Package: kwb.read (via r-universe)

August 26, 2024

Title Functions reading specific input data, e.g. files provided by BWB

Version 0.2.0

Description Functions reading specific input data, e.g. files provided by BWB. E.g. rain data (5 min values) and rain correction tables (on daily basis).

License MIT + file LICENSE

URL https://github.com/KWB-R/kwb.read

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

Imports janitor, kwb.base, kwb.datetime, kwb.db, kwb.en13508.2, kwb.file, kwb.utils, xml2

Suggests covr, knitr, rmarkdown

VignetteBuilder knitr

Remotes github::kwb-r/kwb.base, github::kwb-r/kwb.datetime, github::kwb-r/kwb.db, github::kwb-r/kwb.en13508.2, github::kwb-r/kwb.file, github::kwb-r/kwb.utils

Encoding UTF-8

RoxygenNote 7.2.3

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

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

RemoteRef HEAD

RemoteSha d10426b68869851ff560de6ad7f58c1f58849a76

Contents

BWB_RAIN_GAUGES *BWB_RAIN_GAUGES*

Description

BWB_RAIN_GAUGES

Usage

```
BWB_RAIN_GAUGES(mdb = mdb_rain_meta())
```

Arguments

mdb	path to MS Access Database Containing Meta Data

distanceToNeighbour *Distance Matrix to Matrix of Neighbours*

Description

Convert distance matrix to matrix of neighbours

Usage

```
distanceToNeighbour(mdist)
```

Arguments

mdist	distance matrix as returned by getGaugeDistances

Value

matrix of neighbour names with row names = gauge names and column names = n1, n2, n<n.gauges - 1>

extractRunoffData *Separation of Fixed Width Columns*

Description

Separation of columns of 'columnWidth' characters width each

Usage

```
extractRunoffData(runoff_raw, columnWidth = 8, columnNames = NULL, version = 1)
```

Arguments

runoff_raw	vector of character representing the text lines containing data in columns of fixed width
columnWidth	number of characters in each column
columnNames	column names to be given to the result data frame
version	1: loop, 2: sapply

Value

The function returns a data frame with column names as given in columnNames.

gaugeNamesShort *Rain Gauge Long Name to Short Name*

Description

Rain Gauge Long Name to Short Name

Usage

```
gaugeNamesShort(colnames, underscore.rm = FALSE)
```

Arguments

colnames	Column names read from Excel sheet
underscore.rm	if TRUE, underscores are removed

getAggregatedBwbRainDataFromExcelFiles
Get Aggregated BWB Rain Data From Excel Files

Description

read BWB rain data from multiple XLS files with possible aggregation. Timestamps given in Local Time are converted to UTC+1 (= always winter time)

Usage

```
getAggregatedBwbRainDataFromExcelFiles(
  rain.xls.files,
  tableName,
  aggregation.interval = NA,
  columnNames = NULL,
  use2007Driver = FALSE,
  dbg = FALSE
)
```

Arguments

`rain.xls.files`	vector of full paths to xls files
`tableName`	name of sheet in Excel file(s) containing the rain data
`aggregation.interval`	aggregation interval in seconds. Default: NA (= no aggregation)
`columnNames`	optional. Vector of names of columns (after renaming with `gaugeNamesShort`) to return
`use2007Driver`	if TRUE, 2007 driver is used even if extension of *xls* is "xls"
`dbg`	if `TRUE`, debug messages are shown

Value

data frame with time column *DateTime*, indicating the begin of the 5-minutes interval to which the rain height corresponds.

getBwbRainDataFromExcelFiles *Get BWB Rain Data From Excel Files*

Description

Get BWB Rain Data From Excel Files

Usage

```
getBwbRainDataFromExcelFiles(
  xlsPaths,
  tableName = "Niederschlag$",
  columnNames = NULL,
  use2007Driver = FALSE,
  dbg = FALSE
)
```

Arguments

xlsPaths	(vector of) full path(s) to Excel file(s)
tableName	sheet name, default: "[Niederschlag$]"
columnNames	optional. Vector of names of columns (after renaming with gaugeNamesShort) to return
use2007Driver	if TRUE, 2007 driver is used even if extension of *xls* is "xls"
dbg	if TRUE (default), debug messages are shown

getGaugeDistances *Returns Distances Between Gauges*

Description

Returns matrix containing relative distances between gauges

Usage

```
getGaugeDistances(
  gaugeInfo = getGaugeInfo(),
 gauges = c("NknI", "NknII", "ChbI", "BlnX", "BlnIX", "Wil", "Wila", "BlnV", "BlnXI",
    "Lbg", "Hlg", "ZhlIe", "KoepIf", "Kar", "SpaII", "BlnIV"),
  dbg = FALSE
)
```

