Package: kwb.miacso (via r-universe)

August 26, 2024

Title functions used in KWB project MIA-CSO

Version 0.1.0

Description functions used in KWB project MIA-CSO, for example for plotting data availabilities.

License MIT + file LICENSE

URL `https://github.com/KWB-R/kwb.miacso`

BugReports `https://github.com/KWB-R/kwb.miacso/issues`

Imports kwb.db, kwb.misc, kwb.utils

Suggests testthat

Remotes github::kwb-r/kwb.db, github::kwb-r/kwb.misc, github::kwb-r/kwb.utils

Encoding UTF-8

RoxygenNote 7.1.2

Repository https://kwb-r.r-universe.dev

RemoteUrl https://github.com/KWB-R/kwb.miacso

RemoteRef HEAD

RemoteSha f789ca99ac97760438dc938529757e4889df3cac

Contents

`hmdb` *MS Access Databases (Hydraulic Data)*

Description

MS Access Databases (Hydraulic Data)

Usage

```
hmdb(...)
```

Arguments

`...`	arguments passed to `miamdb`

`hsDataSource` *Get Metadata About Data Sources*

Description

Returns a list containing

1. full path to Access database,
2. table name,
3. name of timestamp field,
4. name of parameter field

for the table that contains data of parameter `parName`, measured at monitoring point `moniPoint` in data quality level `qua.level`.

Usage

```
hsDataSource(
  qua.level = NULL,
  moniPoint = NULL,
  parName,
  kind = "q",
  owner = "KWB",
  dbg = FALSE
)
```

Arguments

`qua.level`	data quality level ("r" = raw, "v" = validated, "c" = calibrated)
`moniPoint`	name of monitoring point, e.g. "STA", "TEG", "MUE"
`parName`	name of parameter, e.g. "AFS", "CSB", "CSBf"
`kind`	kind of data: "q" = water quality, "h" = hydraulic data, "r" = rain
`owner`	owner of the data, one of "KWB", "SEN", "BWB"
`dbg`	whether to show debug messages or not

Value

Returns a list with the following named elements:

1. `mdb`: full path to Access database,
2. `tbl`: table name,
3. `tsField`: name of timestamp field,
4. `parField`: name of parameter field

Examples

```
# Get source information of validated data of parameter AFS from Stallstr.
si <- hsDataSource("v", "STA", "AFS")
si
# ouput:
#  $db
#  [1] "//moby/miacso$/Daten/ACCESS/KwbMonitoring/2VAL/KWB_STA_VAL.mdb"
#
#  $tbl
#  [1] "KWB_STA_ScanPar_AFS_VAL"
#
#  $tsfield
#  [1] "myDateTime"
#
#  $parfield
#  [1] "AFS_A"

# Access the pieces of information with the $ operator:
si$mdb # [1] "//moby/miacso$/Daten/ACCESS/KwbMonitoring/2VAL/KWB_STA_VAL.mdb"
si$tbl # [1] "KWB_STA_ScanPar_AFS_VAL"
```

`hsFpFields` *Names of Fields in Fingerprint Table*

Description

String representing comma separated list of fields in fingerprint table

Usage

```
hsFpFields(moniPoint)
```

Arguments

`moniPoint`	acronym of monitoring point: "STA", "TEG" or "MUE"

Value

String representing comma separated list of fields in fingerprint table

`hsGetFpAndValRaw` *Read Values and Fingerprints from Database*

Description

For the given parameters, the values and fingerprints are read from the database.

Usage

```
hsGetFpAndValRaw(moniPoint, parNames, year, firstDate, lastDate, home = FALSE)
```

Arguments

`moniPoint`	name of monitoring point
`parNames`	vector of parameter names
`year`	year of which data is requested
`firstDate`	first date of requested time period
`lastDate`	last date of requested time period
`home`	if `TRUE`, Hauke's home path is used instead of kwb office path

Value

Return the query result as a new data.frame (which is a special type of a list) with elements "dts" (date-times), "pars" (parameter values) and "fps" (fingerprints)

`hsGetValData` *Get Validated Data from Validated Database*

Description

2011-10-25: created

Usage

```
hsGetValData(moniPoint, parName, firstDate, lastDate)
```

Arguments

`moniPoint`	name of monitoring point, e.g. "TEG", "STA", "MUE"
`parName`	parameter name, e.g. "AFS"
`firstDate`	first date to be selected as "mm/dd/yyyy hh:nn:ss"
`lastDate`	last date to be selected as "mm/dd/yyyy hh:nn:ss"

Value

data.frame containing validated data

`hsIndexOfWavelength` *Index of Wavelength*

Description

Returns the index at which the wavelength given in nm can be found in a vector (i = 1: 200nm, i = 2: 202.5nm, ..., i = 217: 740nm).

