# Package: kwb.monitoring (via r-universe)

October 17, 2024

**Title** Functions Used Within Different Kwb Monitoring Projects

**Version** 0.2.0

**Description** Functions used within different KWB projects dealing with monitoring data.

**License** MIT + file LICENSE

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

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

**Imports** gplots, kwb.base, kwb.datetime, kwb.event, kwb.plot, kwb.utils, lattice, lubridate, manipulate

**Suggests** knitr, kwb.logger, rmarkdown, testthat

**VignetteBuilder** knitr

**Remotes** github::kwb-r/kwb.base, github::kwb-r/kwb.datetime, github::kwb-r/kwb.event, github::kwb-r/kwb.logger, github::kwb-r/kwb.plot, github::kwb-r/kwb.utils

**Encoding** UTF-8

**RoxygenNote** 6.1.1

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

**RemoteUrl** https://github.com/KWB-R/kwb.monitoring

**RemoteRef** HEAD

**RemoteSha** 6aefc1d076c4cfa8ffbdd1abf8f143f7deb8b9e5

## Contents

.plotRainData . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 3
.toIntervalList . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
addFakeEntriesForDaysWithoutData . . . . . . . . . . . . . . . . . . . . . . . . . . . 4
addSampleTimesToPlot . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
addStatisticsToEvents . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 5
addSumRow . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
addZoomToHistory . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 6

appendColumn_samplesOk . . . 7
appendInterpolColumns . . . 7
apply_H_threshold . . . 8
availableAutoSamplerFiles . . . 8
availableAutoSamplerFiles2 . . . 9
bottleEventsToSamplerEvents . . . 9
calculateVolumeCompositeSample . . . 10
configure . . . 10
createDummyEventThresholdFiles . . . 12
dateToDateStringInPath . . . 13
filterForRelevantBottles . . . 13
filterSampleEventsForFilename . . . 14
formatEvent . . . 14
formatEventStatistics . . . 15
formatEventStatisticsTable . . . 15
formatSettings . . . 16
getAllSamplerEvents . . . 16
getAllTypesOfEvents . . . 17
getFunctionValueOrDefault2 . . . 17
getHydraulicEvents . . . 18
getIndicesWithinEvents . . . 18
getMergedEvents . . . 19
getModelFromFile . . . 19
getOrCreatePath . . . 20
getPredictionOfQ . . . 20
getSampleInformation . . . 21
getSamplerEvents . . . 22
getStatisticsByDay . . . 22
getStatisticsByEvent . . . 23
H_above_threshold . . . 23
indicesInIntervals . . . 24
mergeParallelRainEventStat . . . 24
pathDictionary . . . 25
plotEventDistribution . . . 25
plotEventOverview . . . 26
plotOverview . . . 26
plotSampleInformation . . . 27
plotTotalDischargeVersusRainProperties . . . 27
plot_hydraulic_event . . . 28
plot_hydraulic_events . . . 29
plot_H_columns . . . 29
plot_sampled_event . . . 30
printSampleInformation . . . 31
rainGaugesNearStation . . . 31
readAndJoinSamplerFiles . . . 32
removeDuplicates . . . 32
removeIntervals . . . 33
removeZoomFromHistory . . . 33

sampleDataToSamplingEvents . . . . . 34
sampleLogFileToSampleName . . . . . 34
samplingEventsToBottleEvents . . . . . 35
saveRegressionModel . . . . . 35
saveSampleInformation . . . . . 36
selectIntervalsForCorrelation . . . . . 36
setPausesOfMergedEvents . . . . . 37
showOverview . . . . . 37
updateZoomHistory . . . . . 38
validateAndFillHydraulicData . . . . . 38
whichAboveThresholds . . . . . 39
writeCsvToPathFromDictionary . . . . . 40
writeDataAndOpenDirectory . . . . . 40

**Index** **41**

## `.plotRainData` *Plot Rain Data*

### Description

Plot Rain Data

### Usage

```
.plotRainData(rainData, gauges, xlim, innerMargins, eventAndStat = NULL)
```

### Arguments

| | |
|---|---|
| `rainData` | data frame with columns *DateTime* |
| `gauges` | character vector of gauge names |
| `xlim` | vector of two POSIXct determining the x limits |
| `innerMargins` | passed to `plotRain` |
| `eventAndStat` | if not NULL (default) this must be a one row data frame with columns ... |

## .toIntervalList *Character Vectors to List of "lubridate" Intervals*

### Description

Converts two character vectors representing the beginning and end timestamps of time intervals into a list of lubridate interval objects created by `interval`

### Usage

```
.toIntervalList(from = NULL, to = NULL, tzone = "UTC")
```

### Arguments

| | |
|---|---|
| `from` | vector of timestamps (vector of character will be converted to vector of POSIXct using `kwb.datetime::stringToPosix`) representing the starts of the intervals |
| `to` | vector of timestamps (vector of character will be converted to vector of POSIXct using `kwb.datetime::stringToPosix`) representing the ends of the intervals |
| `tzone` | passed to `kwb.datetime::stringToPosix` and `lubridate::interval` |