Arguments

gaugeInfo	data frame as returned by getGaugeInfo
gauges	vector of names of gauges for which distances are to be calculated
dbg	if TRUE (default), debug messages are shown

---

getGaugeInfo *Get Rain Gauge Coordinates from Rain Meta Database*

Description

Get Rain Gauge Coordinates from Rain Meta Database

Usage

```
getGaugeInfo(mdb = mdb_rain_meta())
```

Arguments

mdb	mdb containing table "tblPwInfo"

Value

data frame with columns pw, xpos, ypos

---

getPiColumnRenames *Get List of Renamings for PI-Identifiers*

Description

Get a list of assignements mapping PI identifiers to "real" (short) names

Usage

```
getPiColumnRenames(mdb = mdb_rain_meta())
```

Arguments

mdb	path to MS Access Database Containing Meta Data

Value

list of key = value pairs with the keys representing the PI-identifiers and the values representing the real (short) names

get_overview_of_columns *Get Overview of Columns in Data Frames*

Description

Get Overview of Columns in Data Frames

Usage

```
get_overview_of_columns(x)
```

Arguments

x	list of data frames

Value

data frame with each row representing one data frame in x and with as many columns as there are distinct columns in all of the data frames in x. In a row *i* the data frame contains "x" in those columns that are contained in the *i*-th data frame in list x and an empty string otherwise.

Examples

```
get_overview_of_columns(list(
  df1 = data.frame(a = 1, b = 2),
  df2 = data.frame(b = 3, c = 4),
  df3 = data.frame(a = 5, c = 6)
))
```

get_PI_data_Halensee *Get PI Data Halensee*

Description

Get PI Data Halensee

Usage

```
get_PI_data_Halensee(xls.file, sheetName = "mydata")
```

Arguments

xls.file	full path to MS Excel file containing PI data of Halensee
sheetName	name of the cell range or worksheet to read. Default: "mydata". There should be a cell range named "mydata" in the Excel file. Create this named cell range by 1. selecting one non-empty cell within the actual body of the table, 2. pressing Strg+A (select all), 3. writing "mydata" into the name field (white input field, just above the cell area and left of the "edit bar" labelled "fx"), and (iv) pressing Enter.

---

hsGetAllGsData1 *Get All Grab Sample Data (1)*

Description

get Senate's grab sample data (type 1) from all csv files in a directory and return it in a "all-in-one-table" data frame

Usage

```
hsGetAllGsData1(
  csvdir,
  sep = ";",
  dateFormat = "%d.%m.%Y",
  dec = ",",
  data.block.number = NA,
  csvFiles = dir(csvdir, "*.csv", full.names = TRUE)
)
```

Arguments

csvdir	absolute path to directory in which csv files to be read are located.
sep	separator in csv file, e.g. ";" or ","
dateFormat	date format specifier describing the format in which dates are represented in the csv file. Use placeholders , "%d" (day), "%m" (month), "%y" (2-digit year), "%Y" (4-digit year) to describe the date format. "%d.%m.%Y", "%d/%m/%y", "%Y-%m-%d" are examples for valid format specifiers.
dec	decimal character: "." or ","
data.block.number	passed to hsGetGsData1, for a description see there.
csvFiles	vector of full paths to csv files to be read. All files must contain data for the same monitoring points, in the same order! Default: all csv files in *csvdir*

See Also

hsGetGsData1

hsGetGsData1 *Read SENATE's Grab Sample Data from CSV (type 1)*

Description

SENATE's grab sample data is read from a csv file (type 1) and returned in forms of a list (see Value section below). The csv file is expected to contain grab sample data of exactly one monitoring point. For a format description of the csv file see the Details section below.

Usage

```
hsGetGsData1(
  csv,
  sep = ";",
  dateFormat = "%d.%m.%Y",
  dec = ",",
  data.block.number = NA,
  dbg = FALSE
)
```

Arguments

csv	full path to csv file
sep	separator in csv file, e.g. ";" or ","
dateFormat	date format specifier describing the format in which dates are represented in the csv file. Use placeholders , "%d" (day), "%m" (month), "%y" (2-digit year), "%Y" (4-digit year) to describe the date format. "%d.%m.%Y", "%d/%m/%y", "%Y-%m-%d" are examples for valid format specifiers.
dec	decimal character: "." or ","
data.block.number	if the file contains data for more than one monitoring point, *data.block.number* needs to be an integer number between one and the number of monitoring points for which data are provided in the file. Only data of monitoring point corresponding to the given number are extracted. If *data.block.number* is NA (default) it is expected that the file contains data for only one monitoring point. The program will stop if data is provided for more than one monitoring point
dbg	if TRUE (default), debug messages are shown

