Package ‘RObsDat’

February 27, 2015

Type Package

Title Data Mangement for Hydrology and Beyond Using the Observations Data Model

Version 15.02

Date 2015-02-26

Author Dominik Reusser

Maintainer Dominik Reusser <reusser@pik-potsdam.de>

Depends R (>= 2.11.0), methods, zoo

Imports spacetime, xts, DBI, vwr, e1071, sp

Suggests RSQLite, testthat, RColorBrewer, maptools, SSOAP, XML, geonames

Additional_repositories http://www.omegahat.org/R

Description Data management in hydrology and other fields is facilitated with functions to enter and modify data in a database according to the Observations Data Model (ODM) standard by CUASHI (Consortium of Universities for the Advancement of Hydrologic Science). While this data model has been developed in hydrology, it is also useful for other fields. RObsDat helps in the setup of the database within one of the free database systems MariaDB, PostgreSQL or SQLite. It imports the controlled water vocabulary from the CUASHI web service and provides a smart interface between the analyst and the database: Already existing data entries are detected and duplicates avoided. The data import function converts different data table designs to make import simple. Cleaning and modifications of data are handled with a simple version control system. Variable and location names are treated in a user friendly way, accepting and processing multiple versions. When querying data from the database, it is stored in a spacetime objects within R for subsequent processing.

License GPL

NeedsCompilation no

Repository CRAN

Date/Publication 2015-02-27 16:17:14
**Description**

This package provides classes and functions for handling observational (spatio-temporal) data.

The package provides a number of features for your convenience:  
- Automatic retrieval of standard vocabularies from CUAHSI.  
- Fuzzy matching of terms for more convenient insertion.  
- Automatic handling of synonyms for more convenient data processing.

Note: tell me what additional questions should be answered here.

The package has a large number of dependencies. To install, use the following command:

```r
```

**Details**

Use **addDataValues** to store data in the database.

Use **getMetadata** and **getdataValues** to retrieve information from the database.

**Author(s)**

Dominik Reusser

Maintainer: Dominik Reusser <reusser@pik-potsdam.de>
addCV

References
Reusser et al. 2012: Todo add details

Examples

example(addDataValues)
example(updateValues)

---

addCV **Extend controlled vocabularies**

Description
Add a term to one of the controlled vocabularies. The better way is to have CUAHSI add the term to the official vocabulary through their web-page: http://his.cuahsi.org/mastercvreg/
updateCV queries the official vocabulary again and updates the database tables.
CVtables returns a list of controlled vocabularies.

Usage

addCV(table, term, definition)
updateCV()
CVtables()

Arguments

table Character: Name of the controlled vocabulary. Use CVtables for a list of tables.
term Character: Term to be added.
definition Character: Definition of the term

Value
Nothing returned

Author(s)
Dominik Reusser

See Also
getMetadata allows to query for meta data.
Examples

#connect to standard database
getDefaultDB()
#add data
## Not run: updateCV()
addCV("VariableName", "test", "Test entry to check the possibility to add a term to a vocabulary.")

getMetadata("VariableName", Term="test")

addDataValues  Add, delete or modify data to the observations database

Description

Add, delete or modify data to the observations database. The add function takes either a xts object or values and a date vector as separate objects. Entries for metadata can be provided either corresponding to the columns, the rows or each entry (as matrix) of values. See the CUAHSI observations data model (ODM) for more information about the data model.

Usage

addDataValues(DataZoo = NULL, Date = NULL, Value = NULL, ValueAccuracy = rep(NA, NCOL(DataZoo)), Site, Variable, Offset = rep(NA, NCOL(DataZoo)), OffsetType = rep("No", NCOL(DataZoo)), CensorCode = rep("nc", NCOL(DataZoo)), Qualifier = rep("No", NCOL(DataZoo)), Method = rep("No", NCOL(DataZoo)), Source, Sample = rep("No", NCOL(DataZoo)), DerivedFrom = NULL, QualityControlLevel, tolerance = 0)

deleteDataValues(ID = NULL, reason = NULL)

updateDataValues(getDataResult, reason = NULL)

Arguments

DataZoo  A xts object containing the data. Multiple columns are possible. Either DataZoo or Date and Value needs to be supplied.
Date    A object of class POSIXct containing the time information. Either DataZoo or Date and Value needs to be supplied.
Value    A matrix of containing the values. Either DataZoo or Date and Value needs to be supplied.
ValueAccuracy  Information about the accuracy of the data.
Site    Information about the Site at which the data was observed.
Variable  Information about what variable was observed.
### addDataValues

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Offset</td>
<td>Information about the offset of the observation. See also OffsetType.</td>
</tr>
<tr>
<td>OffsetType</td>
<td>Information about the type of the offset as defined in the OffsetTypes table. See also addoffsettype.</td>
</tr>
<tr>
<td>CensorCode</td>
<td>Information about the censor used for the observation.</td>
</tr>
<tr>
<td>Qualifier</td>
<td>Qualifying information that can note anything unusual or problematic about individual observations such as, for example, 'holding time for analysis exceeded' or 'incomplete or inexact daily total.'</td>
</tr>
<tr>
<td>Method</td>
<td>The method of field data collection, which may specify 'how' a physical observation was made or collected</td>
</tr>
<tr>
<td>Source</td>
<td>Reference to the original sources of the data, providing information sufficient to retrieve and reconstruct the data value from the original data files</td>
</tr>
<tr>
<td>Sample</td>
<td>Information about physical samples analysed in a laboratory to obtain an observation.</td>
</tr>
<tr>
<td>DerivedFrom</td>
<td>Reference to another record in the database, from which a value was derived.</td>
</tr>
<tr>
<td>QualityControlLevel</td>
<td>Level of quality controlled applied to a dataset.</td>
</tr>
<tr>
<td>tolerance</td>
<td>Upon import, it is checked whether a dataset already exists. Tolerance gives the allowed difference between existing and new record, in order to judge them to be the same.</td>
</tr>
<tr>
<td>getDataResult</td>
<td>A object obtained from getDataResults for updating. The returned results can be modified before resubmitting to the database. By using version management, deleted and previous versions of records are still available with getDataVersions.</td>
</tr>
<tr>
<td>ID</td>
<td>Vector of the IDs of the records to be deleted or an object obtained from getDataResults that contains exactly the records to be deleted.</td>
</tr>
<tr>
<td>reason</td>
<td>The reason why the records are deleted in plain text.</td>
</tr>
</tbody>
</table>