## addFakeEntriesForDaysWithoutData *Add Fake Entries for Days Without Data*

### Description

Add Fake Entries for Days Without Data

### Usage

```
addFakeEntriesForDaysWithoutData(dataFrame, parameterName = "Q")
```

### Arguments

| | |
|---|---|
| `dataFrame` | data frame with columns *day*, *parName*, *parVal* |
| `parameterName` | name of parameter, default: `"Q"` |

## addSampleTimesToPlot *Add Sample Times to Plot*

### Description

Add Sample Times to Plot

### Usage

```
addSampleTimesToPlot(sampleTimes, ymax,
  timeFormat = default_day_time_format(), cex.text = 0.7,
  showArrows = FALSE, showTimes = FALSE, showBottleNumbers = FALSE,
  legendPosition = "bottom")
```

### Arguments

| | |
|---|---|
| sampleTimes | data frame with columns sampleTime, bottle, result |
| ymax | maximum y value of plot. Used to determine arrow lengths and positions |
| timeFormat | default: "%d.%m %H:%M" |
| cex.text | default: 0.7 |
| showArrows | default: FALSE |
| showTimes | default: FALSE |
| showBottleNumbers | default: FALSE |
| legendPosition | character string specifying the legend position, e.g. "bottom", "[top\|bottom]left", "top", "[top\|bottom]right" |

## addStatisticsToEvents *Add Statistics to Events*

### Description

Add Statistics to Events

### Usage

```
addStatisticsToEvents(events, hydraulicData)
```

### Arguments

| | |
|---|---|
| events | list with elements hydraulic and merged each of which is a data frame containing event information |
| hydraulicData | passed to getStatisticsByEvent |

## `addSumRow` *Add Sum Row*

### Description

Add Sum Row

### Usage

```
addSumRow(x)
```

### Arguments

| | |
|---|---|
| `x` | matrix or data frame to which a sum row is added as its last row |

## `addZoomToHistory` *Add Zoom to History*

### Description

Add Zoom to History

### Usage

```
addZoomToHistory(zoomHistory, i, j)
```

### Arguments

| | |
|---|---|
| `zoomHistory` | list of (i, j) pairs, storing the history of zoom settings |
| `i` | index in range of slider "left" |
| `j` | index in range of slider "right" |

## appendColumn_samplesOk *Append Column "samplesOk"*

### Description

Append Column "samplesOk"

### Usage

```
appendColumn_samplesOk(samplingEvents, bottleEvents,
  successWord = "SUCCESS", columnName = "samplesOk")
```

### Arguments

| | |
|---|---|
| `samplingEvents` | data frame with columns `samplerFile`, `bottle`, `result` |
| `bottleEvents` | bottle events (data frame) |
| `successWord` | word indicating a successful sampling in column `result` of `samplingEvents` |
| `columnName` | name of appended column |

## appendInterpolColumns *Append Interpol Columns*

### Description

Append Interpol Columns

### Usage

```
appendInterpolColumns(hydraulicData, settings, columnQraw = "Q.raw",
  columnQ = "Q", columnQInterpol = "Q.interpol",
  columnHraw = "H.raw", columnH = "H",
  columnHInterpol = "H.interpol")
```

### Arguments

| | |
|---|---|
| `hydraulicData` | data frame containing hydraulic data |
| `settings` | list of settings (not used!) |
| `columnQraw` | name of column containing raw discharge |
| `columnQ` | name of column containing (valid) discharge |
| `columnQInterpol` | name of column to be added containing interpolated discharges |
| `columnHraw` | name of column containing raw water levels |
| `columnH` | name of column containing (valid) water levels |
| `columnHInterpol` | name of column to be added containing interpolated water levels |

### Value

data frame with columns *H.interpol*, *Q.interpol* appended, containing only the interpolated values and NA for times when H (or Q, respectively) was already available in column *H.raw* (or *Q.raw*, respectively)

## apply_H_threshold *Apply H Threshold*

### Description

Apply H threshold given in settings to H in *dat.raw*

### Usage

```
apply_H_threshold(dat.raw, settings)
```

### Arguments

| | |
|---|---|
| `dat.raw` | data frame with column *H* |
| `settings` | list as returned by `configure` with list element *Hthresholds* |

## availableAutoSamplerFiles *Available "sample_log"-Files*

### Description

Available "sample_log"-Files

### Usage

```
availableAutoSamplerFiles(sampleDirectory = getOrCreatePath("SAMPLE_DIR",
dictionary), pattern = do_resolve("SAMPLE_CSV_PATTERN", dictionary),
dictionary = NULL, warn = TRUE)
```

### Arguments

| | |
|---|---|
| `sampleDirectory` | directory in which to look for sample files. By default the directory is looked up in the *dictionary* at keyword: SAMPLE_DIR |
| `pattern` | file name pattern to which file names are matched. By default pattern is looked up in the *dictionary* at keyword: SAMPLE_CSV_PATTERN |
| `dictionary` | dictionary (list) with elements *SAMPLE_DIR* and *SAMPLE_CSV_PATTERN* |
| `warn` | if TRUE, a warning is given if there are no sample files |

## availableAutoSamplerFiles2 — *Available Auto Sampler Files 2*

### Description

Available Auto Sampler Files 2

### Usage

```
availableAutoSamplerFiles2(rootDirectory, station, dictionaryFile)
```

### Arguments

| | |
|---|---|
| `rootDirectory` | passed as `RAW_DIR` to `pathDictionary` |
| `station` | passed as `STATION` to `pathDictionary` |
| `dictionaryFile` | passed to `pathDictionary` |