Details

The csv file is expected to look like the following example. It is structured into blocks each of which represents a measured parameter. Within each block grab sample results are organized in three columns: 1. sample date, 2. measurement value, 3. character indicating if the detection limit was underrun or exceeded. (see Parameter Sichttiefe in the example below). The third column is empty if the detection limit was not underrun or exceeded (as in almost all data lines of the example below).

```
Export Parameterwerte;

110 Mueggelspree, Faehre Rahnsdorf (0,5 m);

10001 Lufttemperatur [<degree>C];

;110 - 10001
06.02.1995;6,000
06.03.1995;5,000
...
03.12.2007;7,000

10005 Wassertemperatur [<degree>C];

;110 - 10005
06.02.1995;3,700
06.03.1995;4,600
...
03.12.2007;3,900

10010 Sichttiefe [cm];

;110 - 10010
06.02.1995;170,000
06.03.1995;80,000
...
13.11.1995;260,000;>
...
```

Value

Grab sample data is returned in forms of a list, e.g.:

```
$ moniPoint:List of 2
..$ mpID  : chr "110"
..$ mpName: chr "M<ue>ggelspree, F<ae>hre Rahnsdorf (0,5 m)"
$ byPar    :List of 104
..$ 10001  :List of 7
.. ..$ parID  : chr "10001"
.. ..$ parName: chr "Lufttemperatur"
.. ..$ parUnit: chr "<degree>C"
.. ..$ smpDate: Date[1:247], format: "1995-02-06" "1995-03-06" ...
.. ..$ parTxt : chr [1:247] "6,000" "5,000" "12,000" "8,000" ...
.. ..$ parVal : num [1:247] 6 5 12 8 4 5 24 13 16 19 ...
.. ..$ parLim : chr [1:247] "" "" "" "" ...
..$ 10005  :List of 7
.. ..$ parID  : chr "10005"
.. ..$ parName: chr "Wassertemperatur"
.. ..$ parUnit: chr "<degree>C"
```

```
.. ..$ smpDate: Date[1:247], format: "1995-02-06" "1995-03-06" ...
.. ..$ parTxt : chr [1:247] "3,700" "4,600" "7,100" "7,900" ...
.. ..$ parVal : num [1:247] 3.7 4.6 7.1 7.9 13.5 11.3 21.3 15.6 18.2 23.6 ...
.. ..$ parLim : chr [1:247] "" "" "" "" ...
..$ 10010  :List of 7
.. ..$ parID  : chr "10010"
.. ..$ parName: chr "Sichttiefe"
.. ..$ parUnit: chr "cm"
.. ..$ smpDate: Date[1:238], format: "1995-02-06" "1995-03-06" ...
.. ..$ parTxt : chr [1:238] "170,000" "80,000" "50,000" "110,000" ...
.. ..$ parVal : num [1:238] 170 80 50 110 100 190 140 180 200 200 ...
.. ..$ parLim : chr [1:238] "" "" "" "" ...
...
```

Within the sub-structures representing the parameters, `parTxt` contains the values in text format as they were read from the file, `parVal` contains the values as they were converted from text to double and `parLim` contains the characters indicating underrunning or exceedance of detection limits, as they were extracted from `parTxt`.

See Also

`hsGetGsData2`, `hsGsData1ToList`

`hsGetGsData2` *Read SENATE's Grab Sample Data From CSV (type 2)*

Description

SENATE's grab sample data is read from a csv file (type 2) and returned in forms of a list (see Value section below). The csv file is expected to contain grab sample data of more than one monitoring point, organized in sections each of which contains measurement results of all available parameters in a matrix-style format. For a format description of the csv file see the Details section below.

Usage

```
hsGetGsData2(
  csv,
  sep = ";",
  dateFormat = "%d.%m.%Y",
  dec = ",",
  blockBeginPtrn = PATTERN_STATION(),
  dbg = FALSE,
  outlevel = 1
)
```

Arguments

csv	full path to csv file
sep	separator in csv file, e.g. ";" or ","
dateFormat	date format specifier describing the format in which dates are represented in the csv file. Use placeholders , "%d" (day), "%m" (month), "%y" (2-digit year), "%Y" (4-digit year) to describe the date format. "%d.%m.%Y", "%d/%m/%y", "%Y-%m-%d" are examples for valid format specifiers.
dec	decimal character: "." or ","
blockBeginPtrn	pattern indicating the begin of a data block in the file
dbg	if TRUE (default), debug messages are shown
outlevel	Output level. Expected values are 1 or 2. The default is 1. If the output level is 2, the raw data block is passed to kwb.read:::.hsExtractData before being returned.