### Details

Raw data often needs to be cleaned before it can be used. RObsDat supports this and allows to reconstruct data modification operations by use of version management. This is valid for deleted and previous versions. To be written. Tell me (see maintainer) if something is missing.

### Value

nothing returned

### Author(s)

Dominik Reusser

### See Also

To retrieve data from the database, see getDataValues and getDataValues. getDataVersions allows to access previous versions of a record, if Version management is implemented by the database system.
Examples

```r
# connect to database
getDefautlDB()

## Not run:
# connect to PostgreSQL database
require("RObsDat")
require("RPostgreSQL")
m <- dbDriver("PostgreSQL")
con <- dbConnect(m, user="a_user", password="secret", dbname="obsdat")
sqhandler <- new("odml_1Ver", con=con)
options(odm.handler=sqhandler)

# connect to MySQL database
require("RObsDat")
require("RMysql")
m <- dbDriver("MySQL")
con <- dbConnect(m, user="a_user", password="secret", dbname="obsdat")
sqhandler <- new("odml_1Ver", con=con)
options(odm.handler=sqhandler)

# connect to SQLite database
require("RObsDat")
require("RSQLite")
m <- dbDriver("SQLite")
database.db = "database.db"
con <- dbConnect(m, dbname = database.db)
sqhandler <- new("odml_1Ver", con=con)
options(odm.handler=sqhandler)

## End(Not run)

# Store metadata in database
addSite(Code="test", Name="Virtual test site", x=-5, y=46,
LatLongDatum="WGS84", Elevation=1500, State="Germany")
addVariable(Name="Distance", Unit="cm", ValueType="Field Observation",
GeneralCategory="Instrumentation", Code="test_dist")
addQualityControlLevel(ID=6, Code="test_ok", Definition="The default")

addISOMetadata(TopicCategory="Unknown", Title="Testdata",
Abstract="This data is created to test the functions of RObDsDat")
addSource(Organization="Your Org", SourceDescription="Madeup data",
SourceLink="RObsDat Documentation", ContactName="Yourself",
Metadata="Testdata")

library(xts)
library(spacetime)

example.data <- xts(1:40, seq(as.POSIXct("2014-01-01", tz="UTC"),
as.POSIXct("2014-02-09", tz="UTC"), length.out=40))
example.data[40] <- 30
```
addISOMetadata

example.data[35] <- 22
addDataValues(example.data[1:20], Site="Virtual test site", Variable="test_dist", Source="Madeup", QualityControlLevel="test_ok")
# Avoid duplicates automatically
example.data[75] <- 30
addDataValues(example.data, Site="Virtual test site", Variable="test_dist", Source="Madeup", QualityControlLevel="test_ok")
indb <- getDataValues(Site="test")
stplot(indb, mode="ts")

# Version management
indb <- getDataValues(Site="test")
to.correct <- which(indb@data > 30)
indb@data[to.correct,] <- 20
if(NROW(indb@data)>=30){
  indb@data[30,] <- 32
  updateDataValues(indb, "Correction of wrong value")
}
ver2 <- indb
if(NROW(ver2@data)>=13){
  ver2@data[10:13,] <- 60
  updateDataValues(ver2, "Changing more data")
}
indb <- getDataValues(Site="test")
ver3 <- indb
if(NROW(ver3@data)>=32){
  ver3@data[30:32,] <- 33
  updateDataValues(ver3, "Ups, I used 60 instead of 33 by mistake")
}

# If only one time point or one location is selected you have to use '@ValueId's in addition.
# If only one time point or one location is selected you have to use '@ValueId's in addition.
if(dim(indb@ValueId)==29) deleteDataValues(indb@ValueId[2,29], "Remove a value")
if(dim(indb@ValueId)==14) deleteDataValues(indb@ValueId[10:14], "Remove several values")

# When more than one spatial point and times are given,
# then use the usually selection without 'ValueId's.
# deleteDataValues(testSiteData[2,20:24], "Remove several values.")
getDataVersions()

versionQuery1 <- getDataValues(Site=1, VersionID=1)
stplot(versionQuery1, mode="ts")

versionQuery2 <- getDataValues(Site=1, VersionID=2)
stplot(versionQuery2, mode="ts")

addISOMetadata   Functions to provide reference to data sources
addISOMetadata

Description

The ISOMetadata table contains dataset and project level meta data required by the CUAHSI HIS meta data system (http://www.cuahsi.org/his/documentation.html) for compliance with standards such as the draft ISO 19115 or ISO 8601.