## bottleEventsToSamplerEvents — *Bottle Events to Sampler Events*

### Description

Bottle Events to Sampler Events

### Usage

```
bottleEventsToSamplerEvents(bottleEvents, signalWidth)
```

### Arguments

| | |
|---|---|
| `bottleEvents` | data frame with columns *samplerFile* |
| `signalWidth` | passed to `toEvents` |

## calculateVolumeCompositeSample *Calculate Volume Composite Sample*

### Description

Calculate Volume Composite Sample

### Usage

```
calculateVolumeCompositeSample(hydraulicSubset, settings,
  show.all.bottles = FALSE)
```

### Arguments

`hydraulicSubset`
: data frame with columns *DateTime*, *Q*, *bottle*

`settings`
: settings as returned by `configure`

`show.all.bottles`
: if `TRUE`, the volume table is shown not only for the selected bottles but also for all bottles (inclusive discarded ones). Default: `FALSE`

## configure *Configure*

### Description

Generate a configuration for an analysis run

### Usage

```
configure(rawdir, station, sampleEventIndex = -1,
  sampleEventMethod = "centre", replaceMissingQMethod = "interpolate",
  bottlesToDiscard = NA, Vbottle = 1600, Vmax = 5000,
  Hthresholds = 0, Qthresholds = NULL, Vthresholds = NULL,
  tstep.fill.s = 60, evtSepTime = 2 * 3600,
  sampleEventSeparationTime = NA, durationThreshold = 5,
  outsep = ";", outdec = ",", context = c(left = 0.1, right = 0.2),
  plotchars = c(1, 3, 4, 4), rain.aggregation.interval = 600,
  max.samples.ok = 4, bottlesToConsider = NA, dictionaryFile = "")
```

## Arguments

`rawdir` Where is the "root" directory to raw data?

`station` which monitoring station? E.g in OGRE: one of c("EFH", "STR", "ALT", "NEU", "GEW").

`sampleEventIndex`
which sample event (logged by sampler), according to the list of files recorded by the autosampler, sorted by name. Give a negative number to select files from the end of the list of files: -1 means "last", -2 "one before last", etc.

`sampleEventMethod`
one of c("centre", "left", "right"). "left": sample time is begin of time interval, "right": sample time is end of time interval, "centre": sample time is middle of time interval

`replaceMissingQMethod`
one of c("interpolate", "predict"). "interpolate": linear interpolation "predict": prediction from water levels using a saved square regression

`bottlesToDiscard`
which bottles are to be discarded (because they are not full)? Default: NA

`Vbottle` sample volume (in mL) given to the bottle representing the time interval with highest flow volume. Default: 1600

`Vmax` maximum total volume for mixed sample, in mL. Default: 5000

`Hthresholds` What are the level thresholds (in m) that trigger the start of the sampler? Named vector of numeric with names representing the site codes. E.g. for project OGRE: c("EFH", "STR", "ALT", "NEU", "GEW"). Default: 0

`Qthresholds` flow thresholds for each station. Mark flow tresholds to define runoff events, same structure as Hthresholds

`Vthresholds` hydraulic event volume thresholds for each station

`tstep.fill.s` time step in seconds used to fill up data. Default: 60 seconds = 1 minute.

`evtSepTime` separation of events (how long in seconds may Hthreshold be underrun within an event?). Default: 2*3600 (= 2 hours)

`sampleEventSeparationTime`
separation of sampled events within one and the same sampler file. If the difference between two sample times is greater than this time (in seconds) two sampled events are distinguished. Set to NA to prevent the splitting of sampled events.

`durationThreshold`
minimum duration (in minutes) of hydraulic events to be considered. Default: 5 minutes

`outsep` separator to be used in output files (csv). Default: ";"

`outdec` decimal character to be used in output files (csv). Default: ","

`context` Vector of two elements giving the "context" before and after the event to be plotted, as percentages of the event duration. E.g. c(0.1, 0.2) means that the time interval to be plotted starts 10 percent of the event duration before the event begin and ends 20 percent of the event duration after the end of the event.

`plotchars` plotting characters for Q.raw, Q.signal, ...

`rain.aggregation.interval`
time interval in seconds for rain data aggregation, e.g. 600 = 10 minutes. NA = no aggregation of original rain data

`max.samples.ok` maximum number of "successful" samples within one bottle. Used for scaling the bottle "height" in the sample plot

`bottlesToConsider`
numeric vector of bottle numbers. Only bottles of the given numbers are considered (read from the sampler file). Set to NA to consider all bottles

`dictionaryFile` full path to "dictionary" file that defines the folder structure and file name patterns in which the input files are expected to reside and to which the output files will be written. See kwb.ogre::OGRE_DEFAULT_DICTIONARY or kwb.dswt::DSWT_DEFAULT_DIC

---

`createDummyEventThresholdFiles`
*Create Threshold Dummy Files*

---

## Description

Create dummy files defining H and Q thresholds for different time intervals

## Usage

```
createDummyEventThresholdFiles(stations = names(kwb.utils::selectElements(settings,
 "Hthresholds")), outdir = file.path(kwb.utils::selectElements(settings,
 "rawdir"), "..", "META"), settings = NULL)
```

## Arguments

`stations` names of monitoring stations

`outdir` path to output directory

`settings` list from which to take non-given arguments

## Value

returns a list (with the stations as element names) containing the paths to the created files.