Details

The csv file is expected to look like the following example. It is structured into blocks each of which represents a monitoring point. Such, it is different from the format supported by hsGetGsData1 where each block represents a measured parameter and there is only one monitoring point per file. Here, a block starts with a line naming the monitoring point, followed by a header line showing the names and units of the analysed parameters. The header line is followed by data lines each of which represents a sample with sample data and the measured values according to the names in the header line. The value fields may contain a leading character indicating that the detection limit was underrun or exceeded. (Shortened) example of an input file:

```
G<ue>te - Parameterwerte (Tabelle 3);;; ...
;;; ...
Messstelle: 110 M<ue>ggelspree, F<ae>hre Rahnsdorf M<ue>ggelspree;;; ...
;TL, gesamt [<degree>C];TW, gesamt [<degree>C];Sicht, gesamt [cm];pH, gesamt [Wert]; ...
13.01.2011;3,000;0,400;;7,200;647,000;7,500;51,826;36,000;< 0,050; ...
10.02.2011;1,000;3,100;;7,600;621,000;11,300;84,231;35,000;< 0,050; ...
...
;;; ...
Messstelle: 130 Spree, F<ae>hre Baumschulenweg Spree;;; ...
;TL, gesamt [<degree>C];TW, gesamt [<degree>C];Sicht, gesamt [cm];pH, gesamt [Wert]; ...
02.11.2010;7,000;7,400;120,000;8,000;616,000;9,000;75,069;37,000; ...
30.11.2010;-4,000;2,700;150,000;7,900;608,000;10,100;74,464;36,000; ...
...
```

Value

Grab sample data is returned in forms of a list, e.g.:

```
$ moniPoint:List of 2
..$ mpID  : chr , e.g. "730"
..$ mpName: chr , e.g. "Panke, Muendung (Nordhafen-Vorbecken) (0,5 m)"
$ byPar    :List
```

```
..$ <parID1>:List of 7
.. ..$ parID  : chr , e.g. "10001"
.. ..$ parName: chr , e.g. "Lufttemperatur"
.. ..$ parUnit: chr , e.g. "<degree>C"
.. ..$ smpDate: Date[], format: ...
.. ..$ parTxt : chr [], e.g. "3,000" "4,000" "9,300" "19,900" ...
.. ..$ parVal : num [], e.g. 3 4 9.3 19.9 12.1 13.1 19.6 28 13.9 7.2 ...
.. ..$ parLim : chr [], e.g. "" "<" "<" "" ...
..$ <parID2> ...
```

See Also

hsGetGsData1, hsGsData2ToList

---

hsGsData1ToList *Transform Grab Sample Data (Type 1)*

Description

Transform grab sample data (type 1) as returned by hsGetGsData1 to "all-in-one-table"-format

Usage

```
hsGsData1ToList(gsData1, namesAsFactor = FALSE, dbg = TRUE)
```

Arguments

gsData1	grab sample data structure as returned by hsGetGsData1
namesAsFactor	passed to rbindAll
dbg	if TRUE (default), debug messages are shown

See Also

hsGetGsData1, hsGsData2ToList

hsGsData2ToList *Grab Sample Data 2 to List*

Description

Transform grab sample data (type 2) as returned by hsGetGsData2 to "all-in-one-table"-format

Usage

```
hsGsData2ToList(gsData2, dbg = FALSE)
```

Arguments

gsData2	grab sample data structure as returned by hsGetGsData2
dbg	if TRUE (default), debug messages are shown

See Also

hsGetGsData2, hsGsData1ToList

mdb_rain_meta *Path to Rain Meta Database (on KWB's server). Name of KWB server needs to be defined in environment variable* SERVERNAME

Description

Path to Rain Meta Database (on KWB's server). Name of KWB server needs to be defined in environment variable SERVERNAME

Usage

```
mdb_rain_meta()
```

merge_Neubrandenburg_data *Merge Neubrandenburg Data*

Description

Merge Neubrandenburg data into one consistent data frame

Usage

```
merge_Neubrandenburg_data(allData, keep.all = FALSE)
```

Arguments

allData	data frame
keep.all	logical. If TRUE (the default is FALSE) all columns from allData are contained in the result data frame, otherwise only the first column (timestamp).

niceStationNames *Nice Station Names*

Description

Substitutions: Berlin -> Bln, Umlaut o -> oe, spaces removed

Usage

```
niceStationNames(stations)
```

Arguments

stations	vector of character containing station (= gauge) names to be cleaned

PATTERN_DATA *Pattern matching Data*

Description

Pattern matching Data

Usage

```
PATTERN_DATA()
```

PATTERN_PARAMETER *Pattern Matching a Parameter*

Description

Pattern Matching a Parameter

Usage

```
PATTERN_PARAMETER()
```

PATTERN_STATION *Pattern Matching a Monitoring Station*

Description

Pattern Matching a Monitoring Station

Usage

```
PATTERN_STATION()
```

readAllBwbCorrections *Read and Merge BWB Correction Files*

Description

Read and Merge BWB Correction Files

Usage

```
readAllBwbCorrections(dir.in, args.general, args, dbg = TRUE)
```

Arguments

dir.in	full path to input directory where to look for CSV files to import
args.general	list of general argument settings to be used when calling read_BWB_rain_correction. For available argument names, see the help pages of the sub-functions read_BWB_rain_correction_long read_BWB_rain_correction_wide_day, read_BWB_rain_correction_wide_gauge.
args	list of argument settings for each file. The elements in the list are expected to be the names of files in dir.in.
dbg	if TRUE (default), debug messages are shown

readAllBwbCumulativeHeights *Read and Combine Rain Heights*

Description

Read rain height files with readBwbRainFromCsv2, combine them by rows, remove duplicates and order by timestamp

