and the default is `FALSE`.
- `hasArgoTestCache()` is not a user-facing function. Rather, its purpose is to speed the running of test suites during development, by preventing multiple downloads of data already downloaded.

Value

A value as described above, depending on the particular function in question.

Examples

```
argoDefaultServer()
argoDefaultDestdir()
argoDefaultIndexAge()
argoDefaultProfileAge()
argoDefaultBathymetry()
```

---

argoFloats-class *Base Class for argoFloats Objects*

Description

In the normal situation, users will not create these objects directly. Instead, they are created by functions such as `getIndex()`.

Slots

`data` The `data` slot is a list with contents that vary with the object type. For example, `getIndex()` creates objects of type `"index"` that have a single unnamed element in `data` that is a data frame. This data frame has a column named `file` that is used in combination with `metadata@ftpRoot` to form a URL for downloading, along with columns named `date`, `latitude`, `longitude`, `ocean`, `profiler_type`, `institution` and `date_update`. Other "get" functions create objects with different contents.

`metadata` The `metadata` slot is a list containing information about the data. The contents vary between objects and object types. That type is indicated by elements named `type` and `subtype`, which are checked by many functions within the package.

`processingLog` The `processingLog` slot is a list containing time-stamped processing steps that may be stored in the object by argoFloats functions. These are noted in `summary()` calls.

Examples

```
str(new("argoFloats"))
```

`argoFloatsClearCache` *Clear the Cache*

Description

Clear the cache. This is meant mainly for developers working in interactive sessions to test coding changes.

Usage

```
argoFloatsClearCache(debug = 0L)
```

Arguments

debug	an integer, passed to `argoFloatsDebug()`.

Value

integer, returned invisibly, indicating the number of items removed.

Author(s)

Dan Kelley, making a thin copy of code written by Dewey Dunnington

See Also

Other functions relating to cached values: `argoFloatsGetFromCache()`, `argoFloatsIsCached()`, `argoFloatsListCache()`, `argoFloatsStoreInCache()`, `argoFloatsWhenCached()`

`argoFloatsDebug` *Print Debugging Information*

Description

This function is intended mainly for use within the package, but users may also call it directly in their own code. Within the package, the value of debug is generally reduced by 1 on each nested function call, leading to indented messages. Most functions start and end with a call to `argoFloatsDebug()` that has `style="bold"` and `unindent=1`.

Usage

```
argoFloatsDebug(
  debug = 0,
  ...,
  style = "plain",
  showTime = FALSE,
  unindent = 0
)
```

Arguments

debug	an integer specifying the level of debugging. Values greater than zero indicate that some printing should be done. Values greater than 3 are trimmed to 3. Many functions pass debug=debug-1 down to deeper functions, which yields a nesting-indent format in the output.
...	values to be printed, analogous to the ... argument list of cat().
style	character value indicating special formatting, with "plain" for normal text, "bold" for bold-faced text, "italic" for italicized text, "red" for red text, "green" for green text, or "blue" for blue text. These codes may not be combined.
showTime	logical value indicating whether to preface message with the present time. This can be useful for learning about which operations are using the most time, but the default is not to show this, in the interests of brevity.
unindent	integer specifying the degree of reverse indentation to be done, as explained in the "Details" section.

Value

None (invisible NULL).

Author(s)

Dan Kelley

Examples

```
argoFloatsDebug(1, "plain text\n")
argoFloatsDebug(1, "red text\n", style = "red")
argoFloatsDebug(1, "blue text\n", style = "blue")
argoFloatsDebug(1, "bold text\n", style = "bold")
argoFloatsDebug(1, "italic text with time stamp\n", style = "italic", showTime = TRUE)
```

argoFloatsGetFromCache *Get an Item From The Cache*

Description

Get an Item From The Cache

Usage

```
argoFloatsGetFromCache(name, debug = 0)
```

Arguments

name	character value, naming the item.
debug	an integer, passed to argoFloatsDebug().

Value

The cached value, as stored with argoFloatsStoreInCache().

See Also

Other functions relating to cached values: argoFloatsClearCache(), argoFloatsIsCached(), argoFloatsListCache(), argoFloatsStoreInCache(), argoFloatsWhenCached()

argoFloatsIsCached *Check Whether an Item is Cached*

Description

Check Whether an Item is Cached

Usage

```
argoFloatsIsCached(name, debug = 0L)
```

Arguments

name	character value, naming the item.
debug	an integer, passed to argoFloatsDebug().

Value

A logical value indicating whether a cached value is available.

Author(s)

Dan Kelley, making a thin copy of code written by Dewey Dunnington

See Also

Other functions relating to cached values: `argoFloatsClearCache()`, `argoFloatsGetFromCache()`, `argoFloatsListCache()`, `argoFloatsStoreInCache()`, `argoFloatsWhenCached()`

`argoFloatsListCache` *List Items in the Cache*

Description

List items in the cache.

Usage

```
argoFloatsListCache(debug = 0L)
```

Arguments

debug	an integer, passed to `argoFloatsDebug()`.

Value

character vector of names of items.

Author(s)

Dan Kelley, making a thin copy of code written by Dewey Dunnington

See Also

Other functions relating to cached values: `argoFloatsClearCache()`, `argoFloatsGetFromCache()`, `argoFloatsIsCached()`, `argoFloatsStoreInCache()`, `argoFloatsWhenCached()`

argoFloatsStoreInCache *Store an Item in the Cache*

Description

Store an Item in the Cache

Usage

```
argoFloatsStoreInCache(name, value, debug = 0)
```

Arguments

name	character value, naming the item.
value	the new contents of the item.
debug	an integer, passed to argoFloatsDebug().

Value

None (invisible NULL).

See Also

Other functions relating to cached values: argoFloatsClearCache(), argoFloatsGetFromCache(), argoFloatsIsCached(), argoFloatsListCache(), argoFloatsWhenCached()

argoFloatsWhenCached *Check When an Item was Cached*

Description

Check When an Item was Cached

Usage

```
argoFloatsWhenCached(name, debug = 0L)
```

Arguments

name	character value, naming the item.
debug	an integer, passed to argoFloatsDebug().

Value

A logical value indicating whether a cached value is available.

Author(s)

Dan Kelley, making a thin copy of code written by Dewey Dunnington

See Also

Other functions relating to cached values: `argoFloatsClearCache()`, `argoFloatsGetFromCache()`, `argoFloatsIsCached()`, `argoFloatsListCache()`, `argoFloatsStoreInCache()`

D4900785_048.nc *Sample Argo File (Delayed Core Data)*

Description

This is NetCDF file for delayed-mode data for cycle 48 of Argo float 4900785, downloaded from `https://data-argo.ifremer.fr/dac/aoml/4900785/profiles/D4900785_048.nc` on 2020 June 24.

See Also

Other raw datasets: `R3901602_163.nc`, `SD5903586_001.nc`, `SR2902204_131.nc`

Examples

```
library(argoFloats)
a <- readProfiles(system.file("extdata", "D4900785_048.nc", package = "argoFloats"))
summary(a)
summary(a[[1]])
```

---

downloadWithRetries *Download and Cache a Dataset*

Description

General function for downloading and caching a dataset.

Usage

```
downloadWithRetries(
  url,
  destdir,
  destfile,
  quiet = FALSE,
  age = argoDefaultProfileAge(),
  retries = 3,
  async = FALSE,
  debug = 0
)
```

Arguments

url	character value giving the full URL of a file to be downloaded.
destdir	character value indicating the directory in which to store downloaded files. The default value is to compute this using argoDefaultDestdir(), which returns ~/data/argo by default, although it also provides ways to set other values using options(). Set destdir=NULL if destfile is a filename with full path information. File clutter is reduced by creating a top-level directory called data, with subdirectories for various file types; see "Examples".
destfile	either character value indicating the name of the file or NULL (the default), in which case the file name is constructed from the other parameters of the function call, so that subsequent calls with the same parameters will yield the same result; this is useful for caching.
quiet	logical value; use FALSE to show more verbose information when downloading files. (Even if quiet is TRUE, problems will still be reported.)
age	a numerical value indicating a time interval, in days. If the file to be downloaded from the server already exists locally, and was created is less than age days in the past, it will not be downloaded. The default, argoDefaultProfileAge(), is one year. Setting age=0 will force a download.
retries	integer telling how many times to retry a download, if the first attempt fails.
async	Use TRUE to perform requests asynchronously. This is much faster for many small files (e.g., Argo profile NetCDF files).
debug	integer value indicating level of debugging. If this is less than 1, no debugging is done. Otherwise, some functions will print debugging information. If a function call fails, the first step should be to rerun the function with debug=1, to see if the output suggests a problem in the call.

Value

A character value indicating the full pathname to the downloaded file, or NA, if there was a problem with the download.

Author(s)

Dan Kelley

---

getIndex *Get an Index of Available Argo Float Profiles*

Description

This function gets an index of available Argo float profiles, typically for later use as the first argument to getProfiles(). The source for the index may be (a) a remote data repository, (b) a local repository (see the keep argument), or (c) a cached RDA file that contains the result of a previous call to getIndex() (see the age parameter).

Usage

```
getIndex(
  filename = "core",
  server = argoDefaultServer(),
  destdir = argoDefaultDestdir(),
  age = argoDefaultIndexAge(),
  quiet = FALSE,
  keep = FALSE,
  debug = 0L
)
```

Arguments

filename	character value that indicates the file name to be downloaded from a remote server, or (if server is set to NULL) the name of a local file. For the remote case, the value of server must be taken from the first column of the table given in "Details", or (for some file types) as in the nickname given in the middle column. Note that the downloaded file name will be based on the full file name given as this argument, and that nicknames are expanded to the full filenames before saving. Note that the downloaded file is in gzipped format (indicated by a file name ending in .gz) and it is examined and processed by getIndex() to produce an R archive file (ending in .rda) that is stored locally. The .gz file is discarded by default, unless keep is set to TRUE. (See also the documentation on the server parameter, next, and the subsection entitled "Using a previously-downloaded index".)
server	an indication of the source for filename. There are 2 possibilities for this. (1) If server is NULL, then filename is taken to be the name of a local index file (ending in suffix .gz) that was previously downloaded from a server. The easiest way to get such a file is with a previous call to getIndex() with keep set to TRUE. (2) If server is a character vector (as is it is by default), it is taken to represent remote servers to be tried as sources for an index file. The use of multiple servers is a way to avoid errors that can result if a server refuses a download request. As of March 2023, the three servers known to work are "https://data-argo.ifremer.fr", "ftp://ftp.ifremer.fr/ifremer/argo" and "https://usgodae.org/pub/outgoing/argo". These may be referred to with nicknames "ifremer-https", "ifremer"and "usgodae". Any URL that can be used in curl::curl_download() is a valid value provided that the file structure is identical to the mirrors listed above. See argoDefaultServer() for how to provide a default value for server.
destdir	character value indicating the directory in which to store downloaded files. The default value is to compute this using argoDefaultDestdir(), which returns ~/data/argo by default, although it also provides ways to set other values using options(). Set destdir=NULL if destfile is a filename with full path information. File clutter is reduced by creating a top-level directory called data, with subdirectories for various file types; see "Examples".
age	numeric value indicating how old a downloaded file must be (in days), for it to be considered out-of-date. The default, argoDefaultIndexAge(), limits down-

loads to once per day, as a way to avoid slowing down a workflow with a download that might take a minute or so. Setting `age=0` will force a new download, regardless of the age of the local file, and that age is changed to 0 if `keep` is `TRUE`. The value of `age` is ignored if `server` is NULL (see "Using a previously downloaded Index" in "Details").