Usage

```sql
addISOMetadata(TopicCategory = "Unknown", Title = "Unknown",
Abstract = "Unknown", ProfileVersion = "Unknown",
MetadataLink = NULL)
addSource(Organization, SourceDescription, SourceLink = NULL,
ContactName = rep("Unknown", length(Organization)),
    Phone = rep("Unknown", length(Organization)), Email =
    rep("Unknown", length(Organization)), Address =
    rep("Unknown", length(Organization)), City =
    rep("Unknown", length(Organization)), State =
    rep("Unknown", length(Organization)), ZipCode =
    rep("Unknown", length(Organization)), Citation =
    rep("Unknown", length(Organization)), Metadata =
    rep("Unknown", length(Organization)))
```

Arguments

- **TopicCategory**: Reference to the TopicCategory table, giving the broad ISO19115 metadata topic category for data from this source.
- **Title**: Title of the data source
- **Abstract**: A short abstract characterizing the data source
- **ProfileVersion**: Name of the metadata profile used by the data source
- **MetadataLink**: Link to additional metadata reference material.
- **Organization**: Name of the organization that collected the data.
- **SourceDescription**: Full text description of the source of the data.
- **SourceLink**: Link that can be pointed at the original data file and/or associated metadata stored in the digital library or URL of data source.
- **ContactName**: Name of the contact person for the data source.
- **Phone**: Phone number for the contact person.
- **Email**: Email address for the contact person.
- **Address**: Street address for the contact person.
- **City**: City in which the contact person is located.
- **State**: State in which the contact person is located. Use two letter abbreviations for US. For other countries give the full country name.
- **ZipCode**: US Zip Code or country postal code
- **Citation**: Text string that give the citation to be used when the data from each source are referenced.
- **Metadata**: Reference to the metadata table.
addOffsetType

Value
Nothing is returned

Author(s)
Dominik Reusser

See Also
Use getMetadata to retrieve values.

Examples
#connect to standard database
getDefautlDB()
#add data
addISOMetadata(TopicCategory="inlandWaters", Title="FAOStat land use",
MetadataLink="faostat.fao.org/site/377/default.aspx")
addSource(Organization="FAO", SourceDescription="FAO Stat Land Use",
SourceLink="http://faostat.fao.org/site/377/default.aspx#anchor",
ContactName="AskFAOSTAT: http://www.fao.org/askfao/agriculturalstatisticaldata/en/",
Metadata="FAOStat land use")
getMetadata(table="Source", Description="Land Use")

Description
OffsetTypes table lists full descriptive information for each of the measurement offsets.

Usage
addOffsetType(Units, Description)

Arguments
Units Units of the OffsetValue.
Description Full text description of the offset type.

Value
nothing returned

Author(s)
Dominik Reusser
addQualityControlLevel

Add Quality metadata vocabulary

Description

Define the level of quality control processing that the data value has been subjected to

Usage

addQualityControlLevel(ID, Code, Definition, Explanation = "")

Arguments

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>numerical ID</td>
</tr>
<tr>
<td>Code</td>
<td>Short code for quality control level</td>
</tr>
<tr>
<td>Definition</td>
<td>Exact definition of the level</td>
</tr>
<tr>
<td>Explanation</td>
<td>Further explanations</td>
</tr>
</tbody>
</table>

Details

From CUAHSI documentation: UAHSI ODM is consistent with the practice of other data systems:

- QualityControlLevelCode = "0" - Raw Data Raw data is defined as unprocessed data and data products that have not undergone quality control. Depending on the data type and data transmission system, raw data may be available within seconds or minutes after real-time. Examples include real-time precipitation, stream flow and water quality measurements.

- QualityControlLevelCode = "1" - Quality Controlled Data Quality controlled data have passed quality assurance procedures such as routine estimation of timing and sensor calibration or visual inspection and removal of obvious errors. An example is USGS published stream flow records following parsing through USGS quality control procedures.

- QualityControlLevelCode = "2" - Derived Products

Derived products require scientific and technical interpretation and include multiple-sensor data. An example might be basin average precipitation derived from rain gages using an interpolation procedure.
- QualityControlLevelCode = "3" - Interpreted Products These products require researcher (PI) driven analysis and interpretation, model-based interpretation using other data and/or strong prior assumptions. An example is basin average precipitation derived from the combination of rain gages and radar return data.

- QualityControlLevelCode = "4" - Knowledge Products These products require researcher (PI) driven scientific interpretation and multidisciplinary data integration and include model-based interpretation using other data and/or strong prior assumptions. An example is percentages of old or new water in a hydro-graph inferred from an isotope analysis.

These definitions for quality control level are stored in the QualityControlLevels table. These definitions are recommended for use, but users can define their own quality control level system. The QualityControlLevels table is not a controlled vocabulary, but specification of a quality control level for each data value is required.

**Value**

Nothing returned

**Author(s)**

Dominik Reusser / CUAHSI

**References**

CUAHSI Community Observations Data Model (ODM), Version 1.1, Design Specifications

**See Also**

addDataValues

**Examples**

getDefaultDB()

addQualityControlLevel(ID=1,Code="1", Definition="Quality Controlled Data", Explanation="Quality controlled data have passed quality assurance procedures such as routine estimation of timing and sensor calibration or visual inspection and removal of obvious errors. An example is USGS published stream flow records following parsing through USGS quality control procedures.")