Usage

```
readAllBwbCumulativeHeights(files.data, ..., dbg = TRUE)
```

Arguments

files.data	character vector containing the full paths to the CSV files to be read
...	arguments passed to readBwbRainFromCsv2
dbg	if TRUE debug messages are shown

readAllBwbSignals *Read and Merge BWB Rain Data Files*

Description

Read and Merge BWB rain data files with readBwbRainFromCsv2

Usage

```
readAllBwbSignals(files.data, date.format = "%Y-%m-%d", dbg = TRUE)
```

Arguments

files.data	character vector containing the full paths to the CSV files to be read
date.format	format used to interpret the date string. See format.POSIXct
dbg	if TRUE debug messages are shown

readBwbCorrFromCsv *Read BWB Rain Correction Data from CSV File*

Description

Read BWB rain correction data from a CSV file that has been created by saving the corresponding Excel file as CSV

Usage

```
readBwbCorrFromCsv(file, sep = ",", country = "de", format = "%d.%m.%Y")
```

Arguments

file	path to CSV file
sep	column separator
country	contry code specifying decimal and thousands separators, one
format	date format, passed to as.Date

readBwbCorrFromExcel *Read BWB Rain Correction Data from Excel File*

Description

Read BWB Rain Correction Data from Excel File

Usage

```
readBwbCorrFromExcel(
  file,
  tblCorr = "Bericht 1$",
  tblGaug = "gaugeNames",
  dbg = TRUE
)
```

Arguments

file	full path to Excel file
tblCorr	name of sheet containing correction data. Default: "Bericht 1$"
tblGaug	name of sheet containing gauge Names. Default: "gaugeNames$"
dbg	logical. If TRUE, debug messages are shown.

readBwbRainCorrection *Read BWB Rain Correction Data*

Description

read BWB rain correction data from Excel file or CSV file

Usage

```
readBwbRainCorrection(file, ..., zerolines.rm = TRUE, dbg = FALSE)
```

Arguments

file	full path to Excel file or CSV file
...	arguments given to readBwbCorrFromExcel or readBwbCorrFromCSV
zerolines.rm	logical. If TRUE, rows in which the sum of absolute values is zero, are removed
dbg	logical. If TRUE, debug messages are shown.

readBwbRainData *Read BWB Rain Data*

Description

Read BWB Rain Data

Usage

```
readBwbRainData(
  file,
  tableName = "Niederschlag$",
  columnNames = NULL,
  toUTC = TRUE,
  toNormaltime = FALSE,
  dbg = TRUE,
  use2007Driver = kwb.db::isExcel2007File(file),
  ...
)
```

Arguments

file	full path to Excel file
tableName	sheet name, default: "[Niederschlag$]"
columnNames	optional. Vector of names of columns (after renaming with gaugeNamesShort) to return
toUTC	default: TRUE
toNormaltime	if TRUE, a column *tBeg_WT* is created containing the begin of the time interval in Normal time (winter time, UTC+01). default: FALSE
dbg	default: TRUE
use2007Driver	if TRUE, 2007 driver is used even if extension of *file* is ".xls"
...	additional parameters passed to readBwbRainFromCsv

---

readBwbRainFromCsv *Read BWB Rain Data from CSV File*

Description

read BWB rain data from a CSV file that has been created by saving the corresponding Excel file as CSV

Usage

```
readBwbRainFromCsv(
  file,
  sep = ";",
  dec = ",",
  format = "%d.%m.%y %H:%M:%S"
)
```

Arguments

file	path to CSV file
sep	column separator
dec	decimal character
format	date format string, passed to as.POSIXct

readBwbRainFromCsv2 *Read BWB rain data from CSV file*

Description

Read BWB rain data from CSV file

Usage

```
readBwbRainFromCsv2(
  file,
  sep = ";",
  country = "de",
  date.format = "%d.%m.%Y %H:%M",
  dbg = TRUE
)
```