`quiet` logical value indicating whether to silence some progress indicators. The default is to show such indicators.

`keep` logical value indicating whether to retain the raw index file as downloaded from the server. This is `FALSE` by default, indicating that the raw index file is to be discarded once it has been analyzed and used to create a cached file (which is an RDA file). Note that if `keep` is `TRUE`, then the supplied value of `age` is converted to 0, to force a new download.

`debug` an integer indicating the level of debugging. If this is 0, then the function works somewhat quietly. If it is 1, messages are printed at various steps in the process. If it is any number higher than 1, then those messages will be prefixed by an indication of the time, down to the millisecond.

Details

Using an index from a remote server

The first step is to construct a URL for downloading, based on the `url` and `file` arguments. That URL will be a string ending in `.gz`, or `.txt` and from this the name of a local file is constructed by changing the suffix to `.rda` and prepending the file directory specified by `destdir`. If an `.rda` file of that name already exists, and is less than `age` days old, then no downloading takes place. This caching procedure is a way to save time, because the download can take from a minute to an hour, depending on the bandwidth of the connection to the server.

The resultant `.rda` file, which is named in the return value of this function, holds a list named `index` that holds following elements:

- `ftpRoot`, the FTP root stored in the header of the source `file` (see next paragraph).
- `server`, the URL at which the index was found, and from which `getProfiles()` can construct URLs from which to download the NetCDF files for individual float profiles.
- `filename`, the argument provided here.
- `header`, the preliminary lines in the source file that start with the # character.
- `data`, a data frame containing the items in the source file. The names of these items are determined automatically from `"core"`,`"bgcargo"`, `"synthetic"` files.

Some expertise is required in deciding on the value for the `file` argument to `getIndex()`. As of March 2023, the sites `https://usgodae.org/pub/outgoing/argo` and `ftp://ftp.ifremer.fr/ifremer/argo` contain multiple index files, as listed in the left-hand column of the following table. The middle column lists nicknames for some of the files. These can be provided as the `file` argument, as alternatives to the full names. The right-hand column describes the file contents. Note that the servers also provide files with names similar to those given in the table, but ending in `.txt`. These are uncompressed equivalents of the `.gz` files that offer no advantage and take longer to download, so `getIndex()` is not designed to work with them.

File Name	*Nickname*	*Contents*
ar_greylist.txt	-	Suspicious/malfunctioning floats
ar_index_global_meta.txt.gz	-	Metadata files
ar_index_global_prof.txt.gz	"argo" or "core"	Argo data
ar_index_global_tech.txt.gz	-	Technical files
ar_index_global_traj.txt.gz	"traj"	Trajectory files
argo_bio-profile_index.txt.gz	"bgc" or "bgcargo"	Biogeochemical Argo data (without S or T)
argo_bio-traj_index.txt.gz	"bio-traj"	Bio-trajectory files
argo_synthetic-profile_index.txt.gz	"synthetic"	Synthetic data, successor to "merge"

Using a previously downloaded index

In some situations, it can be desirable to work with local index file that has been copied directly from a remote server. This can be useful if there is a desire to work with the files in R separately from the argoFloats package, or with python, etc. It can also be useful for group work, in which it is important for all participants to use the same source file.

This need can be handled with getIndex(), by specifying filename as the full path name to the previously downloaded file, and at the same time specifying server as NULL. This works for both the raw files as downloaded from the server (which end in .gz, and for the R-data-archive files produced by getIndex(), which end in .rda. Since the .rda files load an order of magnitude faster than the .gz files, this is usually the preferred approach. However, if the .gz files are preferred, perhaps because part of a software chain uses python code that works with such files, then it should be noted that calling getIndex() with keep=TRUE will save the .gz file in the destdir directory.

Value

An object of class argoFloats with type="index", which is suitable as the first argument of getProfiles().

Author(s)

Dan Kelley and Jaimie Harbin

References

Kelley, D. E., Harbin, J., & Richards, C. (2021). argoFloats: An R package for analyzing Argo data. Frontiers in Marine Science, (8), 636922. doi:10.3389/fmars.2021.635922

---

getProfileFromUrl *Get Data for an Argo Float Profile*

Description

Get Data for an Argo Float Profile

Usage

```
getProfileFromUrl(
  url = NULL,
  destdir = argoDefaultDestdir(),
  destfile = NULL,
  age = argoDefaultProfileAge(),
  retries = 3,
  quiet = FALSE,
  debug = 0
)
```

Arguments

`url`	character value giving the URL for a NetCDF file containing an particular profile of a particular Argo float.
`destdir`	character value indicating the directory in which to store downloaded files. The default value is to compute this using `argoDefaultDestdir()`, which returns `~/data/argo` by default, although it also provides ways to set other values using `options()`. Set `destdir=NULL` if `destfile` is a filename with full path information. File clutter is reduced by creating a top-level directory called `data`, with subdirectories for various file types; see "Examples".
`destfile`	optional character value that specifies the name to be used for the downloaded file. If this is not specified, then a name is determined from the value of `url`.
`age`	a numerical value indicating a time interval, in days. If the file to be downloaded from the server already exists locally, and was created is less than `age` days in the past, it will not be downloaded. The default, `argoDefaultProfileAge()`, is one year. Setting `age=0` will force a download.
`retries`	integer telling how many times to retry a download, if the first attempt fails.
`quiet`	logical value; use `FALSE` to show more verbose information when downloading files. (Even if `quiet` is TRUE, problems will still be reported.)
`debug`	integer value indicating level of debugging. If this is less than 1, no debugging is done. Otherwise, some functions will print debugging information. If a function call fails, the first step should be to rerun the function with `debug=1`, to see if the output suggests a problem in the call.

Value

A character value naming the local location of the downloaded file, or `NA` if the file could not be downloaded.

Author(s)

Dan Kelley

getProfiles *Get "cycles"/"trajectory" Files Named in an argoFloats Index*

Description

This takes an index constructed with `getIndex()`, possibly after focusing with `subset,argoFloats-method()`, and creates a list of files to download from the server named in the index. Then these files are downloaded to the `destdir` directory, using filenames inferred from the source filenames. The value returned by `getProfiles()` is suitable for use by `readProfiles()`.

Usage

```
getProfiles(
  index,
  destdir = argoDefaultDestdir(),
  age = argoDefaultProfileAge(),
  retries = 3,
  skip = TRUE,
  quiet = TRUE,
  debug = 0
)
```

Arguments

`index`	an `argoFloats` object of type `"index"`, as created by `getIndex()`.
`destdir`	character value indicating the directory in which to store downloaded files. The default value is to compute this using `argoDefaultDestdir()`, which returns `~/data/argo` by default, although it also provides ways to set other values using `options()`. Set `destdir=NULL` if `destfile` is a filename with full path information. File clutter is reduced by creating a top-level directory called `data`, with subdirectories for various file types; see "Examples".
`age`	Option 1) a numerical value indicating a time interval, in days. If the file to be downloaded from the server already exists locally, and was created is less than age days in the past, it will not be downloaded. The default is one year. Setting age=0 will force a download. Option 2) "latest" meaning the file will only be downloaded if A) the file doesn't exist or B) the file does exist and the time it was created is older than the date_update in the index file
`retries`	integer telling how many times to retry a download, if the first attempt fails.
`skip`	A logical value indicating whether to skip over netcdf files that cannot be downloaded from the server. This is `FALSE` by default, so that `getProfiles()` will raise an error if it is impossible to re-download an outdated file. This is not unexpected with built-in index files, e.g. `data(index)`, because this package cannot be updated every time the Argo servers change the names of netcdf files. However, users are commonly working with index files they created with `getIndex()`, so they ought to be up-to-date.

quiet	logical value; use FALSE to show more verbose information when downloading files. (Even if quiet is TRUE, problems will still be reported.)
debug	integer value indicating level of debugging. If this is less than 1, no debugging is done. Otherwise, some functions will print debugging information. If a function call fails, the first step should be to rerun the function with debug=1, to see if the output suggests a problem in the call.

Details

It should be noted that the constructed server URL follows a different pattern on the USGODAE an Ifremer servers, and so if some other server is used, the URL may be wrong, leading to an error. Similarly, if the patterns on these two servers change, then getProfiles() will fail. Users who encounter such problems are requested to report them to the authors.

If a particular data file cannot be downloaded after multiple trials, then the behaviour depends on the value of the skip argument. If that is TRUE then a NA value is inserted in the corresponding spot in the return value, but if it is FALSE (the default), then an error is reported. Note that readProfiles() skips over any such NA entries, while reporting their positions within index. For more on this function, see Kelley et al. (2021).

Value

An object of class argoFloats with type="profiles", the data slot of which contains two items: url, which holds the URLs from which the NetCDF files were downloaded, and file, which holds the path names of the downloaded files; the latter is used by readProfiles().

Author(s)

Dan Kelley

References

Kelley, D. E., Harbin, J., & Richards, C. (2021). argoFloats: An R package for analyzing Argo data. Frontiers in Marine Science, (8), 636922. doi:10.3389/fmars.2021.635922

---

hexToBits *Convert Hexadecimal Digit to Integer Vector*

Description

hexToBits converts a string holding hexadecimal digits to a sequence of integers 0 or 1, for the bits. This is mainly for for use within showQCTests().