**Description**

Information about the spatial location at which data values have been collected.
Usage

```r
addSite(Code, Name, x, y, Elevation = rep(0, length(Code)),
        LatLongDatum, LocalProjection = NULL, isLocal = NULL,
        VerticalDatum = NULL, PositionAccuracy = rep(0,
        length(Code)), State = NULL, County = NULL, Comment = NULL)
```

Arguments

- **Code**: Code used by organization that collects the data to identify the site
- **Name**: Full name of the sampling site.
- **x**: x coordinate
- **y**: y coordinate
- **Elevation**: Elevation of sampling location (in m).
- **LatLongDatum**: Spatial Reference System of the latitude and longitude coordinates
- **LocalProjection**: Spatial Reference System of the local coordinates
- **isLocal**: Boolean, indicating if x and y are in the lat/long or local coordinate system.
- **VerticalDatum**: Vertical datum of the elevation.
- **PositionAccuracy**: Value giving the accuracy with which the positional information is specified in meters.
- **State**: Name of state in which the monitoring site is located.
- **County**: Name of county in which the monitoring site is located.
- **Comment**: Additional comments for the location.

Value

Nothing is returned.

Author(s)

Dominik Reusser

See Also

Use `getMetadata` to retrieve values.

Examples

```r
#connect to standard database
getDefaultDB()

#add data
addSite(Code="10109000", Name="LOGAN RIVER ABOVE STATE DAM, NEAR LOGAN,UT",
        x=-100.47, y=45.32, LatLongDatum="WGS84", Elevation="1432",
        VerticalDatum="NAVD88", PositionAccuracy=100, Stat="Utah")

getMetadata(table="Site", Name="LOGAN")
```
addSpatialReferences  Add spatial reference system

Description

Add a term to the spatial references controlled vocabularies. The better way is to have CUAHSI add the term to the official vocabulary through their web page: http://his.cuahsi.org/mastercvreg/

Usage

addSpatialReferences(ID, SRSID, Name, IsGeographic, Notes)

Arguments

ID  Unique ID.
Name  Name of the Spatial Reference System.
IsGeographic  Boolean, indicating whether the spatial reference system uses geographic coordinates.
Notes  Descriptive information about the Spatial Reference System.

Value

Nothing returned

Author(s)

Dominik Reusser

See Also

Use getMetadata to retrieve values.

Examples

#connect to standard database
getDefaultDB()
#add data
addSpatialReferences(ID=4269, SRSID=4269, Name="NAD83",
IsGeographic=TRUE, Notes="todo: include notes in this example")
addSynonym

Add a synonym for one of the entries in any meta data table

Description

For more convenience while importing data, synonyms for existing entries can be handled by the package. This command allows you to add a synonym. Existing synonyms are then used to match the right record when searching and importing data.

Usage

addSynonym(table, phrase, id)

Arguments

table: Name of the table, in which an entry exists.
phrase: The synonym.
id: Unique ID of the record.

Details

Do you need more? Let me know (see package maintainer).

Value

nothing returned

Author(s)

Dominik Reusser

See Also

gID, exportSynonyms

Examples

#connect to standard database
gDefaultDB()
#add data
addSite(Code="10109000", Name="LOGAN RIVER ABOVE STATE DAM, NEAR LOGAN,UT", 
x=-100.47, y= 45.32, LatLongDatum="WGS84", Elevation="1432", 
VerticalDatum="NAVD88", PositionAccuracy=100, Stat="Utah")
theID <- gID(table="Site", "Logan")
addSynonym(table="Site", phrase="logan dam", id=theID)
addUnits

Add a term to the controlled vocabulary for units. The better way is to have CUAHSI add the term to the official vocabulary through their webpage: http://his.cuahsi.org/mastercvreg/

Usage

addUnits(Name, Type, Abbreviation)

Arguments

Name
Name of the unit. e.g. square meter

Type
Type of the unit. e.g. area

Abbreviation
SI abbreviation e.g. m^2

Value

Nothing returned

Author(s)

Dominik Reusser

See Also

updateCV to retrieve the vocabularies from the CUAHSI web service and update the database.
getMetadata allows to query for meta data.

Examples

getDefaultDB()
#add data
addUnits(Name="decisiemens per meter", Type="Electrical Conductivity", Abbreviation="dS/m")
getMetadata("Units", Name="per meter")
addVariable

Add an entry to the variables table

Description

The Variables table lists the full descriptive information about what variables have been measured. This function allows to add an entry.

Usage

addVariable(Code, Name, Speciation = rep("Unknown", NROW(Code)), Unit,
SampleMedium = rep("Unknown", NROW(Code)),
ValueType = rep("Unknown", NROW(Code)),
IsRegular = rep("True", NROW(Code)),
TimeSupport = rep(0, NROW(Code)),
TimeUnits = rep("Julian year", NROW(Code)),
DataType = rep("Unknown", NROW(Code)),
GeneralCategory = rep("Unknown", NROW(Code)),
NoDataValue = rep(-999999, NROW(Code)))

Arguments

<table>
<thead>
<tr>
<th>Code</th>
<th>Text code used by the organization that collects the data to identify the variable.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Full text name of the variable that was measured, observed, modelled. Refers to the controlled vocabulary.</td>
</tr>
<tr>
<td>Speciation</td>
<td>Text code used to identify how the data value is expressed (i.e., total phosphorus concentration expressed as P). This should be from the SpeciationCV controlled vocabulary table.</td>
</tr>
<tr>
<td>Unit</td>
<td>the units of the data values associated with the variable. Refers to the Units controlled vocabulary.</td>
</tr>
<tr>
<td>SampleMedium</td>
<td>The medium in which the sample or observation was taken or made. This should be from the SampleMediumCV controlled vocabulary table.</td>
</tr>
<tr>
<td>ValueType</td>
<td>Text value indicating what type of data value is being recorded. This should be from the ValueTypeCV controlled vocabulary table.</td>
</tr>
<tr>
<td>IsRegular</td>
<td>Value that indicates whether the data values are from a regularly sampled time series.</td>
</tr>
<tr>
<td>TimeSupport</td>
<td>Numerical value that indicates the time support (or temporal footprint) of the data values. 0 is used to indicate data values that are instantaneous. Other values indicate the time over which the data values are implicitly or explicitly averaged or aggregated.</td>
</tr>
<tr>
<td>TimeUnits</td>
<td>Integer identifier that references the record in the Units table giving the Units of the time support. If TimeSupport is 0, indicating an instantaneous observation, a unit needs to still be given for completeness, although it is somewhat arbitrary.</td>
</tr>
<tr>
<td>DataType</td>
<td>Text value that identifies the data values as one of several types from the DataTypeCV controlled vocabulary table.</td>
</tr>
</tbody>
</table>
getDataValues

General Category
General category of the data values from the GeneralCategoryCV controlled vocabulary table.