## `dateToDateStringInPath` *Date to Date String in Path*

### Description

Date to Date String in Path

### Usage

```
dateToDateStringInPath(x)
```

### Arguments

| | |
|---|---|
| `x` | date or time object passed to `hsDateStr` |

## `filterForRelevantBottles` *Filter Data Frame by Bottle Number*

### Description

Filter Data Frame by Bottle Number

### Usage

```
filterForRelevantBottles(sampleDataExtended, bottlesToConsider)
```

### Arguments

| | |
|---|---|
| `sampleDataExtended` | dataFrame with column *bottle* |
| `bottlesToConsider` | vector of bottle numbers to be considered |

### Value

data frame with rows in which column *bottle* is one of the numbers given in *bottlesToConsider*

## `filterSampleEventsForFilename` *Filter Sample Events for Filename*

### Description

Filter Sample Events for Filename

### Usage

```
filterSampleEventsForFilename(events, fileName)
```

### Arguments

| | |
|---|---|
| `events` | list with required elements *samplingEvents*, *bottleEvents*, *samplerEvents* |
| `fileName` | file name to be filtered for in column *samplerFile* of each of the data frames *events$samplingEvents*, *events$bottleEvents* and *events$samplerEvents* |

### Value

list with elements *samplingEvents*, *bottleEvents*, *samplerEvents*

## `formatEvent` *Format Event*

### Description

Format Event

### Usage

```
formatEvent(event, eventNumber = 1, precisionLevel = NULL)
```

### Arguments

| | |
|---|---|
| `event` | one row of event data frame as returned by `hsEvents` |
| `eventNumber` | number of the event |
| `precisionLevel` | 1 (less precise) or 2 (more precise) |

## `formatEventStatistics` *Format Event Statistics*

### Description

Format Event Statistics

### Usage

```
formatEventStatistics(eventStatistics, precisionLevel = NULL)
```

### Arguments

`eventStatistics`
date frame with columns *V.m3*, *H.max*, *Q.max*, *Q.raw.na*,

`precisionLevel` number of digits after decimal point

## `formatEventStatisticsTable` *Format Event Statistics Table*

### Description

Format Event Statistics Table

### Usage

```
formatEventStatisticsTable(eventStatisticsExtended,
  precisionLevel = NULL)
```

### Arguments

`eventStatisticsExtended`
list of event properties

`precisionLevel` 1 (less precise) or 2 (more precise)

## `formatSettings` *Format Settings*

### Description

Format Settings

### Usage

```
formatSettings(settings, settingNames = names(settings),
  do.stop = FALSE)
```

### Arguments

| | |
|---|---|
| `settings` | list of settings |
| `settingNames` | names of the settings, by default: `names(settings)` |
| `do.stop` | passed to `kwb.monitoring:::get_H_threshold`, `kwb.monitoring:::get_Q_threshold`, `kwb.monitoring:::get_V_threshold` |

## `getAllSamplerEvents` *Get All Sampler Events*

### Description

Get All Sampler Events

### Usage

```
getAllSamplerEvents(rootDirectory, dictionaryFile, stations,
  FUN.readSamplerFile, bottlesToConsider = NA,
  sampleEventSeparationTime = 3600, method = "centre",
  signalWidth = 60)
```

### Arguments

| | |
|---|---|
| `rootDirectory` | passed to `availableAutoSamplerFiles2` |
| `dictionaryFile` | passed to `availableAutoSamplerFiles2` |
| `stations` | vector of monitoring station names, each of which is passed to `availableAutoSamplerFiles2` |
| `FUN.readSamplerFile` | passed to `getAllSamplerEventsFromFiles` |
| `bottlesToConsider` | passed to `getAllSamplerEventsFromFiles` |
| `sampleEventSeparationTime` | passed to `getAllSamplerEventsFromFiles` |
| `method` | passed to `getAllSamplerEventsFromFiles` |
| `signalWidth` | passed to `getAllSamplerEventsFromFiles` |

---

`getAllTypesOfEvents` *Get All Types of Events*

---

## Description

Get All Types of Events

## Usage

```
getAllTypesOfEvents(hydraulicData = NULL, settings, FUN.readSamplerFile)
```

## Arguments

| | |
|---|---|
| `hydraulicData` | data frame containing hydraulic data that is passed to `getHydraulicEvents` |
| `settings` | list of settings, passed to `getSamplerEvents` and containing element `tstep.fill.s` |
| `FUN.readSamplerFile` | e.g. kwb.ogre::readOgreSamplerFileByName or kwb.dswt::readDswtSamplerFileByName |

---

`getFunctionValueOrDefault2`
*Get Function Value or Default 2*

---

## Description

Get Function Value or Default 2

## Usage

```
getFunctionValueOrDefault2(values, FUN, default, timestamps = NULL,
  columnName = "")
```

## Arguments

| | |
|---|---|
| `values` | passed to `getFunctionValueOrDefault` |
| `FUN` | passed to `getFunctionValueOrDefault` |
| `default` | passed to `getFunctionValueOrDefault` |
| `timestamps` | vector of timestamps (used in warning message) |
| `columnName` | column name (used in warning message) |