Arguments

file	full path to CSV file
sep	column separator
country	"en" for English number format, "de" for German number format
date.format	date format string, passed to reformatTimestamp
dbg	if TRUE (default), debug messages are shown

readPackageFile *Read CSV File from Package's "extdata" Folder*

Description

Read CSV File from Package's "extdata" Folder

Usage

```
readPackageFile(file, ...)
```

Arguments

file	file name (without path)
...	additional arguments passed to read.csv

Value

data frame representing the content of file

readYearlyRainHeightsFromOneCsvFile

Read Yearly Rain Heights From One Csv File

Description

reads a CSV file containing daily rain heights as provided by BWB (Mario Grunwald). Example filename: "Niederschlaege_1994__BERICHT.csv". The files are expected to contain a header of three rows (first row: DATUM/station names, second row: variable name ["Regenhoehe" or "Regendauer"], third row: unit ["in mm" or "in mm"])

Usage

```
readYearlyRainHeightsFromOneCsvFile(csv, sep = ",", dateformat = "%d/%m/%Y")
```

Arguments

csv	full path to csv file
sep	column separator. default: comma ","
dateformat	date format, default: "%d/%m/%Y"

Value

data frame with columns...

read_BWB_rain_correction

Read Correction Data

Description

Read BWB's rain correction data (different formats supported)

Usage

```
read_BWB_rain_correction(files, type = (BWB_rain_correction_types())[1], ...)
```

Arguments

files	paths to input files
type	file type, must be one of the type strings returned by kwb.read:::BWB_rain_correction_types
...	arguments passed to the read-function corresponding to the given type

read_BWB_rain_correction_long *Read BWB Rain Correction Data*

Description

Read BWB Rain Correction Data in "long" format. The data is provided by BWB in the form of an Excel file. Save the file to CSV and run this function on that CSV file.

Usage

```
read_BWB_rain_correction_long(
  files,
  sep = ";",
  country = c("de", "en")[1],
  date.format = "%A %d. %B %Y",
  locale = .localeString(country),
  wide = TRUE,
  dbg = TRUE
)
```

Arguments

files	full path to CSV file containing rain height correction data in "long" format
sep	column separator in files
country	one of "de" (German) or "en" (English) according to the format numeric strings are given in
date.format	format used to convert the date string into a date object
locale	locale string, passed to read_BWB_rain_correction_long
wide	if TRUE (default) the data will be returned in "wide" format, i.e. with the date in the first column and each further column representing a rain gauge
dbg	logical. If TRUE, debug messages are shown.

Value

data frame either in "wide" format (with the date in the first column and the rain heights in the following columns named according to the "object ids" of the rain gauges) or in "long" format (with columns Date, ObjNr Pumpwerk Niederschlag). The mapping between object ids and gauge names is returned in the attribute "gauges".

Examples

```
## Not run:
# Provide the path to an example file in this package
file <- "Niederschlag_example.csv"
file <- system.file("extdata", file, package = "kwb.read")
```

```
# Read the CSV file into a data frame in "long" format
correction.wide <- read_BWB_rain_correction_long(file)

# Read the CSV file into a data frame in "wide" format (gauges in columns)
correction.long <- read_BWB_rain_correction_long(file, wide = FALSE)

# Get the mapping between object IDs and gauge names from attribute "gauges"
kwb.utils::getAttribute(correction.wide, "gauges")

## End(Not run)
```

`read_BWB_rain_correction_wide_day` *Read Correction "Wide" Format*

Description

Read correction data from CSV file(s) in "wide" format with days in columns

Usage

```
read_BWB_rain_correction_wide_day(
  files,
  sep = ";",
  country = "de",
  transpose = TRUE,
  dbg = TRUE,
  date.format = c("%d %B %Y")[1],
  encoding = ""
)
```

Arguments

`files`	paths to rain correction files
`sep`	column separator
`country`	one of "de" (German) or "en" (English) according to the format numeric strings are given in
`transpose`	logical. If `TRUE` (default) data are transposed so that the different rain gauges appear in different columns
`dbg`	logical. If `TRUE`, debug messages are shown.
`date.format`	format used to convert the date string into a date object
`encoding`	Encoding used when reading the CSV file. Best would be to use "UTF-8" but MS Excel uses system default

---

`read_BWB_rain_correction_wide_gauge`
Read Corrections in "wide" format

Description

Read correction data from CSV file(s) in "wide" format with gauges in columns

Usage

```
read_BWB_rain_correction_wide_gauge(
  files,
  sep = ";",
  country = "de",
  date.format = c("%d.%m.%Y", "%A %d. %B %Y")[1],
  dbg = TRUE
)
```

Arguments

`files`	paths to rain correction files
`sep`	column separator
`country`	one of "de" (German) or "en" (English) according to the format numeric strings are given in
`date.format`	format used to convert the date string into a date object
`dbg`	logical. If `TRUE`, debug messages are shown.