Usage

```
hexToBits(hex)
```

Arguments

hex	a vector of character values corresponding to a sequence of one or more hexadecimal digits (i.e. "0" through "9","a" through "f", or "A" through "F").

Value

An integer vector holding the bits as values 0 or 1. The inverse of 'mathematical' order is used, as is the case for the base R function rawToBits(); see the "Examples".

Author(s)

Jaimie Harbin and Dan Kelley

Examples

```
library(argoFloats)
hexToBits("3") # 1 1 0 0
hexToBits("4000") # 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0
```

---

index *A Sample Index of Profiles*

Description

This was created by subsetting a global index to the Argo profiles that were within a 200km radius of Marsh Harbour, Abaco Island, Bahamas, using the following code.

```
indexAll <- getIndex()
index <- subset(indexAll,
    circle=list(longitude=-77.06, latitude=26.54, radius=200))
```

Caveat about out-of-date index files

Note that the NetCDF files on Argo repositories are changeable, not just in content, but also in file name. For example, the data acquired in a given profile of a given float may initially be provided in real-time mode (with a file name containing an "R" as the first or second character), but later be replaced later with a delayed-mode file (with a "D" in the first or second character). Since index files name data files directly, this means that index files can become out-of-date, containing references to netcdf files that no longer exist on the server. This applies to the sample index files provided with this package, and to user files, and it explains why getProfiles() skips over files that cannot be downloaded.

See Also

Other datasets provided with argoFloats: indexBgc, indexDeep, indexSynthetic

Examples

```
library(argoFloats)
data(index)
plot(index, bathymetry = FALSE)
```

---

indexBgc *A Sample Index of Biogeochemical-Argo Profiles*

Description

This was created by subsetting a global index to the BGC Argo profiles that were within a 300km radius of Marsh Harbour, Abaco Island, Bahamas, using the following code.

```
library(argoFloats)
indexAll <- getIndex("bgc")
indexBgc <- subset(indexAll,
    circle=list(longitude=-77.06, latitude=26.54, radius=300))
```

Caveat about out-of-date index files

Note that the NetCDF files on Argo repositories are changeable, not just in content, but also in file name. For example, the data acquired in a given profile of a given float may initially be provided in real-time mode (with a file name containing an "R" as the first or second character), but later be replaced later with a delayed-mode file (with a "D" in the first or second character). Since index files name data files directly, this means that index files can become out-of-date, containing references to netcdf files that no longer exist on the server. This applies to the sample index files provided with this package, and to user files, and it explains why `getProfiles()` skips over files that cannot be downloaded.

See Also

Other datasets provided with argoFloats: `index`, `indexDeep`, `indexSynthetic`

Examples

```
library(argoFloats)
data(indexBgc)
plot(indexBgc, bathymetry = FALSE)
summary(indexBgc)
unique(indexBgc[["parameters"]])
```

indexDeep *A Sample Index of Deep Argo*

Description

This was created by subsetting a global index to the deep Argo profiles that were within a 800km radius of Antarctica (67S,105E), using the following code.

```
library(argoFloats)
subset <- subset(getIndex(), deep=TRUE)
sub2 <- subset(subset, circle=list(longitude=105, latitude=-67, radius=800))
```

Caveat about out-of-date index files

Note that the NetCDF files on Argo repositories are changeable, not just in content, but also in file name. For example, the data acquired in a given profile of a given float may initially be provided in real-time mode (with a file name containing an "R" as the first or second character), but later be replaced later with a delayed-mode file (with a "D" in the first or second character). Since index files name data files directly, this means that index files can become out-of-date, containing references to netcdf files that no longer exist on the server. This applies to the sample index files provided with this package, and to user files, and it explains why `getProfiles()` skips over files that cannot be downloaded.

See Also

Other datasets provided with argoFloats: `index`, `indexBgc`, `indexSynthetic`

Examples

```
library(argoFloats)
data(indexDeep)
plot(indexDeep, bathymetry = FALSE)
summary(indexDeep)
```

indexSynthetic *A Sample Index of Synthetic Profiles*

Description

This was created by subsetting a global index to the BGC Argo profiles that were within a 300km radius of Marsh Harbour, Abaco Island, Bahamas, using the following code.

```
library(argoFloats)
indexAll <- getIndex("synthetic")
indexSynthetic <- subset(indexAll,
    circle=list(longitude=-77.06, latitude=26.54, radius=300))
```

This "synthetic" type of index is a replacement for the older "merged" index. See http://www.argodatamgt.org/Data-Mgt- for more on the data file format, the reasons for the change, and the timetable for the transition from "merged".

Caveat about out-of-date index files

Note that the NetCDF files on Argo repositories are changeable, not just in content, but also in file name. For example, the data acquired in a given profile of a given float may initially be provided in real-time mode (with a file name containing an "R" as the first or second character), but later be replaced later with a delayed-mode file (with a "D" in the first or second character). Since index files name data files directly, this means that index files can become out-of-date, containing references to netcdf files that no longer exist on the server. This applies to the sample index files provided with this package, and to user files, and it explains why `getProfiles()` skips over files that cannot be downloaded.

See Also

Other datasets provided with argoFloats: `index`, `indexBgc`, `indexDeep`

Examples

```
library(argoFloats)
data(indexSynthetic)
plot(indexSynthetic, bathymetry = FALSE)
summary(indexSynthetic)
unique(indexSynthetic[["parameters"]])
```

---

mapApp *Interactive App For Viewing Argo Float Positions*

Description

`mapApp()` sets up an interactive "shiny app" session for exploring the spatial-temporal distribution of Argo profiles. The graphical user interface (GUI) is meant to be reasonably self-explanatory, and a button labelled Help yields a pop-up window with more information that might not be evident at first glance. A number of keystrokes exist to assist the user including:
u: undo previous actions
i: zoom in
o: zoom out
n: go north
e: go east
s: go south
w: go west
f: go forward in time
b: go backward in time
r: reset to initial state
h: hold active hover message (press h again to undo)
0: unzoom an area

d: toggle debugging flag
More details are provided in the rest of this documentation page, and in Section 4 of Kelley et al. (2021).

Usage

```
mapApp(
  age = argoDefaultIndexAge(),
  server = argoDefaultServer(),
  destdir = argoDefaultDestdir(),
  colLand = "lightgray",
  debug = 0
)
```

Arguments

age	numeric value indicating how old a downloaded file must be (in days), for it to be considered out-of-date. The default, argoDefaultIndexAge(), limits downloads to once per day, as a way to avoid slowing down a workflow with a download that might take a minute or so. Note that setting age=0 will force a new download, regardless of the age of the local file.
server	character value, or vector of character values, indicating the name of servers that supply argo data to be acquired with getIndex(). If not supplied, the default will be determined with argoDefaultServer(), which uses a value set by options("argoFloats.server"=URL) where URL is an appropriate URL, or "ifremer-https" if no such option was set.
destdir	character value indicating the directory in which to store downloaded files. The default value is to compute this using argoDefaultDestdir(), which returns ~/data/argo by default, although it also provides ways to set other values using options(). Set destdir=NULL if destfile is a filename with full path information. File clutter is reduced by creating a top-level directory called data, with subdirectories for various file types; see "Examples".
colLand	a colour specification for the land.
debug	integer value that controls how much information mapApp() prints to the console as it works. The default value of 0 leads to a fairly limited amount of printing, while higher values lead to more information. This information can be helpful in diagnosing problems or bottlenecks.

Details

mapApp() uses getIndex() to download index files from an Argo server the first time it runs. This can be slow, taking a minute or more, depending on the internet connection and load on the server. Once the index files are downloaded, mapApp() categorizes profiles as Core, BGC, or Deep (which may be displayed in any combination, using GUI elements).

The hi-res button will only affect the coastline, not the topography, unless there is a local file named topoWorldFine.rda that contains an alternative topographic information. Here is how to create such a file:

```
library(oce)
topoFile <- download.topo(west=-180, east=180,
                          south=-90, north=90,
                          resolution=10,
                          format="netcdf", destdir=".")
topoWorldFine <- read.topo(topoFile)
save(topoWorldFine, file="topoWorldFine.rda")
unlink(topoFile) # clean up
```

Value

mapApp has no return value, since it creates a shiny app by passing control to shiny::runApp(), which never returns.

Author(s)

Dan Kelley and Jaimie Harbin

References

Kelley, D. E., Harbin, J., & Richards, C. (2021). argoFloats: An R package for analyzing Argo data. Frontiers in Marine Science, (8), 636922. doi:10.3389/fmars.2021.635922

Examples

```
if (interactive()) {
    library(argoFloats)
    mapApp()
}
```

---

merge,argoFloats-method

Merge argoFloats Indices

Description

Merge argoFloats Indices

Usage

```
## S4 method for signature 'argoFloats'
merge(x, y, ...)
```

Arguments

x, y	two argoFloats objects of type index, e.g. as created by getIndex().
...	optional additional objects like x and y.

Value

An argoFloats object of type index.

Author(s)

Dan Kelley

Examples

```
library(argoFloats)
data(index)

# Index of floats within 50km of Abaca Island
C <- subset(index, circle = list(longitude = -77.5, latitude = 27.5, radius = 50))

# Index of floats within a rectangle near Abaca Island
lonRect <- c(-76.5, -76)
latRect <- c(26.5, 27.5)
R <- subset(index, rectangle = list(longitude = lonRect, latitude = latRect))

RC <- merge(C, R)

plot(RC, bathymetry = FALSE)
```

---

plot,argoFloats-method

Plot an argoFloats Object

Description

The action depends on the type of the object, and this is set up by the function that created the object; see "Details". These are basic plot styles, with somewhat limited scope for customization. Since the data with argoFloats objects are easy to extract, users should not find it difficult to create their own plots to meet a particular aesthetic. See "Examples" and Kelley et al. (2021) for more plotting examples.