## getHydraulicEvents *Get Hydraulic Events*

### Description

Get Hydraulic Events

### Usage

```
getHydraulicEvents(hydraulicData, settings,
  eventSettings = settings$event[[settings$station]], columnQ = "Q",
  columnH = "H")
```

### Arguments

| | |
|---|---|
| `hydraulicData` | data frame with columns ... |
| `settings` | settings as returned by ... |
| `eventSettings` | default: `settings$event[[settings$station]]` |
| `columnQ` | name of column containing water flows. Default: "Q" |
| `columnH` | name of column containing water levels. Default: "H" |

## getIndicesWithinEvents *Get Indices Within Events*

### Description

Get Indices Within Events

### Usage

```
getIndicesWithinEvents(hydraulicData, eventSettings = NULL,
  thresholds = c(H = NA, Q = NA), columns = c(H = "H", Q = "Q"))
```

### Arguments

| | |
|---|---|
| `hydraulicData` | data frame with a columns *DateTime* and two columns named as given in *columnH*, *columnQ* |
| `eventSettings` | data frame with columns *tBeg*, *tEnd* (begin and end of period in which thresholds are valid), *Hthreshold*, *Qthreshold* |
| `thresholds` | default H and Q thresholds to be applied for time intervals for which no special thresholds are defined |
| `columns` | optional. Named vector of character with column names for `H` and `Q` |

## getMergedEvents *Merge hydraulic events and sampler events*

### Description

Merge hydraulic events and sampler events

### Usage

```
getMergedEvents(hydraulicEvents, samplerEvents, signalWidth)
```

### Arguments

| | |
|---|---|
| hydraulicEvents | data frame containing information on hydraulic events |
| samplerEvents | data frame containing information on sampler events |
| signalWidth | passed to toEvents |

## getModelFromFile *Read Regression Model from Text File*

### Description

Read Regression Model from Text File

### Usage

```
getModelFromFile(modelFile, warn = TRUE)
```

### Arguments

| | |
|---|---|
| modelFile | path to .RData file containing the model |
| warn | if TRUE (the default), a warning is given if the model file does not exist |

## getOrCreatePath *Get or Create Path*

### Description

Get or Create Path

### Usage

```
getOrCreatePath(variableName, dictionary = settings$dictionary,
  settings = NULL, create.dir = FALSE, stop.on.no.resolving = TRUE,
  dbg = FALSE, ...)
```

### Arguments

| | |
|---|---|
| `variableName` | key to be looked up in *dictionary*, resolving to a file path |
| `dictionary` | dictionary (list of key/value pairs) in which *variableName* is looked up |
| `settings` | default: NULL |
| `create.dir` | if TRUE, the directory is created |
| `stop.on.no.resolving` | if TRUE and *variableName* could not be resolved the program stops |
| `dbg` | if TRUE, debug messages are shown |
| `...` | arguments passed to `resolve` |

## getPredictionOfQ *Get Prediction of Q*

### Description

Get Prediction of Q

### Usage

```
getPredictionOfQ(hydraulicData, regressionModels = NULL, modelDir,
  columns = c(DateTime = "DateTime", H = "H", Q.raw = "Q.raw"),
  only.if.na = TRUE)
```

### Arguments

`hydraulicData` data frame with columns as named in `columns`

`regressionModels`
: data frame with character columns *from*, *to* (or POSIXct columns *tBeg* and *tEnd*) and character columns *modelFile* determining the time intervals to which the different correlation models are assigned.

`modelDir` full path to directory where the model files (of the current station) are stored

`columns` names of columns containing Date and Time, water levels and water flows

`only.if.na` logical. Not used!

---

## `getSampleInformation` *Get Sample Information*

---

### Description

Get Sample Information

### Usage

```
getSampleInformation(dictionary = settings$dictionary,
  sampleEventSeparationTime = settings$sampleEventSeparationTime,
  sampleEventIndices = -1,
  bottlesToConsider = settings$bottlesToConsider,
  method = settings$sampleEventMethod, FUN.readSamplerFile,
  settings = NULL, signalWidth = 1)
```

### Arguments

`dictionary` default: settings$dictionary

`sampleEventSeparationTime`
: default: settings$sampleEventSeparationTime

`sampleEventIndices`
: default: -1

`bottlesToConsider`
: default: settings$bottlesToConsider

`method` one of the methods supported by `sampleDataToSamplingEvents`. default: settings$sampleEventMethod

`FUN.readSamplerFile`
: function to be used to read an auto sampler file

`settings` list of settings, as returned by `configure`

`signalWidth` signal width in seconds. Default: 1

### Value

list with elements *sampleTimes*, *samplingEvents*, *bottle events* (data frame with columns...)