`read_BWB_rain_correction_wide_gauge_from_pdf`
Read Corrections in "Wide" Format

Description

Read correction data from CSV file(s) in "wide" format with gauges in columns. Version 2: As generated from PDF-file with "pdf to Excel"

Usage

```
read_BWB_rain_correction_wide_gauge_from_pdf(file, sep = ";", ...)
```

Arguments

`file`	path(s) to CSV file(s)
`sep`	column separator
`...`	passed to `kwb.read:::.extract_wide_gauge`

read_BWB_rain_correction_wide_gauge_group *Read Correction Data*

Description

Read Correction Data in "wide" format, grouped by groups of pumping stations

Usage

```
read_BWB_rain_correction_wide_gauge_group(
  file,
  encoding = default_encoding(),
  ...
)
```

Arguments

file	path to rain correction data file
encoding	file encoding, passed to readLines
...	arguments passed to kwb.read:::.extract_wide_gauge_group

read_cctv_from_xml_be *Read CCTV Data from XML File(s) in Belgian Standard Format*

Description

Read CCTV Data from XML File(s) in Belgian Standard Format

Usage

```
read_cctv_from_xml_be(files, as_is = FALSE, dbg = TRUE)
```

Arguments

files	(vector of) path(s) to XML file(s)
as_is	if TRUE the column names in the table of observations are more or less kept as they are in the xml files, otherwise, they are renamed to the names that are defined in the EN 13508.2 standard. These names are e.g. required by the RERAU evaluation functions in kwb.rerau. The default is FALSE.
dbg	if TRUE (default), debug messages are shown

read_Neubrandenburg_Data

Join Data from Multiple Excel Files

Description

Join Data from Multiple Excel Files

Usage

```
read_Neubrandenburg_Data(
  xls.dir,
  area.Gruendach.m2,
  area.Kiesdach.m2,
  pattern = "\\.xlsx$"
)
```

Arguments

xls.dir	path to directory containing Excel files to be read
area.Gruendach.m2	area of green roof in square metres. Used to convert between litres and litres per square metres (mm) as given in the result data frame
area.Kiesdach.m2	area of pebble roof in square metres. Used to convert between litres and litres per square metres (mm) as given in the result data frame
pattern	pattern matching the names of files in xls.dir to be imported

read_Neubrandenburg_Data_from_Excel

Read Neubrandenburg Data

Description

Read Neubrandenburg data from Excel file

Usage

```
read_Neubrandenburg_Data_from_Excel(xls)
```

Arguments

xls	full path to MS Excel file containing Neubrandenburg rain gauge and green roof discharge data

Value

data frame with columns *DateTime* (representing Local Normal Time but with attribute tzone = "UTC"), *NSB*: Niederschlag, *KFS*: Kiesdach (Kippe, Fallrohr, Stein), *KFZ*: Gruendach (Kippe, Fallrohr, Zinko)

---

read_Neubrandenburg_Data_from_Excel_files *Read Neubrandenburg Data from Excel Files*

Description

Read and join Neubrandenburg data from multiple Excel files

Usage

```
read_Neubrandenburg_Data_from_Excel_files(xls.files)
```

Arguments

xls.files	vector of (full paths to) MS Excel files containing Neubrandenburg data

---

read_qin_file *Read "qin"-File*

Description

Read data, extract starttime, timestep, manhole IDs and runoff data

Usage

```
read_qin_file(infile, columnWidth = 8, n = -1, headerSize = 528, dbg = TRUE)
```

Arguments

infile	full path to input file.
columnWidth	width of the data columns. It is assumed that all data columns have the same width!
n	number of (data) rows to read. Set to -1 if all rows are to be read. Default: -1
headerSize	number of header rows preceeding the data rows in *infile*
dbg	if TRUE, debug messages are shown, else not

read_rain *Read Rain Data from BWB Excel Files*

Description

Read Rain Data from BWB Excel Files

Usage

```
read_rain(
  directory = kwb.utils::resolve("RAIN_DIR", dictionary),
  pattern = kwb.utils::resolve("RAIN_XLS_PATTERN", dictionary),
  tableName = kwb.utils::resolve("RAIN_XLS_TABLE", dictionary),
  dictionary = kwb.utils::selectElements(settings, "dictionary"),
  rain.xls.files = dir(directory, pattern, full.names = TRUE),
 aggregation.interval = kwb.utils::selectElements(settings, "rain.aggregation.interval"),
  settings = NULL,
  use2007Driver = TRUE
)
```

Arguments

directory	directory containing excel files from which rain data are read. Default: kwb.utils::resolve("RAIN_DIR dictionary)
pattern	file name pattern matching names of files to be read. Default: kwb.utils::resolve("RAIN_XLS_PATTERI dictionary)
tableName	name of sheet in the excel files containing the rain data. Default: kwb.utils::resolve("RAIN_XLS_TABL dictionary)
dictionary	list of key/value pairs that can be used as a dictionary to look up the following keywords (if *directory*, *pattern*, *tableName* are not given): RAIN_DIR, RAIN_XLS_PATTERN, RAIN_XLS_TABLE
rain.xls.files	full paths to xls or xlsx files to be read. If given, these paths are used. Otherwise all files in *directory* matching the filename *pattern* are used.
aggregation.interval	aggregation interval in seconds. Default: settings$rain.aggregation.interval
settings	list that may contain arguments to this function in elements rain.aggregation.interval or dictionary
use2007Driver	passed to getAggregatedBwbRainDataFromExcelFiles

Value

data frame with column DateTime containing timestamps in UTC timezone but representing Berlin normal (= winter) time.