Usage

```
## S4 method for signature 'argoFloats'
plot(
  x,
  which = "map",
  bathymetry = argoDefaultBathymetry(),
  geographical = 0,
  xlim = NULL,
  ylim = NULL,
```

```
  xlab = NULL,
  ylab = NULL,
  type = "p",
  cex = par("cex"),
  col = par("fg"),
  pch = par("pch"),
  bg = par("bg"),
  eos = "gsw",
  mapControl = NULL,
  profileControl = NULL,
  QCControl = NULL,
  summaryControl = NULL,
  TSControl = NULL,
  debug = 0,
  ...
)
```

Arguments

x	an argoFloats object.
which	a character value indicating the type of plot. The possible choices are "map", "profile", "QC", "summary" and "TS"; see "Details".
bathymetry	an argument used only if which="map", to control whether (and how) to indicate water depth. Note that the default was TRUE until 2021-12-02, but was changed to FALSE on that date, to avoid a bathymetry download, which can be a slow operations. See "Details" for details, and Example 4 for a hint on compensating for the margin adjustment done if an image is used to show bathymetry.
geographical	flag indicating the style of axes for the which="map" case, but only if no projection is called for in the mapControl argument. With geographical=0 (which is the default), the axis ticks are labeled with signed longitudes and latitudes, measured in degrees. The signs are dropped with geographical=1. In the geographical=4 case, signs are also dropped, but hemispheres are indicated by writing S, N, W or E after axis tick labels, except at the equator and prime meridian. Note that this scheme mimics that used by oce::plot,coastline-method().
xlim, ylim	numerical values, each a two-element vector, that set the x and y limits of plot axes, as for plot.default() and other conventional plotting functions.
xlab	a character value indicating the name for the horizontal axis, or NULL, which indicates that this function should choose an appropriate name depending on the value of which. Note that xlab is not obeyed if which="TS", because altering that label can be confusing to the user.
ylab	as xlab, but for the vertical axis.
type	a character value that controls the line type, with "p" for unconnected points, "l" for line segments between undrawn points, etc.; see the docs for par(), If type not specified, it defaults to "p".
cex	a character expansion factor for plot symbols, or NULL, to get an value that depends on the value of which.

col — the colour to be used for plot symbols, or NULL, to get an value that depends on the value of which (see "Details"). If which="TS", then the TSControl argument takes precedence over col.

pch — an integer or character value indicating the type of plot symbol, or NULL, to get a value that depends on the value of which. (See par() for more on specifying pch.)

bg — the colour to be used for plot symbol interior, for pch values that distinguish between the interior of the symbol and the border, e.g. for pch=21.

eos — a character value indicating the equation of state to use if which="TS". This must be "gsw" (the default) or "unesco"; see oce::plotTS().

mapControl — a list that permits particular control of the which="map" case. If provided, it may contain elements named bathymetry (which has the same effect as the parameter bathymetry), colLand (which indicates the colour of the land), and projection (which may be FALSE, meaning to plot longitude and latitude on rectilinear axes, TRUE, meaning to plot with oce::mapPlot(), using Mollweide projection that is suitable mainly for world-scale views, or a character value that will be supplied to oce::mapPlot(). If a projection is used, then the positions of the Argo floats are plotted with oce::mapPoints(), rather than with points(), and if the user wishes to locate points with mouse clicks, then oce::mapLocator() must be used instead of locator(). If bathymetry is not contained in mapControl, it defaults to FALSE, and if projection is not supplied, it defaults to FALSE. Note that mapControl takes precedence over the bathymetry argument, if both are provided. Also note that, at present, bathymetry cannot be shown with map projections.

profileControl — a list that permits control of the which="profile" case. If provided, it may contain elements named parameter (a character value naming the quantity to plot on the x axis), ytype (a character value equal to either "pressure" or "sigma0") and connect (a logical value indicating whether to skip across NA values if the type argument is "l", "o", or "b"). If profileControl is not provided, it defaults to list(parameter="temperature", ytype="pressure", connect=FALSE). Alternatively, if profileControl is missing any of the three elements, then they are given defaults as in the previous sentence.

QCControl — a list that permits control of the which="QC" case. If provided, it may contain an element named parameter, a character value naming the quantity for which the quality-control information is to be plotted, and an element named dataStateIndicator, a logical value controlling whether to add a panel showing this quantity (see Reference Table 6 of reference 1 to learn more about what is encoded in dataStateIndicator). If not provided, QCControl defaults to list(parameter="temperature",dataStateIndicator=FALSE).

summaryControl — a list that permits control of the which="summary". If provided, it should contain an element named items, a character vector naming the items to be shown. The possible entries in this vector are "dataStateIndicator" (see Reference Table 6 of reference 1, for more information on this quantity), "length" (the number of levels in the profile), "deepest" (the highest pressure recorded), "longitude" and "latitude". If summaryControl is not provided, all of these will be shown. If all the elements of x have the same ID, then the top panel will have ticks on its top axis, indicating the cycle.

TSControl	a list that permits control of the which="TS" case, and is ignored for the other cases. If TSControl is supplied, then other arguments that normally control plots are ignored. TSControl may have elements named drawByCycle, cex, col, pch, lty, lwdandtype. The first of these is a logical value indicating whether to plo trol=list(drawByCycle=TRUE,col=1:2,type="l")‘ indicates to draw lines for the individual cycles, with the first being black, the second red, the third black, etc. **FIXME: THIS IS NOT CODED YET.**
debug	an integer specifying the level of debugging.
...	extra arguments passed to the plot calls that are made within this function.

Details

The various plot types are as follows.

- For which="map", a map of profile locations is created if subtype is equal to cycles, or a rectangle highlighting the trajectory of a float ID is created when subtype is equal to trajectories. This only works if the type is "index" (meaning that x was created by getIndex() or a subset of such an object, created with subset,argoFloats-method()), or argos (meaning that x was created with readProfiles(). The plot range is auto-selected. If the ocedata package is available, then its coastlineWorldFine dataset is used to draw a coastline (which will be visible only if the plot region is large enough); otherwise, if the oce package is available, then its coastlineWorld dataset is used. The bathymetry argument controls whether (and how) to draw a map underlay that shows water depth. There are three possible values for bathymetry:
  1. FALSE (the default, via argoDefaultBathymetry()), meaning not to draw bathymetry;
  2. TRUE, meaning to draw bathymetry as an image, using data downloaded with oce::download.topo(). Example 4 illustrates this, and also shows how to adjust the margins after the plot, in case there is a need to add extra graphical elements using points(), lines(), etc.
  3. A list with items controlling both the bathymetry data and its representation in the plot (see Example 4). Those items are:
     (a) source, a mandatory value that is either (a) the string "auto" (the default) to use oce::download.topo() to download the data or (b) a value returned by oce::read.topo().
     (b) contour, an optional logical value (with FALSE as the default) indicating whether to represent bathymetry with contours (with depths of 100m, 200m, 500m shown, along with 1km, 2km up to 10km), as opposed to an image;
     (c) colormap, ignored if contour is TRUE, an optional value that is either the string "auto" (the default) for a form of GEBCO colors computed with oce::oceColorsGebco(), or a value computed with oce::colormap() applied to the bathymetry data; and
     (d) palette, ignored if contour is TRUE, an optional logical value (with TRUE as the default) indicating whether to draw a depth-color palette to the right of the plot. Note that the default value for bathymetry is set by a call to argoDefaultBathymetry(), and that this second function can only handle possibilities 1 and 2 above.
- For which="profile", a profile plot is created, showing the variation of some quantity with pressure or potential density anomaly, as specified by the profileControl argument.
- For which="QC", two time-series panels are shown, with time being that recorded in the individual profile in the dataset. An additional argument named parameter must be given, to name the quantity of interest. The function only works if x is an argoFloats object created

with readProfiles(). The top panel shows the percent of data flagged with codes 1 (meaning good data), 2 (probably good), 5 (changed) or 8 (estimated), as a function of time (lower axis) and (if all cycles are from a single Argo float) cycle number (upper axis, with smaller font). Thus, low values on the top panel reveal profiles that are questionable. Note that if all of data at a given time have flag 0, meaning not assessed, then a quality of 0 is plotted at that time. The bottom panel shows the mean value of the parameter in question regardless of the flag value.

- For which="summary", one or more time-series panels are shown in a vertical stack. If there is only one ID in x, then the cycle values are indicated along the top axis of the top panel. The choice of panels is set by the summaryControl argument.
- For which="TS", a TS plot is created, by calling plotArgoTS() with the specified x, xlim, ylim, xlab, ylab, type, cex, col, pch, bg, eos, and TSControl, along with debug-1. See the help for plotArgoTS() for how these parameters are interpreted.

Value

None (invisible NULL).

Author(s)

Dan Kelley and Jaimie Harbin

References

1. Carval, Thierry, Bob Keeley, Yasushi Takatsuki, Takashi Yoshida, Stephen Loch, Claudia Schmid, and Roger Goldsmith. Argo User's Manual V3.3. Ifremer, 2019. doi:10.13155/29825
2. Kelley, D. E., Harbin, J., & Richards, C. (2021). argoFloats: An R package for analyzing Argo data. Frontiers in Marine Science, (8), 636922. doi:10.3389/fmars.2021.635922