## getSamplerEvents *Get Sampler Events*

### Description

Get sample event information from all available auto sampler files

### Usage

```
getSamplerEvents(settings, FUN.readSamplerFile, warn = TRUE)
```

### Arguments

| | |
|---|---|
| `settings` | passed to `getSampleInformation` |
| `FUN.readSamplerFile` | passed to `getSampleInformation` |
| `warn` | passed to `availableAutoSamplerFiles` |

## getStatisticsByDay *Get Statistics by Day*

### Description

Get Statistics by Day

### Usage

```
getStatisticsByDay(dataFrame)
```

### Arguments

| | |
|---|---|
| `dataFrame` | data frame with columns *parName*, *parVal*, *day* |

### Value

data frame with additional columns *info* ("day: H max = ... cm, Q max = ... L/s")

## `getStatisticsByEvent` *Get Statistics by Event*

### Description

Get Statistics by Event

### Usage

```
getStatisticsByEvent(hydraulicData, events)
```

### Arguments

| | |
|---|---|
| `hydraulicData` | data frame containing hydraulic data |
| `events` | data frame containing information on events, passed to `hsEventNumber` |

## `H_above_threshold` *H Above Threshold*

### Description

Vector of TRUE/FALSE with TRUE at positions where H is above the threshold

### Usage

```
H_above_threshold(dat.raw, settings)
```

### Arguments

| | |
|---|---|
| `dat.raw` | data frame with column *H* |
| `settings` | list as returned by `configure` with list element *Hthresholds* |

## `indicesInIntervals` *Indices of Rows Belonging to Time Intervals*

### Description

Indices of Rows Belonging to Time Intervals

### Usage

```
indicesInIntervals(timestamps, intervals)
```

### Arguments

| | |
|---|---|
| `timestamps` | vector of POSIXct timestamps |
| `intervals` | vector of Interval objects as returned by lubridate::interval |

### Value

indices of elements in *timestamps* that belong to the given time *intervals*

## `mergeParallelRainEventStat` *Merge Parallel Rain Event Statistics*

### Description

Merge Parallel Rain Event Statistics

### Usage

```
mergeParallelRainEventStat(hydraulicEvents, rainEvents, rainData,
  seriesName, offset = 0, plot.merged.event.info = TRUE)
```

### Arguments

| | |
|---|---|
| `hydraulicEvents` | data frame containing information on hydraulic events, with columns `eventNumber`, `tBeg` |
| `rainEvents` | list of data frames containing information on rain events, with one list entry per rain gauge of which one is named as given in `seriesName` |
| `rainData` | passed to `getEventStatistics` |
| `seriesName` | name of rain gauge |
| `offset` | time in seconds by which tBeg is shifted backwards |
| `plot.merged.event.info` | if `TRUE` (default), `plotMergedEventInfoForValidation` is called |

## pathDictionary *Path Dictionary*

### Description

Read dictionary from file and set RAW_DIR and STATION

### Usage

```
pathDictionary(dictionaryFile, RAW_DIR = settings$rawdir,
  STATION = settings$station, settings = NULL)
```

### Arguments

| | |
|---|---|
| `dictionaryFile` | full path to file defining a dictionary and being read with `readDictionary` |
| `RAW_DIR` | value for placeholder of the same name in the dictionary |
| `STATION` | value for placeholder of the same name in the dictionary |
| `settings` | optional. List with elements `rawdir`, `station` from which to take values to be used for `RAW_DIR`, `STATION` |

## plotEventDistribution *Plot Event Distribution*

### Description

Plot Event Distribution

### Usage

```
plotEventDistribution(eventsAndStat, settings)
```

### Arguments

| | |
|---|---|
| `eventsAndStat` | data frame containing event information |
| `settings` | list of settings, at least with elements `station`, `precisionLevel` |

## `plotEventOverview` *Gantt-Plots for Event Overview*

### Description

Gantt-Plots for Event Overview

### Usage

```
plotEventOverview(events, settings, dbg = FALSE)
```

### Arguments

| | |
|---|---|
| `events` | list with elements *hydraulic*, *sample*, *merged* each of which is a data frame with columns *tBeg*, *tEnd* |
| `settings` | list with elements *evtSepTime*, *station*, *Hthresholds* |
| `dbg` | logical. If `TRUE`, the x axis coordinates are printed |

## `plotOverview` *Plot Overview*

### Description

Plot Overview

### Usage

```
plotOverview(dat, station, Qmax = NULL, Hmax = NULL)
```

### Arguments

| | |
|---|---|
| `dat` | data frame containing the data to be plotted |
| `station` | name of monitoring station, used in plot title |
| `Qmax` | passed to `kwb.monitoring:::plotOverview_byDay` |
| `Hmax` | passed to `kwb.monitoring:::plotOverview_byDay` |

## plotSampleInformation *Plot Sample Information*

### Description

plotSampleInformation. TODO: simplify interface, e.g. plotSampleInformation(getSampleInformation(getSampleFiles()[1])

### Usage

```
plotSampleInformation(sampleInformation, add = FALSE,
  xlim = kwb.datetime::toUTC(range(c(sampleInformation$samplingEvents$tBeg,
  sampleInformation$samplingEvents$tEnd))), ylim = c(-2, 7), main = NA,
  cex.legend = 0.6, density = 0, plotSampleIntervals = TRUE,
  maxSamplesOk = NULL, plotSamplingPoints = plotSampleIntervals)
```

### Arguments

| | |
|---|---|
| `sampleInformation` | list with elements *samplingEvents*, *bottleEvents* |
| `add` | passed to `ganttPlotEvents` |
| `xlim` | passed to `ganttPlotEvents` |
| `ylim` | passed to `ganttPlotEvents` |
| `main` | plot title |
| `cex.legend` | passed to `addSampleTimesToPlot` |
| `density` | passed to `ganttPlotEvents` |
| `plotSampleIntervals` | logical. If `TRUE` `ganttPlotEvents` is called for `sampleInformation$samplingEvents` |
| `maxSamplesOk` | maximum number of valid samples. This value is used to calculate an y coordinate |
| `plotSamplingPoints` | = logical. If `TRUE` `addSampleTimesToPlot` is called |