NoDataValue
Numeric value used to encode no data values for this variable.

Details
What is missing?

Value
Nothing is returned

Author(s)
Dominik Reusser

See Also
updateCV to retrieve the vocabularies from the CUAHSI web service and update the database.
getMetadata allows to query for meta data.

Examples

#connect to standard database
getDefaultDB()
#add data
addVariable(Code="00060", Name="Discharge", Speciation = "Unknown",
Unit="m^3/s", SampleMedium = "Not Relevant",
ValueType = "Field Observation", TimeSupport = 0,
DataType = "Unknown", GeneralCategory = "Hydrology")
getMetadata("Variable", Name="Discharge")

---

**getDataValues**

Retrieves data from the observations database

Description
getDataValues allows to query data from the database.
getDataVersions displays all the versions available in the database.
Usage

getDataValues(ID = NULL, from = NULL, to = NULL,
tz=c("global", "UTC", "GMT", "0", "local"), Site = NULL,
Variable = NULL, Offset = NULL, OffsetType = NULL,
CensorCode = NULL, Qualifier = NULL, Method = NULL,
Source = NULL, Sample = NULL, DerivedFromID = NULL,
QualityControlLevel = NULL, VersionID = NULL,
VersionDate = NULL, show.deleted = FALSE,
all.ID = FALSE)

dataVersions()

Arguments

ID
from
to
tz
Site
Variable
Offset
OffsetType
CensorCode
Qualifier
Method
Source
Sample
DerivedFromID
QualityControlLevel
VersionID
VersionDate
show.deleted
all.ID

Unique ID of the DataValue
Starting date of the period for which to retrieve the information
End date of the period for which to retrieve the information
String indicating whether the data should be obtained in the local or global time zone.
Information about the Site at which the data was observed.
Information about what variable was observed.
Information about the offset of the observation. See also OffsetType.
Information about the type of the offset as defined in the OffsetTypes table. See also addOffsetType.
Information about the censor used for the observation.
Qualifying information that can note anything unusual or problematic about individual observations such as, for example, 'holding time for analysis exceeded' or 'incomplete or inexact daily total.'
The method of field data collection, which may specify 'how' a physical observation was made or collected.
Reference to the original sources of the data, providing information sufficient to retrieve and reconstruct the data value from the original data files
Information about physical samples analysed in a laboratory to obtain an observation.
Reference to another record in the database, from which a value was derived.
Level of quality controlled applied to a dataset.
Data version to be retrieved (see details about version management)
Creation date of the version to be retrieved (see details about version management)
Include records that have been deleted (see details about version management)
Boolean: Can the function assume that all values are passed as ID. Reduces number of Database queries. Do not use normally, mainly for internal use.
**getMetadata**

**Details**

Raw data often needs to be cleaned before it can be used. RObDat supports this and allows to reconstruct data modification operations by use of version management. This is valid for deleted and previous versions. What do you want to know in addition?

**Value**

The function `getDataValues` returns a spacetime object. Through using the R package spacetime it combine spatial data (sp), time series (xts) and the measurements. Furthermore they are attached ValueIDs, DerivedFromIDs and Metadata of the spacetime object.

**Author(s)**

Dominik Reusser

**See Also**

For information how to insert data into the database `addDataValues`

**Examples**

```r
getDefaultDB()

#Put data into the database
example(addDataValues)

# Get a list of available variables with ID's)
getMetadata("Variable")$Name

# Get ID of variable of interest
var.id <- getID("Variable", "Distance")

# Get data
data <- getDataValues(Variable="Distance", Site="test site")
# returns data object of class 'spacetime' with 3 slots

# Get data values:
data@data #returns values of spacetime-object
```

---

**getMetadata**  
*Get information about meta data*
Description

getMetadata allows to query entries from the meta data tables.

getID allows to find the ID for an entry from a meta table. A smart matching algorithm is used to retrieve relevant entries.

id2name is an internal function. It takes a data frame (e.g. returned from IgetDataValues) and converts each column with a name starting with ID to the name or term from the corresponding meta data table.

Usage

getMetadata(table, EXACT = FALSE, ...)
getID(table, value, remove.special.character=TRUE)
id2name(dataframe)

Arguments

table Table name of the meta data table.
EXACT boolean indicating whether only exact matches should be returned or partial matches of the term should be accepted.
remove.special.character boolean indicating whether special characters should be removed when comparing. Usually, this should be turned on.
value String describing the entry in the table.
dataframe Data frame to be converted from ID - columns to names.
... Additional columns to filter entries.

Details

The matching algorithm of getID first attempts to find exact matches in one of the column. If no result is obtained, partial matches and similar entries based on the Levenstein algorithm are searched.

ToDo explain use of synonyms

Value

getID returns a vector of the IDs. getMetadata returns a data frame representing the meta data table.