Usage

```
hsIndexOfWavelength(wavelength)
```

Arguments

`wavelength`	Wavelength for which corresponding index shall be determined

Details

2011-12-19: moved from hsLibFingerprint.r

Value

Index corresponding the given wavelength

`hsLastWL` *Last Available Wavelength for Given Monitoring Point*

Description

Returns the last available wavelength of the spectrometer installed at the given monitoring point.

Usage

```
hsLastWL(moniPoint)
```

Arguments

`moniPoint`	Name of monitoring point, e.g. "TEG", "STA", "MUE"

Details

2011-12-19: moved from hsLibFingerprint.r

Value

Last wavelength being provided by spectrometer at given monitoring point

`hsMiaCsoDataAvailability` *Provide Information on Data Availabilities*

Description

Data availability of raw/validated/calibrated data for parameter `para`, measured at monitoring point `moniPoint` between `dateFirst` and `dateLast`.

Usage

```
hsMiaCsoDataAvailability(
  level,
  moniPoint,
  parName,
  dateFirst = NULL,
  dateLast = NULL,
  tstep = NULL,
  dbg = FALSE
)
```

Arguments

`level`	one of "r" (= raw), "v" (= validated), "c" (= calibrated)
`moniPoint`	one of "STA" (= Stallstr.), "TEG" (= Tegeler Weg), "MUE" (= Muehlendamm)
`parName`	e.g. "AFS", "CSB", "CSBf", ...
`dateFirst`	Date object representing first date to be considered
`dateLast`	Date object representing last date to be considered
`tstep`	expected time step between time stamps in seconds. Default: minimum time difference found between consecutive timestamps in given interval
`dbg`	If `TRUE`, debug messages will be shown

Examples

```
## Not run:
# Get data availability of raw data of parameter "CSBf", measured at
# monitoring point "TEG" (Tegeler Weg) between 2011-04-01 and 2011-10-01
da <- hsMiaCsoDataAvailability("r", "TEG", "CSBf", "2011-04-01", "2011-10-01")
head(da)

## End(Not run)
# Output:
#   myInterval myCount   myAvail
# 1 2011-04-01    1440 100.00000
# 2 2011-04-02    1440 100.00000
# 3 2011-04-03    1440 100.00000
# 4 2011-04-04    1440 100.00000
# 5 2011-04-05    1440 100.00000
# 6 2011-04-06    1409  97.84722
```

`hsMoniPoints` *Names of Available Monitoring Points*

Description

Returns the names of available monitoring points

Usage

```
hsMoniPoints(kind = NULL, owner = "KWB")
```

Arguments

`kind`	kind of data: "q" = water quality, "h" = hydraulic data, "r" = rain
`owner`	owner of the data, one of "KWB", "SEN", "BWB"

hsPars *Names of available parameters*

Description

Names of available parameters

Usage

```
hsPars(
  kind = NULL,
  moniPoint = NULL,
  qua.level = "c",
  owner = "KWB",
  dbg = TRUE
)
```

Arguments

`kind`	kind of data: "q" = water quality, "h" = hydraulic data, "r" = rain
`moniPoint`	name of monitoring point, e.g. "STA", "TEG", "MUE"
`qua.level`	data quality level ("r" = raw, "v" = validated, "c" = calibrated)
`owner`	owner of the data, one of "KWB", "SEN", "BWB"
`dbg`	whether to show debug messages or not

Value

Returns the names of available parameters

hsPlotAllDataAvailabilities *Plot all MIA CSO Data Availabilities*

Description

Plots availability of raw and validated data for different monitoring points and parameters. For each monitoring point a pdf file "hsDataAvailability_<MP>" where <MP> is the acronym of the monitoring point is created in the directory $strPdfDir$.

Usage

```
hsPlotAllDataAvailabilities(moniPoints, parNames, dates, pdfDir)
```

Arguments

`moniPoints`	vector containing names of monitoring points, e.g. c("TEG", "MUE")
`parNames`	vector containing names of parameters, e.g. c("AFS", "CSB", "CSBf")
`dates`	vector containing a list of Date objects
`pdfDir`	path to output directory to which created pdf-files shall be written.