Examples

```
# Example 1: map profiles in index
library(argoFloats)
data(index)
plot(index)

# Example 2: as Example 1, but narrow the margins and highlight floats
# within a circular region of diameter 100 km.
oldpar <- par(no.readonly = TRUE)
par(mar = c(2, 2, 1, 1), mgp = c(2, 0.7, 0))
plot(index)
lon <- index[["longitude"]]
lat <- index[["latitude"]]
near <- oce::geodDist(lon, lat, -77.06, 26.54) < 100
R <- subset(index, near)
points(R[["longitude"]], R[["latitude"]],
    cex = 0.6, pch = 20, col = "red"
)
par(oldpar)
```

```
# Example 3: TS of a built-in data file.
f <- system.file("extdata", "SR2902204_131.nc", package = "argoFloats")
a <- readProfiles(f)
oldpar <- par(no.readonly = TRUE)
par(mar = c(3.3, 3.3, 1, 1), mgp = c(2, 0.7, 0)) # wide margins for axis names
plot(a, which = "TS")
par(oldpar)

# Example 4: map with bathymetry. Note that par(mar) is adjusted
# for the bathymetry palette, so it must be adjusted again after
# the plot(), in order to add new plot elements.
# This example specifies a coarse bathymetry dataset that is provided
# by the 'oce' package.  In typical applications, the user will use
# a finer-scale dataset, either by using bathymetry=TRUE (which
# downloads a file appropriate for the plot view), or by using
# an already-downloaded file.
data(topoWorld, package = "oce")
oldpar <- par(no.readonly = TRUE)
par(mar = c(2, 2, 1, 2), mgp = c(2, 0.7, 0)) # narrow margins for a map
plot(index, bathymetry = list(source = topoWorld))
# For bathymetry plots that use images, plot() temporarily
# adds 2.75 to par("mar")[4] so the same must be done, in order
# to correctly position additional points (shown as black rings).
par(mar = par("mar") + c(0, 0, 0, 2.75))
points(index[["longitude"]], index[["latitude"]],
    cex = 0.6, pch = 20, col = "red"
)
par(oldpar)

# Example 5. Simple contour version, using coarse dataset (ok on basin-scale).
# Hint: use oce::download.topo() to download high-resolution datasets to
# use instead of topoWorld.
oldpar <- par(no.readonly = TRUE)
par(mar = c(2, 2, 1, 1))
data(topoWorld, package = "oce")
plot(index, bathymetry = list(source = topoWorld, contour = TRUE))
par(oldpar)

# Example 6. Customized map, sidestepping plot,argoFloats-method().
lon <- topoWorld[["longitude"]]
lat <- topoWorld[["latitude"]]
asp <- 1 / cos(pi / 180 * mean(lat))
# Limit plot region to float region.
xlim <- range(index[["longitude"]])
ylim <- range(index[["latitude"]])
# Colourize 1km, 2km, etc, isobaths.
contour(
    x = lon, y = lat, z = topoWorld[["z"]], xlab = "", ylab = "",
    xlim = xlim, ylim = ylim, asp = asp,
    col = 1:6, lwd = 2, levels = -1000 * 1:6, drawlabels = FALSE
)
# Show land
data(coastlineWorldFine, package = "ocedata")
```

```
polygon(coastlineWorldFine[["longitude"]],
    coastlineWorldFine[["latitude"]],
    col = "lightgray"
)
# Indicate float positions.
points(index[["longitude"]], index[["latitude"]], pch = 20)


# Example 7: Temperature profile of the 131st cycle of float with ID 2902204
a <- readProfiles(system.file("extdata", "SR2902204_131.nc", package = "argoFloats"))
oldpar <- par(no.readonly = TRUE)
par(mfrow = c(1, 1))
par(mgp = c(2, 0.7, 0)) # mimic the oce::plotProfile() default
par(mar = c(1, 3.5, 3.5, 2)) # mimic the oce::plotProfile() default
plot(a, which = "profile")
par(oldpar)

# Example 8: As Example 7, but showing temperature dependence on potential density anomaly.
a <- readProfiles(system.file("extdata", "SR2902204_131.nc", package = "argoFloats"))
oldpar <- par(no.readonly = TRUE)
par(mgp = c(2, 0.7, 0)) # mimic the oce::plotProfile() default
par(mar = c(1, 3.5, 3.5, 2)) # mimic the oce::plotProfile() default
plot(a, which = "profile", profileControl = list(parameter = "temperature", ytype = "sigma0"))
par(oldpar)
```

---

plotArgoTS *Plot a TS diagram for an argoFloats object*

Description

Plot a TS diagram for an argoFloats object that was created with `readProfiles()`. This function is called by `plot,argoFloats-method()`, but may also be called directly.

Usage

```
plotArgoTS(
  x,
  xlim = NULL,
  ylim = NULL,
  type = "p",
  cex = 1,
  col = NULL,
  pch = 1,
  bg = "white",
  eos = "gsw",
  TSControl = NULL,
```

```
  debug = 0,
  ...
)
```

Arguments

x	an argoFloats object that was created with readProfiles().
xlim, ylim	optional limits of salinity and temperature axes; if not provided, the plot region will be large enough to show all the data.
type	character value indicating the of plot. This has the same meaning as for general R plots: "p" for points, "l" for lines, etc. Note that lines are joined up between cycles, unless the TSControl parameter indicates otherwise.
cex, col, pch, bg	values that have the same meaning as for general R plots if TSControl$groupByCycle is FALSE, except that if col is not provided by the user and if type is "l", then the points are colour-coded to indicate the value of data-quality flags. Black symbols indicate good data, i.e. data for which the flags for pressure, salinity and temperature are all in the set (1, 2, 5, 8). Red is used for bad data, with any of these three variables being flagged in the set (3, 4, 6, 7). And, finally gray is used if data that have not been assessed for quality, with the flags for all three of these variables being 0. (For more on flags, see Reference Table 2 in Section 3.2 of reference 1.)
eos	character value, either "gsw" (the default) for the Gibbs-Seawater (TEOS-10) equation of state or "unesco" for the 1980s-era UNESCO equation of state.
TSControl	an optional list that may have a logical element named groupByCycle, meaning to group the data by cycle. If TSControl is not provided, it is set to list(groupByDefault=FALSE). In grouped cases, the values of cex, col, and pch are passed to rep() to achieve the same length as the number of cycles in x. This can be useful in distinguishing between cycles.
debug	an integer controlling how much information is to be printed during operation. Use 0 for silent work, 1 for some information and 2 for more information. Note that plot,argoFloats-method() reduces its debug value by 1 before passing to plotArgoTS().
...	extra arguments provided to oce::plotTS().

Author(s)

Dan Kelley and Jaimie Harbin

References

1. Argo Data Management. "Argo User's Manual." Ifremer, July 5, 2022. https://doi.org/10.13155/29825.

R3901602_163.nc *Sample Argo File (Real-Time Core Data)*

Description

This is NetCDF file for real-time data for cycle 163 of Argo float 3901602, downloaded from https://data-argo.ifremer.fr/dac/coriolis/3901602/profiles/R3901602_163.nc on 2021 March 7.

See Also

Other raw datasets: D4900785_048.nc, SD5903586_001.nc, SR2902204_131.nc

Examples

```
library(argoFloats)
a <- readProfiles(system.file("extdata", "R3901602_163.nc", package = "argoFloats"))
summary(a)
summary(a[[1]])
```

readProfiles *Read Argo Profiles From Local Files*

Description

This works with either a vector of NetCDF files, or a argoFloats object of type "profiles", as created by getProfiles(). During the reading, argo profile objects are created with oce::read.argo() or a replacement function provided as the FUN argument.

Usage

```
readProfiles(
  profiles,
  FUN,
  destdir = argoDefaultDestdir(),
  quiet = FALSE,
  debug = 0
)
```

Arguments

profiles	either (1) a character vector that holds the names of NetCDF files or (2) an argoFloats object created by getProfiles(). In the first case, any items that start with "ftp:" are taken to represent the full paths to remote files, and these first downloaded to the destdir directory using getProfileFromUrl().
FUN	a function that reads the NetCDF files in which the argo profiles are stored. If FUN not provided, then it defaults to oce::read.argo(). Only experts should consider anything other than this default, or a wrapper to it.
destdir	character value indicating the directory in which to store downloaded files. The default value is to compute this using argoDefaultDestdir(), which returns ~/data/argo by default, although it also provides ways to set other values using options(). Set destdir=NULL if destfile is a filename with full path information. File clutter is reduced by creating a top-level directory called data, with subdirectories for various file types; see "Examples".
quiet	logical value; use FALSE to show more verbose information when downloading files. (Even if quiet is TRUE, problems will still be reported.)
debug	an integer specifying the level of debugging. If this is zero, the work proceeds silently. If it is 1, a small amount of debugging information is printed. Note that debug-1 is passed to oce::read.argo(), which actually reads the file, and so it will print messages if debug exceeds 1.

Details

By default, warnings are issued about any profiles in which 10 percent or more of the measurements are flagged with a quality-control code of 0, 3, 4, 6, 7, or 9 (see the applyQC() documentation for the meanings of these codes). For more on this function, see section 2 of Kelley et al. (2021).

Value

An argoFloats object with type="argos", in which the data slot contains a list named argos that holds objects that are created by oce::read.argo().

Author(s)

Dan Kelley

References

Kelley, D. E., Harbin, J., & Richards, C. (2021). argoFloats: An R package for analyzing Argo data. Frontiers in Marine Science, (8), 636922. doi:10.3389/fmars.2021.635922

Examples

```
# # Omit this, because rhub errors out, evidently because it is running donttest blocks.
# # Example 1: read 5 profiles and plot TS for the first, in raw and QC-cleaned forms.
# # This example involves downloading to a local repository, so it is not run on CRAN.
#
```

```
# library(argoFloats)
# data(index)
# index1 <- subset(index, 1)
# profiles <- getProfiles(index1)
# raw <- readProfiles(profiles)
# clean <- applyQC(raw)
# par(mfrow=c(1, 2))
# file <- gsub(".*/", "",  profiles[[1]])
# aWithNA <- clean[[1]]
# oce::plotTS(raw[[1]], eos="unesco", type="o")
# mtext(file, cex=0.7*par("cex"))
# aWithoutNA <- raw[[1]]
# oce::plotTS(clean[[1]], eos="unesco", type="o")
# mtext(paste(file, "\n (after applying QC)"), cex=0.7*par("cex"))
#

#
# Read from a local file
f <- system.file("extdata", "SR2902204_131.nc", package = "argoFloats")
p <- readProfiles(f)
```

SD5903586_001.nc *Sample Argo File (Delayed Synthetic Data)*

Description

This is the NetCDF file for cycle 1 of Argo float 5903586, downloaded from ftp://usgodae.org/pub/outgoing/argo/dac on 2020 June 24 (this URL appears to be unreliable). As its filename indicates, it holds "synthetic" data in "delayed" mode. The oxygen values are adjusted by 16%, as is shown here, and in the documentation for useAdjusted().

See Also

Other raw datasets: D4900785_048.nc, R3901602_163.nc, SR2902204_131.nc

Examples

```
library(argoFloats)
a <- readProfiles(system.file("extdata", "SD5903586_001.nc", package = "argoFloats"))
a1 <- a[[1]]
# Illustrate oxygen adjustment
p <- a1[["pressure"]]
O <- a1[["oxygen"]]
Oa <- a1[["oxygenAdjusted"]]
Percent <- 100 * (Oa - O) / O
hist(Percent, main = "Oxygen adjustment")
```

show,argoFloats-method *Show Information About argoFloats Object*

Description

This produces a one-line explanation of the contents of the object, that typically indicates the type and the number of items referenced by the object. Those items depend on the type of object: URLs if `metadata$type` is `"index"`, local filenames if `"profiles"`, or oce-created argo objects if `"argos"`. As with other R `show()` methods, it may be invoked in an interactive session just by typing the object, or by using `print()`; see the Examples.

Usage

```
## S4 method for signature 'argoFloats'
show(object)
```

Arguments

`object`	an `argoFloats` object.

Value

None (invisible NULL).