Author(s)

Dominik Reusser

See Also

Use CVtables to obtain a list of controlled vocabularies. See also the ODM documentation from CUAHSI
Examples

# connect to standard database
# this also retrieves the controlled vocabularies from the CUHISI server
getDefaultDB()

# retrieve meta data
getMetadata(table="VariableName")

inherited_stfdf-class  Internal: Class inherited stfdf

Description

This class is an inherited STFDF from the packages spacetime. STFDF is for spatio-temporal data with a full data frame. It shows n spatial points and m times. For each location and time exists observations. It deals with spatio-temporal data and provides special time series analysis. The object of class inherited_stfdf contains one stfdf with the main-data: spatial data of package sp and temporal informations of class xts. The actual data is stored in the form of a data.frame. Furthermore it contains two stfdf-objects with the ‘ValueIDs’ and the ‘DerivedFromIDs’ and also one data.frame with the Meta-informations.

Usage

inherited_stfdf(sp, time, data, endtime,
ValueIDs, DerivedFromIDs, MetadataRel, Metadata)

## S4 method for signature ‘inherited_stfdf’
x[i, j, ..., drop = TRUE]

Arguments

sp          object of class Spatial, with n elements
time        object holding time information, of length m
time         vector of class POSIXct, end points of time intervals; if not specified the time intervals are choose by default, when intervals are regular delta
data         data frame with n*m observations
ValueIDs     an object of class STFDF; see STFDF, which contains the ID’s of the measured values
DerivedFromIDs an object of class STFDF; see STFDF, which contains the ID’s of the measured values
MetadataRel  an object of class STFDF; see STFDF, which contains the metadata-ids for build a relationship with the dataValues
Metadata     an object of class data.frame; see STFDF, which contains metadata of measurements, e.g. variablename, methods, etc.
x            an object of class inherited_stfdf
i variety of spatial entities
j range of temporal entities
... selection of attribute(s)
drop has no effect

**Objects from the Class**

Objects of this class represent full space/time data with a full grid (or lattice) layout

**Slots**

- **sp**: spatial object; see ST-class
- **time**: an object of class xts; see xts
- **data**: data.frame, which holds the measured values; space index cycling first, time order preserved
- **endtime**: it is a vector of class POSIXct. It contains end points of time intervals
- **ValueIDs**: an object of class STFDF; see STFDF, which contains the ID’s of the measured values
- **DerivedFromIDs**: an object of class STFDF; see STFDF, which contains the ID’s of the measured values
- **MetadataRel**: an object of class STFDF; see STFDF, which contains the metadata-ids for build a relationship with the dataValues
- **Metadata**: an object of class data.frame; see STFDF, which contains metadata of measurements, e.g. variablename, methods, etc.

**Methods**

- [ signature(x = "inherited_stfdf")]: selects location, times and variables
- == signature(e1="inherited_stfdf",e2="inherited_stfdf"): compares two inherited stfdf’s

**Author(s)**

Dominik Reusser

**Examples**

# look at documentation of addDataValues. Import more than one location to execute the example.
# when you use getDataValues the method return an inherited_stfdf
element(addDataValues)
inDB = getDataValues()

# Select data
#You can selected the data by using numerics and characters.
#In the following code line ‘inDB’ is sub-setted to the
#second and third location, the first twenty dates and the temperature.
#Advice: The variable must have the correct term. For support
#look at the column name of inDB@data.
selectedData = indB[1, 1:10, "Distance"]

# Furthermore you can miss out some parameters:
selectedData2 = indB[1,10:10]
selectedData3 = indB[,10:10]

# Attention: if the dataset contains only one location
# (or it is subset only one point) it's impossible
# to get a inherited spacetime-object. So you get a object of xts.
# If only one time is selected you get a SpatialPointsDataFrame.

# The inherited stfdf-data can visualized in different plots. For example:

stplot(indB, mode='tp', type = 's')
stplot(indB, mode='xt')

# Further you can access the slots of object by take advantage of the '@'.
inDB@Metadata
inDB@sp
inDB@time
inDB@data

---

odm.close

Open standard SQLite database delivered with the package and close existing connection to observations database.

---

**Description**

Close the existing connection to observations database.

**Usage**

odm.close()

getDefaultDB(file="RODM.db")

**Arguments**

- **file** Name of the file delivered with the package. Normally should not be changed.

**Value**

Nothing returned.

**Author(s)**

Dominik Reusser
Examples

getDefaultDB()

odm.close()

---

**odm1_1-class**

*Classes "odm1_1" and "odm1_1Ver"*

---

**Description**

Observations data works with a hidden layer, in which the commands are translated into SQL-queries and submitted to the database server. The two classes implement this hidden layer. "odm1_1Ver" also provides mechanisms to a simple version management system as described in Reusser et. al 2012.

**Objects from the Class**

Objects can be created by calls of the form `new("odm1_1", con=connection)` and `new("odm1_1Ver", con=connection)`. The objects store the database connection object returned by `dbConnect` in the only slot.

**Slots**

- `con`: Object of class "DBIConnection" returned by `dbConnect`

**Extends**

`odm1_1Ver` extends class `odm1_1` directly.