`read_SenatesGrabSampleCsvFiles_long`
Read Senates' Grab Sample Files

Description

read Senates' grab sample files ('long' format). For a format description see `hsGetGsData1`

Usage

```
read_SenatesGrabSampleCsvFiles_long(csvFiles, ...)
```

Arguments

`csvFiles`	full paths to CSV files to be read
`...`	arguments passed to `read_SenatesGrabSampleCSV_long`

`read_SenatesGrabSampleCsvFiles_wide`
Read Senates' Grab Sample Files

Description

Read Senates' grab sample files ('wide' format). For a format description see `hsGetGsData2`

Usage

```
read_SenatesGrabSampleCsvFiles_wide(
  csvs,
  sep,
  dateFormat,
  country = "de",
  dbg = TRUE,
  oldversion = FALSE
)
```

Arguments

`csvs`	vector of paths to CSV files
`sep`	column separator
`dateFormat`	date format string, passed to `read_SenatesGrabSampleCSV_wide` and `kwb.read:::.doConversion`
`country`	one of "de" (German) or "en" (English) according to the format numeric strings are given in
`dbg`	logical. If `TRUE`, debug messages are shown.
`oldversion`	passed to `kwb.read:::.doConversion`

read_SenatesGrabSampleCSV_long

Read Senates' Grab Sample File

Description

Read Senates' grab sample file ('long' format). For a format description see hsGetGsData1

Usage

```
read_SenatesGrabSampleCSV_long(
  csv,
  sep = kwb.utils::guessSeparator(csv),
  dateFormat,
  dec = ifelse(sep == ",", ".", ","),
  dbg = FALSE
)
```

Arguments

csv	path to CSV file
sep	column separator
dateFormat	date format string, passed to kwb.read:::.removeNonDataRows and kwb.read:::.convertDataTypes
dec	decimal character
dbg	logical. If TRUE, debug messages are shown.

read_SenatesGrabSampleCSV_wide

Read Senates' Grab Sample File

Description

Read Senates' grab sample file ('wide' format). For a format description see hsGetGsData2

Usage

```
read_SenatesGrabSampleCSV_wide(
  csv,
  sep,
  dateFormat,
  country = "de",
  doConversion = TRUE,
  dbg = TRUE,
  oldversion = FALSE
)
```

Arguments

csv	path to CSV file
sep	column separator
dateFormat	date format string, passed to kwb.read:::.doConversion if doConversion is TRUE
country	one of "de" (German) or "en" (English) according to the format numeric strings are given in
doConversion	logical. If TRUE, the function kwb.read:::.doConversion is called on the result
dbg	logical. If TRUE, debug messages are shown.
oldversion	passed to kwb.read:::.doConversion if doConversion is TRUE

---

read_xml_as_path_value

Read XML File as Pairs of Paths and Values

Description

Read XML File as Pairs of Paths and Values

Usage

```
read_xml_as_path_value(xml, dbg = TRUE, max_length = 100)
```

Arguments

xml	path, url or literal xml or anything else that is accepted by read_xml as argument x
dbg	if TRUE, debug messages are shown
max_length	maximum number of characters reserved for printing the path to the folder of the xml file in the debug message

Value

data frame with the full paths to the XML elements in the first and the text values of the XML elements in the second column.

Examples

```
url <- "https://www.w3schools.com/xml/note.xml"

# Original XML content
kwb.utils::catLines(readLines(url, warn = FALSE))

# Interpretation as Paths and Values
kwb.read:::read_xml_as_path_value(url)
```

splitIntoBlocks *Split Character Vector Into Blocks of Lines*

Description

Split Character Vector Into Blocks of Lines

Usage

```
splitIntoBlocks(x, pattern, offset.start = 0, offset.stop = 0)
```

Arguments

x	vector of lines of character
pattern	pattern maching the start lines of the blocks
offset.start	offset added to the indices matching the block starts to identify the first lines of the blocks to be cut
offset.stop	offset defining the end of the blocks. A value of zero (default) means that a block ends one line before the start of the next block (a line matching pattern)

toGaugeInfo *BWB Rain Data Column Name to Gauge Info Table*

Description

Convert a vector of BWB rain data column names "id name - Regenmesser" to a data frame with columns id and <name>

Usage

```
toGaugeInfo(x)
```

Arguments

x	vector of column names of the form "01.02 <gauge> - Regenmesser"

Index