Author(s)

Jaimie Harbin

Examples

```
library(argoFloats)
data(index)
show(index)
print(index)
index
```

showQCTests *Show Real-Time QC Test Results For an Argo Object*

Description

showQCTests prints a summary of the quality-control (QC) tests (if any) that were performed on an Argo profile in real-time (**Caution**: any tests completed and/or failed on delayed mode data are not recorded. This function also assumes tests performed or failed are recorded once, otherwise it produces a warning). It uses hexToBits() to decode the hexadecimal values that may be stored in historyQCTest. From there it pairs the determined test values with the appropriate actions, QC Tests performed or QC Tests failed, found in historyAction within the metadata slot of an individual Argo profile, as read directly with oce::read.argo() or indirectly with readProfiles(), the latter being illustrated in the "Examples" section below. The "Details" section provides an explanation of how showQCTests works at a low level, as an adjunct to the Argo documentation. See section 3.3 of Kelley et al. (2021) for more on this function.

Usage

```
showQCTests(x, style = "brief")
```

Arguments

x	an oce::argo object, as read directly with oce::read.argo() or as extracted from the return value of a call to readProfiles(), as in the "Examples".
style	a character value governing the output printed by showQCFlags, either "brief" (the default) for a single line stating all the tests by numbers, followed by lines giving the number and description of all failed tests, or "full" for a listing of each test that was performed, with an indication of whether x passes or fails it.

Details

The format used in the historyQCTest and historyAction elements of the metadata slot of an oce::argo object is mentioned in Sections 2.2.7, 2.3.7, 5.1, 5.3 and 5.4 of reference 1, in which they are called HISTORY_QCTEST and HISTORY_ACTION, respectively. Both of these things are vectors of character values, with the entries within historyAction providing names for the entries within historyQCTest.

In the context of showQCTests, the focus is on the element of historyAction that equals "QCP$" (which maps to the element of historyQCTest that specifies the QC tests that were performed) and "QCF$" (which maps to the results of those tests). These mapped elements are character values providing hexadecimal digits that are decoded with hexToBits(), possibly after lengthening the historyQCTest value matching"QCF$" by adding "0" digits on the left to make the length be the same as that of the historyQCTest value matching '"QCP$".

The bits decoded from the relevant elements of historyQCTest correspond to QC tests as indicated in the following table. This is based on Table 11 of reference 1, after correcting the "Number" for test 18 from 261144 to 262144, because the former is not an integral power of 2, suggesting a typo

in the manual (since the whole point of the numerical scheme is that the individual bits map to individual tests).

Test	Number	Meaning
1	2	Platform Identification test
2	4	Impossible Date test
3	8	Impossible Location test
4	16	Position on Land test
5	32	Impossible Speed test
6	64	Global Range test
7	128	Regional Global Parameter test
8	256	Pressure Increasing test
9	512	Spike test
10	1024	Top and Bottom Spike test (obsolete)
11	2048	Gradient test
12	4096	Digit Rollover test
13	8192	Stuck Value test
14	16384	Density Inversion test
15	32768	Grey List test
16	65536	Gross Salinity or Temperature Sensor Drift test
17	131072	Visual QC test
18	262144	Frozen profile test
19	524288	Deepest pressure test
20	1048576	Questionable Argos position test
21	2097152	Near-surface unpumped CTD salinity test
22	4194304	Near-surface mixed air/water test
23	8388608	Interim rtqc flag scheme for data deeper than 2000 dbar
24	16777216	Interim rtqc flag scheme for data from experimental sensors
25	33554432	MEDD test

Value

This function returns nothing; its action is in the printing of results.

Author(s)

Jaimie Harbin and Dan Kelley

References

1. Carval, Thierry, Bob Keeley, Yasushi Takatsuki, Takashi Yoshida, Stephen Loch, Claudia Schmid, and Roger Goldsmith. Argo User's Manual V3.3. Ifremer, 2019. doi:10.13155/29825
2. Kelley, D. E., Harbin, J., & Richards, C. (2021). argoFloats: An R package for analyzing Argo data. Frontiers in Marine Science, (8), 636922. doi:10.3389/fmars.2021.635922

Examples

```
library(argoFloats)
```

```
a <- readProfiles(system.file("extdata", "D4900785_048.nc", package = "argoFloats"))
showQCTests(a[[1]])
```

SR2902204_131.nc *Sample Argo File (Real-Time Synthetic Data)*

Description

This is the NetCDF file for cycle 131 of Argo float 2902204, downloaded from ftp://ftp.ifremer.fr/ifremer/argo/dac on 2020 June 24. As its filename indicates, it holds the "synthetic" version of "real-time" data.

See Also

Other raw datasets: D4900785_048.nc, R3901602_163.nc, SD5903586_001.nc

Examples

```
library(argoFloats)
a <- readProfiles(system.file("extdata", "SR2902204_131.nc", package = "argoFloats"))
summary(a)
summary(a[[1]])
```

---

subset,argoFloats-method
Subset an argoFloats Object

Description

Return a subset of an argoFloats object. This applies only to objects of type "index", as created with getIndex() or subset() acting on such a value, or of type "argos", as created with readProfiles(). (It cannot be used on objects of type "profiles", as created with getProfiles().) There are two ways to use subset(). **Method 1.** supply the subset argument. This may be a logical vector indicating which entries to keep (by analogy to the base-R subset() function) or it may be an integer vector holding the indices of entries to be retained. **Method 2.** do not supply subset. In this case, the action is determined by the third (...) argument; see 'Details'.

Usage

```
## S4 method for signature 'argoFloats'
subset(x, subset = NULL, ...)
```

Arguments

x	an argoFloats object as created by getIndex().
subset	optional numerical or logical vector that indicates which indices of x@data$index to keep (example 1).
...	the first entry here must be either (a) a list named circle, rectangle, polygon, parameter, time, institution, ID,ocean,dataMode,cycle, direction, profile, or section. (examples 2 through 8, and 10 through 17), or (b) a logical value named deep (example 9). Optionally, this entry may be followed by second entry named silent, which is a logical value indicating whether to prevent the printing of messages that indicate the number (and percentage) of data that are kept during the subsetting operation. See "Details" and "Examples".

Details

The possibilities for the ... argument are as follows.

1. An integer vector giving indices to keep.
2. A list named circle with numeric elements named longitude, latitude and radius. The first two give the center of the subset region, and the third gives the radius of that region, in kilometers.
3. A list named rectangle that has elements named longitude and latitude, two-element numeric vectors giving the western and eastern, and southern and northern limits of the selection region.
4. A list named polygon that has elements named longitude and latitude that are numeric vectors specifying a polygon within which profiles will be retained. The polygon must not be self-intersecting, and an error message will be issued if it is. If the polygon is not closed (i.e. if the first and last points do not coincide) the first point is pasted onto the end, to close it.
5. A vector or list named parameter that holds character values that specify the names of measured parameters to keep. See section 3.3 of Reference 1 for a list of parameters.
6. A list named time that has elements from and to that are either POSIXt times, or character strings that subset() will convert to POSIXt times using as.POSIXct() with tz="UTC".
7. A list named institution that holds a single character element that names the institution. The permitted values are: "AO" for Atlantic Oceanographic and Meteorological Laboratory; "BO" for British Oceanographic Data Centre; "CS" for Commonwealth Scientific and Industrial Research Organization; "HZ" for China Second Institute of Oceanography; "IF" for Ifremer, France; "IN" for India National Centre for Ocean Information Services; "JA" for Japan Meteorological Agency; "KM" for Korea Meteorological Agency; "KO" for Korea Ocean Research and Development Institute; "ME" for Marine Environment Data Section; and "NM" for National Marine Data & Information Service.
8. A list named deep that holds a logical value indicating whether argo floats are deep argo (i.e. profiler_type 849, 862, and 864).
9. A list named ID that holds a character value specifying a float identifier.
10. A list named ocean that holds a single character element that names the ocean. The permitted values are: "A" for Atlantic Ocean Area, from 70 W to 20 E, "P" for Pacific Ocean Area, from 145 E to 70 W, and "I" for Indian Ocean Area, from 20 E to 145 E.

11. A character value named `dataMode`, equal to either `realtime` or `delayed`, that selects whether to retain real-time data or delayed data. There are two possibilities, depending on the `type` of the `x` argument. **Case 1.** If `x` is of `type="index"`, then the subset is done by looking for the letters `R` or `D` in the source filename. Note that a file in the latter category may contain some profiles that are of delayed mode *and also* some profiles that are of `realtime` or `adjusted` mode. **Case 2.** If `x` is of type `argos`, then the subset operation is done for each profile within the dataset. Sometimes this will yield data arrays with zero columns.
12. An integer or character value named `cycle` that specifies which cycles are to be retained. This is done by regular-expression matching of the filename, looking between the underline character (`"_"`) and the suffix (`.nc`), but note that the expression is made up of a compulsory component comprising 3 or 4 digits, and an optional component that is either blank or the character `"D"` (which designates a descending profile). Thus, `001` will match both `*_001.nc` and `*_001D.nc`. Note this can be used for both `"index"` and `"argos"` types.
13. A character value named `direction`, equal to either "descent" or "ascent", that selects whether to retain data from the ascent or decent phase.
14. An integer value named `profile` that selects which profiles to retain. Note that this type of subset is possible only for objects of type `"argos"`.
15. An integer value named `cycle` that selects which cycles to retain.
16. A character value named `dataStateIndicator`, equal to either "0A", "1A", "2B", "2B+", "2C", "2C+", "3B", or "3C" that selects which `dataStateIndicator` to keep (see table 6 of Reference 1 for details of these codes). This operation only works for objects of type `"argos"`.
17. A list named `section` that has four elements: `longitude`,`latitude`, `width`, and `segments`. The first two of these are numeric vectors that define the spine of the section, in degrees East and North, respectively. These are both mandatory, while both `width` and `segments` are optional. If given, `width` is taken as maximal distance between an Argo and the spine, for that float to be retained. If `width` is not given, it defaults to 100km. If given, `segments` is either `NULL` or a positive integer. Setting `segments` to `NULL` will cause the float-spine distance to be computed along a great-circle route. By contrast, if `segments` is an integer, then the spine is traced using `stats::approx()`, creating `segments` new points. If `segments` is not provided, it defaults to 100.

In all cases, the notation is that longitude is positive for degrees East and negative for degrees West, and that latitude is positive for degrees North and negative for degrees South. For more on this function, see Kelley et al. (2021).

Value

An `argoFloats` object, restricted as indicated.