**Methods**

- `IaddCV` signature(object = "odm1_1")
- `IaddDataValues` signature(object = "odm1_1")
- `IaddDataVersion` signature(object = "odm1_1")
- `IaddISOMetadata` signature(object = "odm1_1")
- `IaddSite` signature(object = "odm1_1")
- `IaddSource` signature(object = "odm1_1")
- `IaddSpatialReferences` signature(object = "odm1_1")
- `IaddSynonym` signature(object = "odm1_1")
- `IaddUnits` signature(object = "odm1_1")
- `IaddVariable` signature(object = "odm1_1")
- `IarchiveDataValues` signature(object = "odm1_1")
- `IdbState` signature(object = "odm1_1")
- `IdeleteDataValues` signature(object = "odm1_1")
IgetCensorCode signature(object = "odm1_1"): ...
IgetCurrentDataVersion signature(object = "odm1_1"): ...
IgetCV signature(object = "odm1_1"): ...
IgetDataType signature(object = "odm1_1"): ...
IgetDataValues signature(object = "odm1_1"): ...
IgetDataVersions signature(object = "odm1_1"): ...
IgetGeneralCategory signature(object = "odm1_1"): ...
IgetISOMetadata signature(object = "odm1_1"): ...
IgetMethods signature(object = "odm1_1"): ...
IgetNo signature(object = "odm1_1"): ...
IgetOffsetTypes signature(object = "odm1_1"): ...
IgetOldDataValues signature(object = "odm1_1"): ...
IgetQualifiers signature(object = "odm1_1"): ...
IgetQualityControlLevels signature(object = "odm1_1"): ...
IgetSampleMedium signature(object = "odm1_1"): ...
IgetSamples signature(object = "odm1_1"): ...
IgetSampleType signature(object = "odm1_1"): ...
IgetSite signature(object = "odm1_1"): ...
IgetSource signature(object = "odm1_1"): ...
IgetSpatialReferences signature(object = "odm1_1"): ...
IgetSpeciation signature(object = "odm1_1"): ...
IgetSynonymID signature(object = "odm1_1"): ...
IgetTopicCategory signature(object = "odm1_1"): ...
IgetUnits signature(object = "odm1_1"): ...
IgetValueType signature(object = "odm1_1"): ...
IgetVariable signature(object = "odm1_1"): ...
IgetVariableName signature(object = "odm1_1"): ...
IgetVerticalDatum signature(object = "odm1_1"): ...
IupdateDataValues signature(object = "odm1_1"): ...
IaddDataVersion signature(object = "odm1_1Ver"): ...
IarchiveDataValues signature(object = "odm1_1Ver"): ...
IdbState signature(object = "odm1_1Ver"): ...
IgetCurrentDataVersion signature(object = "odm1_1Ver"): ...
IgetDataVersions signature(object = "odm1_1Ver"): ...
IgetOldDataValues signature(object = "odm1_1Ver"): ...

Author(s)
Dominik Reusser
Synonym transfer functions

References

Reusser et al. 2012 - todo add exact reference

Examples

```r
showClass("odm1_l")
showClass("odm1_lVer")

require(RSQLite)
m <- dBDriver("SQLite")
con <- dbConnect(m, dbname = "RODM.db")

#dbGetQuery(con, "SELECT * FROM Versions")
sqhandler <- new("odm1_lVer", con=con)
## Not run:
   #without version management
   sqhandler <- new("odm1_l", con=con)

## End(Not run)
options(odm.handler=sqhandler)

getMetadata("VariableName")
```

---

Synonym transfer functions

Transfer synonyms

Description

Functions that allow to transfer synonyms from one database to another. This may be useful because the synonyms include quite a bit of manual matching on large datasets. Transferring those allows to reconstruct a database from source file more easily.

Usage

```r
exportSynonyms(file)
importSynonyms(file)
```

Arguments

- **file**  
  File name for export and import

Value

Used for the side effect of creating/importing a file. The file includes the columns phrase, table and key.
Synonym transfer functions

Author(s)
Dominik Reusser

See Also
getID, addSynonym

Examples
exportSynonyms("synonym_table.tab")
importSynonyms("synonym_table.tab")
Index