## plotTotalDischargeVersusRainProperties *Plot Total Discharge Versus Rain Properties*

### Description

Plot Total Discharge Versus Rain Properties

### Usage

```
plotTotalDischargeVersusRainProperties(hydraulicEvents, stations, main,
  settings, statistics = "sum", to.pdf = FALSE)
```

### Arguments

hydraulicEvents
data frame containing information on hydraulic events

stations vector of monitoring station names

main plot title

settings list of settings

statistics one of c("sum", "mean", "max")

to.pdf logial. If TRUE, the output goes into a PDF file

## plot_hydraulic_event *Plot Hydraulic Event*

### Description

Plot Hydraulic Event

### Usage

```
plot_hydraulic_event(hydraulicData, settings, eventAndStat,
  sampleInformation = NULL, ylim.Q = NULL, rainData = NULL,
  gauges = NULL, left = 0.1, right = 0.1, innerMargins.HQ = c(0.2,
  left, 0.1, right), innerMargins.rain = c(0, left, 0.2, right),
  dbg = FALSE)
```

### Arguments

hydraulicData data frame with columns *DateTime*, *H*, *Q*, *Q.raw*, *Q.raw.signal*, *Q.interpol*

settings list of settings containing additional information and passed to other functions

eventAndStat passed to kwb.monitoring:::.plotRainData

sampleInformation
passed to kwb.monitoring:::.partialPlot_H

ylim.Q passed to kwb.monitoring:::.partialPlot_Q

rainData optional. Data frame containing rain data

gauges passed to kwb.monitoring:::.plotRainData if rainData is given

left fraction of event length by which xlim is extended to the left

right fraction of event length by which xlim is extended to the right

innerMargins.HQ
"inner margins" of H and Q plots. Default: c(0.2, left, 0.1, right)

innerMargins.rain
"inner margins" of rain plots. Default: c(0, left, 0.2, right)

dbg logical. If TRUE eventAndStat is printed.

## `plot_hydraulic_events` *Plot Hydraulic Events*

### Description

Plot Hydraulic Events

### Usage

```
plot_hydraulic_events(hydraulicData, settings, eventsAndStat,
  to.pdf = FALSE, rainData = NULL, gauges = NULL, landscape = TRUE,
  plot.event.overview = TRUE,
  FUN.plot_hydraulic_event = kwb.monitoring::plot_hydraulic_event, ...)
```

### Arguments

`hydraulicData` data frame with column `DateTime`, passed to the function given in `FUN.plot_hydraulic_event`

`settings` list of settings (e.g. `dictionary`), passed to `plotEventDistribution` if `plot.event.overview` is `TRUE`

`eventsAndStat` data frame containing event information

`to.pdf` if `TRUE`, graphical output goes to a temporary pdf file

`rainData` passed to the function given in *FUN.plot_hydraulic_event*

`gauges` passed to the function given in *FUN.plot_hydraulic_event*

`landscape` orientation of pages in PDF file if `to.pdf` is `TRUE`

`plot.event.overview`
if `TRUE`, `plotEventDistribution` is called

`FUN.plot_hydraulic_event`
function to be called to plot one event

`...` arguments passed to the function given in *FUN.plot_hydraulic_event*

## `plot_H_columns` *Plot H Columns*

### Description

Plot H Columns

### Usage

```
plot_H_columns(hydraulicData, h.threshold = 0,
  time.dependent.thresholds = NULL, xlim = NULL, ylim = NULL,
  innerMargins = default_inner_margins())
```

### Arguments

hydraulicData data frame with columns *DateTime*, *H*, *H.interpol*

h.threshold H threshold at which a horizontal line is to be drawn (default: 0)

time.dependent.thresholds
passed to kwb.monitoring:::draw_thresholds_if_applicable

xlim passed to plot_variable

ylim passed to plot_variable

innerMargins passed to plot_variable

## plot_sampled_event *Plot Sampled Event*

### Description

Plot Sampled Event

### Usage

```
plot_sampled_event(hydraulicData, settings, sampleInformation = NULL,
  mergedEventAndStat, volumeCompositeSample = NULL, to.pdf = FALSE,
  interpolate = TRUE, ...)
```

### Arguments

hydraulicData data frame with columns *DateTime*, *H*, *Q*, *Q.raw*, *Q.raw.signal*, *Q.interpol*

settings list with elements dictionary, station, precisionLevel,bottlesToDiscard, bottlesToConsider and being passed to other functions

sampleInformation
passed to kwb.monitoring:::.partialPlot_Q and kwb.monitoring:::.partialPlot_H

mergedEventAndStat
list with event information such as tBeg, tEnd, V.m3 and passed to kwb.monitoring:::formatEventRel

volumeCompositeSample
list with elements bottle, V.bottle.mL, V, passed to kwb.monitoring:::formatVolumeCompositeSampl

to.pdf logical. If TRUE, output goes to a PDF file

interpolate passed to kwb.monitoring:::.partialPlot_Q

... further arguments given to

## printSampleInformation *Print Sample Information*

### Description

Print Sample Information

### Usage

```
printSampleInformation(sampleInformation)
```

### Arguments

| | |
|---|---|
| `sampleInformation` | list with elements `sampleData`, `samplingEvents,bottleEvents` |