Author(s)

Dan Kelley and Jaimie Harbin

References

1. Carval, Thierry, Bob Keeley, Yasushi Takatsuki, Takashi Yoshida, Stephen Loch, Claudia Schmid, and Roger Goldsmith. Argo User's Manual V3.3. Ifremer, 2019. doi:10.13155/29825.
2. Kelley, D. E., Harbin, J., & Richards, C. (2021). argoFloats: An R package for analyzing Argo data. Frontiers in Marine Science, (8), 636922. doi:10.3389/fmars.2021.635922

Examples

```
library(argoFloats)
data(index)
data(indexSynthetic)

# Subset to the first 3 profiles in the (built-in) index
indexFirst3 <- subset(index, 1:3)

# Subset to a circle near Abaco Island
indexCircle <- subset(index, circle = list(longitude = -77.5, latitude = 27.5, radius = 50))
#
# # NOTE: this example was removed because it requires using tempdir, which
# # is not something we want to encourage, since it goes against the whole idea
# # of caching.
# # Example 2B: subset a 300 km radius around Panama using "maps" package
# # This example requires downloading.
#

# library("maps")
# data(world.cities)
# ai <- getIndex()
# panama <- subset(world.cities, name=="Panama")
# index1 <- subset(ai, circle=list(longitude=panama$long, latitude=panama$lat, radius=200))
#


# Subset to a rectangle near Abaco Island
lonlim <- c(-76.5, -76)
latlim <- c(26.5, 27.5)
indexRectangle <- subset(index, rectangle = list(longitude = lonlim, latitude = latlim))

# Subset to a polygon near Abaco Island
lonp <- c(-77.492, -78.219, -77.904, -77.213, -76.728, -77.492)
latp <- c(26.244, 25.247, 24.749, 24.987, 25.421, 26.244)
indexPolygon <- subset(index, polygon = list(longitude = lonp, latitude = latp))
#
# # Show some of these subsets on a map
# plot(index, bathymetry=FALSE)
# points(index2[["longitude"]], index2[["latitude"]], col=2, pch=20, cex=1.4)
# points(index3[["longitude"]], index3[["latitude"]], col=3, pch=20, cex=1.4)
# rect(lonRect[1], latRect[1], lonRect[2], latRect[2], border=3, lwd=2)
# points(index4[["longitude"]], index4[["latitude"]], col=4, pch=20, cex=1.4)
# polygon(p$longitude, p$latitude, border=4)

# Subset to year 2019
index2019 <- subset(index, time = list(from = "2019-01-01", to = "2019-12-31"))

# Subset to Canadian MEDS data
indexMEDS <- subset(index, institution = "ME")
#
# # NOTE: this example was removed because it requires using tempdir, which
# # is not something we want to encourage, since it goes against the whole idea
```

```
# # of caching.
# # Example 8: subset to a specific ID
# # This example requires downloading.
#

# ai <- getIndex(filename="synthetic")
# index9 <- subset(ai, ID="1900722")
#

#
# # NOTE: this example was removed because it requires using tempdir, which
# # is not something we want to encourage, since it goes against the whole idea
# # of caching.
# # Example 9: subset data to only include deep argo
# # This example requires downloading.
#

# ai <- getIndex(filename="synthetic")
# index8 <- subset(ai, deep=TRUE)
#

#
# # NOTE: this example was removed because it requires using tempdir, which
# # is not something we want to encourage, since it goes against the whole idea
# # of caching.
# # Example 10: subset data by ocean
# # This example requires downloading.
#

# ai <- getIndex()
# index10 <- subset(ai, circle=list(longitude=-83, latitude=9, radius=500))
# plot(index10, which="map", bathymetry=FALSE)
# atlantic <- subset(index10, ocean="A") # Subsetting for Atlantic Ocean
# pacific <- subset(index10, ocean="P")
# points(atlantic[["longitude"]], atlantic[["latitude"]], pch=20, col=2)
# points(pacific[["longitude"]], pacific[["latitude"]], pch=20, col=3)
#

#
# # NOTE: there is no example 11.
#
# # NOTE: this example was removed because it requires using tempdir, which
# # is not something we want to encourage, since it goes against the whole idea
# # of caching.
# # Example 11: subset by delayed time
# # This example requires downloading.
#

# data(indexBgc)
# index11 <- subset(indexBgc, dataMode="delayed")
# profiles <- getProfiles(index11)
# argos <- readProfiles(profiles)
# oxygen <- argos[["oxygen"]][[3]]
```

```
# pressure <- argos[["pressure"]][[3]]
# plot(oxygen, pressure, ylim=rev(range(pressure, na.rm=TRUE)),
#      ylab="Pressure (dbar)", xlab="Oxygen (umol/kg)")
#

#
# # Example 12: subset by cycle
# data(index)
# index12A <- subset(index, cycle="124")
# index12B <- subset(index, cycle=0:2)
# cat("File names with cycle number 124:", paste(index12A[["file"]]), "\n")
# cat("File names with cycle number between 0 and 2:", paste(index12B[["file"]]), "\n")
#
#
# # NOTE: this example was removed because it requires using tempdir, which
# # is not something we want to encourage, since it goes against the whole idea
# # of caching.
# # Example 13: subset by direction
# # This example requires downloading.
#

# library(argoFloats)
# index13A <- subset(getIndex(), deep=TRUE)
# index13B <- subset(index13A, direction="descent")
# head(index13B[["file"]])
#

#
# # NOTE: this example was removed because it requires using tempdir, which
# # is not something we want to encourage, since it goes against the whole idea
# # of caching.
# # Example 14: subset by profile (for argos type)
# # This example requires downloading.
#

# library(argoFloats)
# index14A <- subset(getIndex(filename="synthetic"), ID="5903889")
# index14B <- subset(index14A, cycle="074")
# argos14A <- readProfiles(getProfiles(index14B))
# argos14B <- subset(argos14A, profile=1)
# D <- data.frame(Oxygen = argos14A[["oxygen"]],
# col1= argos14B[["oxygen"]][[1]])
#

#
# # NOTE: this example was removed because it requires using tempdir, which
# # is not something we want to encourage, since it goes against the whole idea
# # of caching.
# # Example 15: subset by cycle (for argos type) to create TS diagram
# # This example requires downloading.
#

# data("index")
```

```
# index15 <- subset(index, ID="1901584")
# profiles <- getProfiles(index15)
# argos <- readProfiles(profiles)
# plot(subset(argos, cycle="147"), which="TS")
#

#
# # NOTE: this example was removed because it requires using tempdir, which
# # is not something we want to encourage, since it goes against the whole idea
# # of caching.
# # Example 16: subset by dataStateIndicator
# # This example requires downloading.
#

# data("index")
# index16 <- subset(index, 1:40)
# argos <- readProfiles(getProfiles(index16))
# argos16A <- subset(argos, dataStateIndicator="2C")
# argos16B <- subset(argos, dataStateIndicator="2B")
#

#
# # Example 17: subset by section to create a map plot
# if (requireNamespace("s2")) {
#     data("index")
#     lon <- c(-78, -77, -76)
#     lat <-c(27.5,27.5,26.5)
#     index17 <- subset(index,
#                       section=list(longitude=lon, latitude=lat, width=50))
#     plot(index17, bathymetry=FALSE)
#     points(lon, lat, pch=21, col="black", bg="red", type="o", lwd=3)
# }

# Subset to profiles with oxygen data
indexOxygen <- subset(indexSynthetic, parameter = "DOXY")

# Subset to profiles with both salinity and 380-nm downward irradiance data
indexSalinityIrradiance <- subset(indexSynthetic, parameter = c("PSAL", "DOWN_IRRADIANCE380"))
```

---

summary,argoFloats-method

Summarize an argoFloats Object

Description

Show some key facts about an argoFloats object.

Usage

```
## S4 method for signature 'argoFloats'
summary(object, ...)
```

Arguments

object	an argoFloats object.
...	Further arguments passed to or from other methods.

Value

None (invisible NULL).

Author(s)

Dan Kelley

Examples

```
library(argoFloats)
data(index)
summary(index)
```

---

useAdjusted *Switch [[and Plot to Focus on Adjusted Data, if Available*

Description

useAdjusted returns a copy of an argoFloats object within which the individual oce::argo objects may have been modified so that future calls to [[,argoFloats-method or plot,argoFloats-method will focus with *adjusted* versions of the data. (Note that this modification cannot be done for fields that lack adjusted values, so in such cases future calls to [[,argoFloats-method or plot,argoFloats-method work with the unadjusted fields.) The procedure is done profile by profile and parameter by parameter. The fallback argument offers a way to "fall back" to unadjusted values, depending on the data-mode (real-time, adjusted or delayed) for individual items; see "Details".

Usage

```
useAdjusted(argos, fallback = FALSE, debug = 0)
```

Arguments

argos	an argoFloats object, as read by readProfiles().
fallback	a logical value indicating whether to 'fall back' from adjusted data to raw data for profiles that are in real-time mode. By default, fallback is FALSE, to focus entirely on adjusted data. See "Details".
debug	an integer that controls whether debugging information is printed during processing. If debug is 0 or less, then no information is printed. If it is 1 then minimal overall information is printed. If it exceeds 1, then information is printed for each Argo cycle contained in argos.

Details

There are two cases. First, if fallback is FALSE (which the default) then the focus switches entirely to the adjusted data. This improves the overall reliability of the results, but at the cost of eliminating real-time data. This is because the adjusted fields for real-time data are set to NA as a matter of policy (see section 2.2.5 of reference 1).

Wider data coverage is obtained if fallback is TRUE. In this case, the focus is shifted to adjusted data *only if* the data-mode for the individual profiles is A or D, indicating either Adjusted or Delayed mode. Any profiles that are of the R (Realtime) data-mode are left unaltered. This blending of adjusted and unadjusted data offers improved spatial and temporal coverage, while reducing the overall data quality, and so this approach should be used with caution. For more on this function see section 3.4 of Kelley et. al (2021).

Value

useAdjusted returns an argoFloats object that is similar to its first argument, but which is set up so that future calls to [[,argoFloats-method or plot,argoFloats-method will focus on the "adjusted" data stream.