Details

2012-04-17;HSB;example removed

Examples

```
## Not run:
# Generate pdf files containing data availability plots for parameters
# "AFS", "CSB", "CSBf", measured at monitoring points "MUE" (Muehlendamm),
# "TEG" (Tegeler Weg), "STA" (Stallstr.) within two different
# time intervals: 2010-01-01 to 2010-07-01 and  2011-01-01 to 2011-07-01.
hsPlotAllDataAvailabilities(
  c("MUE", "TEG", "STA"),
  c("AFS", "CSB", "CSBf"),
  as.Date(c("2010-01-01", "2010-07-01", "2011-01-01", "2011-07-01")),
  tempdir()
)

# Show data availability plots for Muehlendamm in pdf viewer
pdfFile <- file.path(tempdir(), "hsDataAvailability_MUE.pdf")
system(paste(options("pdfviewer"), pdfFile))

## End(Not run)
```

`hsPlotMiaCsoAvailabilities` *Plots availability of raw and validated data as bar plot into one plot*

Description

Plots availability of raw and validated data as bar plot into one plot

Usage

```
hsPlotMiaCsoAvailabilities(qTypes, moniPoint, parName, dateFirst, dateLast)
```

Arguments

`qTypes`	vector of data quality type codes, e.g. c("r", "v", "c"): raw, validated and calibrated data
`moniPoint`	name of monitoring point, e.g. "TEG", "MUE", "STA"
`parName`	name of parameter, e.g. "AFS", "CSB", "CSBf"
`dateFirst`	Date object representing first date to be considered
`dateLast`	Date object representing last date to be considered

`hsReadMiaMdbs` *Read MIA-CSO databases*

Description

Read MIA-CSO databases

Usage

```
hsReadMiaMdbs(root, search.new = FALSE, dbg = FALSE)
```

Arguments

`root`	root directory to start searching for new databases
`search.new`	if TRUE, root directory is searched recursively for new databases; if FALSE databases are read from R meta database
`dbg`	whether to show debug messages or not

Value

data frame with columns `mdbFile`, `mdbDesc`, `mdbDir`

`hsUpdateMiaMdbs` *Update Metadata Database*

Description

Update Metadata Database

Usage

```
hsUpdateMiaMdbs(dfMdbs, root)
```

Arguments

`dfMdbs`	data frame with columns `mdbDir`, `mdbFile` containing paths to currently known databases
`root`	path to directory from which to start looking recursively for MS Access database files

Value

Return number n of added database paths

`hsWavelengthAtIndex` *Returns the wavelength in nm that belongs to the given column index i.*

Description

2011-12-19: moved from hsLibFingerprint.r

Usage

```
hsWavelengthAtIndex(i)
```

Arguments

`i`	Index (i = 1: 200nm, i = 2: 202.5nm, ..., i = 217: 740nm)

Value

Wavelength corresponding to index

`miadir` *Experimental!*

Description

This function does not work! The intension was to get the full path to an MS Access database file by filtering for certain criteria (owner, kind of data, quality level (raw, valid, calibrated) and time resolution)...

Usage

```
miadir(
  owner = NA,
  kind = NA,
  quaLevel = NA,
  resol = NA,
  DS = kwb.misc::hsDirStructure(dbg = dbg),
  depth = 1,
  dbg = FALSE
)
```

Arguments

`owner`	data owner
`kind`	kind of data (hydraulic, quality, rain)
`quaLevel`	quality level (raw, validated, calibrated)
`resol`	resolution
`DS`	directory structure
`depth`	depth
`dbg`	whether to show debug messages or not

miamdb *Get Path to MS Access Database Used in MIA-CSO*

Description

Get Path to MS Access Database Used in MIA-CSO

Usage

```
miamdb(kind = NULL, moniPoint = NULL, qua.level = NULL, owner = "KWB")
```

Arguments

`kind`	kind of data: "q" = water quality, "h" = hydraulic data, "r" = rain
`moniPoint`	name of monitoring point, e.g. "STA", "TEG", "MUE"
`qua.level`	data quality level ("r" = raw, "v" = validated, "c" = calibrated)
`owner`	owner of the data, one of "KWB", "SEN", "BWB"

Value

This function returns the full path to the Access database containing the specified kind of data

`miamdb2` *Full Path to MIA CSO Database File*

Description

Full Path to MIA CSO Database File

Usage

```
miamdb2(id = 0)
```

Arguments

`id`	optional. Integer number identifying the database file. If not given the user is asked to enter a number on the console.

Value

full path to database file or empty string `""` if an invalid `id` was given.

`qmdb` *MS Access Databases (Water Quality Data)*

Description

MS Access Databases (Water Quality Data)

Usage

```
qmdb(...)
```

Arguments

`...`	arguments passed to `miamdb`

`rmdb` *MS Access Databases (Rain Data)*

Description

MS Access Databases (Rain Data)

Usage

```
rmdb(...)
```

Arguments

`...`	arguments passed to `miamdb`

Index