*Topic  **textasciitildeodm**
  addQualityControlLevel, 10
*Topic   **classes**
  odm1_1-class, 24
*Topic  **package**
  RObsdat-package, 2
*Topic  **stfdf**
  inherited_stfdf-class, 21
*Topic  **utilities**
  addCV, 3
  addDataValues, 4
  addISOMetadata, 7
  addUnits, 11
  addSite, 17
  addSpatialReferences, 19
  synonym transfer functions, 26
  addSynonym, 14
  addVariable, 16
  getDatavalues, 2, 4, 11, 17
  getMetadata, 19
  getid, 14, 27
  getdefaultdb (odm.close), 23
  getid (getMetadata), 19
  getversion, 2, 3, 9, 10, 12, 13, 15, 17, 19
  iaddcv, odm1_1-method (odm1_1-class), 24
  iadddatavalues, odm1_1-method (odm1_1-class), 24
  iaddedataversion, odm1_1-method (odm1_1-class), 24
  iaddedataversion, odm1_1-ver-method (odm1_1-class), 24
  iaddisometadata, odm1_1-method (odm1_1-class), 24
  iaddsource, odm1_1-method (odm1_1-class), 24
  iaddsite, odm1_1-method (odm1_1-class), 24
  iadddatavalues, odm1_1-method (odm1_1-class), 24
  iaddunits, odm1_1-method (odm1_1-class), 24
  cvtables, 20
  cvtables (addcv), 3
  dbconnect, 24
  deletedatavalues (adddatavalues), 4
  delta, 21
  exportSynonyms, 14
  exportSynonyms (Syonym transfer functions), 26
  getdatavalues, 2, 5, 17
  getdataversions, 5
  getdataversions (getdatavalues), 17
  getdefaultdb (odm.close), 23
  getid, 14, 27
  getid (getmetadata), 19
  getversion, 2, 3, 9, 10, 12, 13, 15, 17, 19
  iaddcv, odm1_1-method (odm1_1-class), 24
  iadddatavalues, odm1_1-method (odm1_1-class), 24
  iaddedataversion, odm1_1-method (odm1_1-class), 24
  iaddedataversion, odm1_1-ver-method (odm1_1-class), 24
  iaddisometadata, odm1_1-method (odm1_1-class), 24
  iaddsource, odm1_1-method (odm1_1-class), 24
  iaddsite, odm1_1-method (odm1_1-class), 24
  iadddatavalues, odm1_1-method (odm1_1-class), 24
  iaddunits, odm1_1-method (odm1_1-class), 24
  addcv, 3
  adddatavalues, 2, 4, 11, 19
  addisometadata, 7
  addoffsettype, 5, 9, 18
  addqualitycontrollevel, 10
  addsite, 11
  addsource (addisometadata), 8
  addspatialreferences, 13
  addsynonym, 14, 27
  addunits, 15
  addvariable, 16
  cVtables, 20
  CVtables (addcv), 3
  dbconnect, 24
  deleteDataValues (addDataValues), 4
  delta, 21
  exportSynonyms, 14
  exportSynonyms (Synonym transfer functions), 26
  getDataValues, 2, 5, 17
  getDataVersions, 5
  getDataVersions (getDataValues), 17
  getDefaultDB (odm.close), 23
  getID, 14, 27
  getID (getMetadata), 19
  getMetadata, 2, 3, 9, 10, 12, 13, 15, 17, 19
  iaddcv, odm1_1-method (odm1_1-class), 24
  iadddatavalues, odm1_1-method (odm1_1-class), 24
  iaddedataversion, odm1_1-method (odm1_1-class), 24
  iaddedataversion, odm1_1-ver-method (odm1_1-class), 24
  iaddisometadata, odm1_1-method (odm1_1-class), 24
  iaddsource, odm1_1-method (odm1_1-class), 24
  iaddsite, odm1_1-method (odm1_1-class), 24
  iadddatavalues, odm1_1-method (odm1_1-class), 24
  iaddunits, odm1_1-method (odm1_1-class), 24
28
<table>
<thead>
<tr>
<th>Function</th>
<th>Class</th>
<th>Line</th>
</tr>
</thead>
<tbody>
<tr>
<td>IaddVariable, odml_1-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>IarchiveDataValues, odml_1-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>IarchiveDataValues, odml_1Ver-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>id2name (getMetadata)</td>
<td></td>
<td>19</td>
</tr>
<tr>
<td>IdbState, odml_1-method (odml_1-class)</td>
<td></td>
<td>24</td>
</tr>
<tr>
<td>IdbState, odml_1Ver-method (odml_1-class)</td>
<td></td>
<td>24</td>
</tr>
<tr>
<td>IdeleteDataValues, odml_1-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>IgetCensorCode, odml_1-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>IgetCurrentDataVersion, odml_1-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>IgetCurrentDataVersion, odml_1Ver-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>IgetCV, odml_1-method (odml_1-class)</td>
<td></td>
<td>24</td>
</tr>
<tr>
<td>IgetDataType, odml_1-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>IgetDataValues, odml_1-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>IgetDataVersions, odml_1-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>IgetDataVersions, odml_1Ver-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>IgetGeneralCategory, odml_1-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>IgetISOMetadata, odml_1-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>IgetMethods, odml_1-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>IgetNo, odml_1-method (odml_1-class)</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>IgetOffsetTypes, odml_1-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>IgetOldDataValues, odml_1-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>IgetOldDataValues, odml_1Ver-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>IgetQualifiers, odml_1-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>IgetQualityControlLevels, odml_1-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>IgetSampleMedium, odml_1-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>IgetSamples, odml_1-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>IgetValueType, odml_1-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>IgetSite, odml_1-method (odml_1-class)</td>
<td></td>
<td>24</td>
</tr>
<tr>
<td>IgetSource, odml_1-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>IgetSpatialReferences, odml_1-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>IgetSpeciation, odml_1-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>IgetSynonymID, odml_1-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>IgetTopicCategory, odml_1-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>IgetUnits, odml_1-method (odml_1-class)</td>
<td></td>
<td>24</td>
</tr>
<tr>
<td>IgetUnits, odml_1-method</td>
<td></td>
<td>24</td>
</tr>
<tr>
<td>IgetVariable, odml_1-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>IgetVariableName, odml_1-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>IgetVerticalDatum, odml_1-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>importSynonyms (Synonym transfer functions)</td>
<td></td>
<td>26</td>
</tr>
<tr>
<td>inherited_stfdf</td>
<td></td>
<td>21</td>
</tr>
<tr>
<td>inherited_stfdf (inherited_stfdf-class)</td>
<td></td>
<td>21</td>
</tr>
<tr>
<td>IupdateDataValues, odml_1-method</td>
<td>odml_1-class</td>
<td>24</td>
</tr>
<tr>
<td>odm.close, 23</td>
<td></td>
<td></td>
</tr>
<tr>
<td>odml_1-class, 24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>odml_1Ver-class (odml_1-class), 24</td>
<td></td>
<td></td>
</tr>
<tr>
<td>POSIXct, 4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>restructureDataResult (inherited_stfdf-class)</td>
<td></td>
<td>21</td>
</tr>
<tr>
<td>RObDsDat (RObsDat-package), 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RObDsDat-package, 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spatial, 21</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ST-class, 22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>STFDF, 21, 22</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Synonym transfer functions, 26</td>
<td></td>
<td></td>
</tr>
<tr>
<td>updateCV, 15, 17</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
updateCV (addCV), 3
updateDataValues (addDataValues), 4

xts, 22