Author(s)

Dan Kelley, Jaimie Harbin and Clark Richards

References

1. Argo Data Management Team. "Argo User's Manual V3.4," January 20, 2021. https://archimer.ifremer.fr/doc/
2. Kelley, D. E., Harbin, J., & Richards, C. (2021). argoFloats: An R package for analyzing Argo data. Frontiers in Marine Science, (8), 636922. doi:10.3389/fmars.2021.635922

Examples

```
library(argoFloats)
file <- "SD5903586_001.nc"
raw <- readProfiles(system.file("extdata", file, package = "argoFloats"))
adj <- useAdjusted(raw)
# Autoscale with adjusted values so frame shows both raw and adjusted.
plot(adj, which = "profile", profileControl = list(parameter = "oxygen"), pch = 2)
points(raw[[1]][["oxygen"]], raw[[1]][["pressure"]], pch = 1)
```

```
legend("bottomright", pch = c(2, 1), legend = c("Raw", "Adjusted"))
```

[[,argoFloats-method *Look up a Value Within an argoFloats Object*

Description

This function provides an easy way to look up values within an argoFloats object, without the need to know the exact structure of the data. The action taken by [[depends on the type element in the metadata slot of the object (which is set by the function that created the object), on the value of i and, in some cases, on the value of j; see "Details".

Usage

```
## S4 method for signature 'argoFloats'
x[[i, j, ...]]
```

Arguments

x	an argoFloats object.
i	a character value that specifies the item to be looked up; see "Details".
j	supplemental index value, used for some x types and i values; see "Details".
...	ignored.

Details

There are several possibilities, depending on the object type, as listed below. Note that these possibilities are checked in the order listed here.

1. For all object types:
   (a) If i is "metadata" then the metadata slot of x is returned.
   (b) Otherwise, if i is "data" then the data slot of x is returned.
   (c) Otherwise, if i is "cycle" then a character vector of the cycle numbers is returned.
   (d) Otherwise, if i is "processingLog" then the processingLog slot of x is returned.
   (e) Otherwise, if i is "ID" then a character vector of the ID numbers is returned.
   (f) Otherwise, the following steps are taken, depending on type.
2. If type is "index", i.e. if x was created with getIndex() or with subset,argoFloats-method() acting on the result of getIndex(), then:
   (a) If i is numeric and j is unspecified, then i is taken to be an index that identifies the row(s) of the data frame that was constructed by getIndex() based on the remote index file downloaded from the Argo server. This has elements file for the remote file name, date for the date of the entry, latitude and longitude for the float position, etc.
   (b) If i is the name of an item in the metadata slot, then that item is returned. The choices are: "destdir", "destfileRda", "filename", "ftpRoot", "header", "server", "type", and "url".

(c) Otherwise, if `i` is the name of an item in the `data` slot, then that item is returned. The choices are: `"date"`, `"date_update"`, `"file"`, `"institution"`, `"latitude"`, `"longitude"`, `"ocean"`, and `"profiler_type"`. Note that `"time"` and `"time_update"` may be used as synonyms for `"date"` and `"date_update"`.

(d) Otherwise, if `i=="index"` then that item from the `data` slot of `x` is returned. (For the possible names, see the previous item in this sub-list.)

(e) Otherwise, if `i` is an integer, then the `i`-th row of the `index` item in the `data` slot is returned. This is a good way to learn the longitude and latitude of the profile, the file name on the server, etc.

(f) Otherwise, if `i` is `"ID"` then the return value is developed from the `index$file` item within the `data` slot of `x`, in one of three cases:

i. If `j` is not supplied, the return value is a vector holding the identifiers (character codes for numbers) for all the data files referred to in the index.

ii. Otherwise, if `j` is numeric, then the return value is a subset of the ID codes, as indexed by `j`.

iii. Otherwise, an error is reported.

(g) If `i` is `"length"`, the number of remote files pointed to by the index is returned.

3. Otherwise, if `type` is `"profiles"`, i.e. if `x` was created with `getProfiles()`, then:

(a) If `i` is numeric and `j` is unspecified, then return the local file name(s) that are identified by using `i` as an index.

(b) If `i` is the name of an item in the `metadata` slot, then that item is returned. The choices are: `"type"` and `"destdir"`.

(c) Otherwise, if `i` is the name of an item in the `data` slot, then that item is returned. There is only one choice: `"file"`.

(d) If `i` is `"length"`, the number of local file names that were downloaded by `getProfiles()` is returned.

4. Otherwise, if `type` is `"argos"`, i.e. if `x` was created with `readProfiles()`, then:

(a) If `i` is equal to `"argos"`, and `j` is unspecified, then a list holding the oce::argo objects stored within `x` is returned.

(b) If `i` is equal to `"argos"`, and `j` is provided, then the associated oce::argo object is returned.

(c) If `i` is numeric and `j` is unspecified, then return the argo objects identified by using `i` as an index.

(d) If `i` is the name of an item in the `metadata` slot, then that item is returned. There is only choice, `"type"`.

(e) Otherwise, if `i` is the name of an item in the `data` slot, then that item is returned as a named list. (At present, there is only one choice: `"argos"`.)

(f) Otherwise, if `i` is `"length"` then the number of oce-type argo objects in `x` is returned.

(g) Otherwise, if `i` is a character value then it is taken to be an item within the `metadata` or `data` slots of the argo objects stored in `x`, and the returned value is a list containing that information with one (unnamed) item per profile. If `j` is provided and equal to `"byLevel"`, then the items from the `metadata` are repeated (if necessary) and formed into matrix or vector of the same shape as the `"pressure"` field; this can be convenient for computations involving items that only occur once per profile, such as `"longitude"`, but it should not be used for items that are not level-specific, such as the various `"HISTORY_*"` elements, which apply to a dataset, not to a level

(h) Otherwise, NULL is reported.

5. Otherwise, an error is reported.

Value

the indicated item, or NULL if it is neither stored within the first argument nor computable from its contents.

Author(s)

Dan Kelley

References

Kelley, D. E., Harbin, J., & Richards, C. (2021). argoFloats: An R package for analyzing Argo data. Frontiers in Marine Science, (8), 636922. doi:10.3389/fmars.2021.635922

Examples

```
data(index)
# Full remote filename for first two item in index
paste0(index[["server"]], "/dac/", index[["cycle", 1:2]])
# File names and geographical locations of first 5 items in index
index5 <- subset(index, 1:5)
data.frame(
    file = gsub(".*/", "", index5[["file"]][1]),
    lon = index5[["longitude"]],
    lat = index5[["latitude"]]
)
```

Index

∗ **datasets provided with argoFloats**
index, 20
indexBgc, 21
indexDeep, 22
indexSynthetic, 22
∗ **functions relating to cached values**
argoFloatsClearCache, 7
argoFloatsGetFromCache, 9
argoFloatsIsCached, 9
argoFloatsListCache, 10
argoFloatsStoreInCache, 11
argoFloatsWhenCached, 11
∗ **raw datasets**
D4900785_048.nc, 12
R3901602_163.nc, 34
SD5903586_001.nc, 36
SR2902204_131.nc, 40
[[,argoFloats-method, 49

applyQC, 3
applyQC(), *35*
argoDefaultBathymetry *(*argoDefaultDestdir*)*, 5
argoDefaultBathymetry(), *29*
argoDefaultDestdir, 5
argoDefaultDestdir(), *13, 14*, *17, 18*, *24*, *35*
argoDefaultIndexAge *(*argoDefaultDestdir*)*, 5
argoDefaultIndexAge(), *14*, *24*
argoDefaultProfileAge *(*argoDefaultDestdir*)*, 5
argoDefaultProfileAge(), *13*, *17*
argoDefaultServer *(*argoDefaultDestdir*)*, 5
argoDefaultServer(), *14*, *24*
argoFloats, *3*, *16*, *18, 19*, *25–27*, *29*, *32*, *34, 35*, *37*, *40–42*, *46–49*
argoFloats-class, 6
argoFloatsClearCache, 7
argoFloatsClearCache(), *9–12*
argoFloatsDebug, 7
argoFloatsDebug(), *7*, *9–11*
argoFloatsGetFromCache, 9
argoFloatsGetFromCache(), *7*, *10–12*
argoFloatsIsCached, 9
argoFloatsIsCached(), *7*, *9–12*
argoFloatsListCache, 10
argoFloatsListCache(), *7*, *9–12*
argoFloatsStoreInCache, 11
argoFloatsStoreInCache(), *7*, *9, 10*, *12*
argoFloatsWhenCached, 11
argoFloatsWhenCached(), *7*, *9–11*
as.POSIXct(), *41*

cat(), *8*
curl::curl_download(), *14*

D4900785_048.nc, 12, *34*, *36*, *40*
downloadWithRetries, 12

getIndex, 13
getIndex(), *5, 6*, *13–16*, *18*, *24, 25*, *29*, *40, 41*, *49*
getProfileFromUrl, 16
getProfileFromUrl(), *35*
getProfiles, 18
getProfiles(), *5*, *13*, *15, 16*, *18–23*, *34, 35*, *40*, *50*

hasArgoTestCache *(*argoDefaultDestdir*)*, 5
hexToBits, 19
hexToBits(), *38*

index, 20, *21–23*
indexBgc, *20*, 21, *22, 23*
indexDeep, *20, 21*, 22, *23*
indexSynthetic, *20–22*, 22

lines(), *29*
locator(), *28*

mapApp, 23
mapApp(), *23, 24*
merge,argoFloats-method, 25

oce::argo, *38*, *47*, *50*
oce::colormap(), *29*
oce::download.topo(), *29*
oce::mapLocator(), *28*
oce::mapPlot(), *28*
oce::mapPoints(), *28*
oce::oceColorsGebco(), *29*
oce::plotTS(), *28*, *33*
oce::read.argo(), *34, 35*, *38*
oce::read.topo(), *29*
options(), *5*, *13, 14*, *17, 18*, *24*, *35*

par(), *27, 28*
plot,argoFloats-method, 26
plot.default(), *27*
plotArgoTS, 32
plotArgoTS(), *30*, *33*
points(), *28, 29*

R3901602_163.nc, *12*, 34, *36*, *40*
rawToBits(), *20*
readProfiles, 34
readProfiles(), *3, 4*, *18, 19*, *29, 30*, *32, 33*, *38*, *40*, *48*, *50*
rep(), *33*

SD5903586_001.nc, *12*, *34*, 36, *40*
shiny::runApp(), *25*
show,argoFloats-method, 37
showQCTests, 38
showQCTests(), *19*
SR2902204_131.nc, *12*, *34*, *36*, 40
subset,argoFloats-method, 40
summary,argoFloats-method, 46

useAdjusted, 47
useAdjusted(), *36*