## rainGaugesNearStation *Rain Gauges Near to Monitoring Sites*

### Description

Rain Gauges Near to Monitoring Sites

### Usage

```
rainGaugesNearStation(station = NULL)
```

### Arguments

| | |
|---|---|
| `station` | name of monitoring station in KWB project OGRE or DSWT |

### Value

list (one list element per monitoring site) of character vectors representing rain gauge names

## `readAndJoinSamplerFiles` *Read Multiple Auto-Sampler Files*

### Description

Read Multiple Auto-Sampler Files

### Usage

```
readAndJoinSamplerFiles(samplerFiles, FUN.readSamplerFile,
  bottlesToConsider = NA, ...)
```

### Arguments

| | |
|---|---|
| `samplerFiles` | vector of paths to sampler files |
| `FUN.readSamplerFile` | function to be used for reading the sampler file |
| `bottlesToConsider` | vector of bottle numbers to consider. Defaults to NA meaning that information on all bottles are to be returened. |
| `...` | further arguments passed to `FUN.readSamplerFile` |

### Value

data frame with columns *file*, *myDateTime*, *sample*, *bottle*, *volume*, *unit*, *result*

## `removeDuplicates` *Remove Duplicates*

### Description

Remove Duplicates

### Usage

```
removeDuplicates(hydraulicData, timeColumnName = "DateTime",
  keep.first = TRUE)
```

### Arguments

| | |
|---|---|
| `hydraulicData` | data frame with date and time column as named in `timeColumnName` |
| `timeColumnName` | name of date and time column in `hydraulicData` |
| `keep.first` | logical. If `TRUE` (`FALSE` is not implemented!) only the first rows of sets of rows with duplicated time stamps are kept. |

## `removeIntervals` *Remove Intervals*

### Description

Remove Intervals

### Usage

```
removeIntervals(dataFrame, intervals,
  dateTimeColumn = names(kwb.utils::posixColumnAtPosition(dataFrame))[1])
```

### Arguments

| | |
|---|---|
| `dataFrame` | data frame with a column as named in `dateTimeColumn` |
| `intervals` | vector of Interval objects as returned by lubridate::interval, passed to `indicesInIntervals` |
| `dateTimeColumn` | name of date and time column in `dataFrame` |

## `removeZoomFromHistory` *Remove Zoom from History*

### Description

Remove Zoom from History

### Usage

```
removeZoomFromHistory(zoomHistory)
```

### Arguments

| | |
|---|---|
| `zoomHistory` | list of (i, j) pairs, storing the history of zoom settings |

## sampleDataToSamplingEvents

*Sample Data to Sampling Events*

### Description

Sample Data to Sampling Events

### Usage

```
sampleDataToSamplingEvents(sampleData, method = "centre",
  default.interval.width.s = 600, signalWidth = 1)
```

### Arguments

| | |
|---|---|
| `sampleData` | sample data as returned by `readAndJoinSamplerFiles` |
| `method` | one of c("left", "right", "centre") |
| `default.interval.width.s` | default interval width in seconds |
| `signalWidth` | interval length in seconds that a sampling action is assumed to represent. It is only used to calculate the duration D of the time interval between two samplings at t1 and t2: D = t2 - t1 + signalWidth |

### Value

data frame with columns *tBeg*, *tEnd*, ...

## sampleLogFileToSampleName

*Sample Log File to Sample Name*

### Description

Sample Log File to Sample Name

### Usage

```
sampleLogFileToSampleName(sampleFile)
```

### Arguments

| | |
|---|---|
| `sampleFile` | full path to auto sampler file |

## samplingEventsToBottleEvents *Sampling Events to Bottle Events*

### Description

Sampling Events to Bottle Events

### Usage

```
samplingEventsToBottleEvents(samplingEvents, signalWidth = 1)
```

### Arguments

| | |
|---|---|
| `samplingEvents` | data frame with columns *samplerFile*, *bottle*, *tBeg*, *tEnd* |
| `signalWidth` | passed to `toEvents` |

### Value

data frame with columns *tBeg*, *tEnd*, *dur*, *bottle*, *samplesOk*

## saveRegressionModel *Save Regression Model*

### Description

Save Regression Model

### Usage

```
saveRegressionModel(regressionModel, settings = NULL,
  dictionary = kwb.utils::selectElements(settings, "dictionary"),
  sep = kwb.utils::selectElements(settings, "outsep"),
  dec = kwb.utils::selectElements(settings, "outdec"))
```

### Arguments

| | |
|---|---|
| `regressionModel` | model object to be stored |
| `settings` | optional. List from which to take the arguments if not given |
| `dictionary` | dictionary (list) containing entries "REGRESSION_MODEL_TXT" and "REGRESSION_COEFF_TXT" |
| `sep` | column separator in created file |
| `dec` | decimal character in created file |

## `saveSampleInformation` *Save Sample Information*

### Description

Save Sample Information

### Usage

```
saveSampleInformation(sampleInformation, settings, sampleFile)
```

### Arguments

`sampleInformation`
: list with elements `samplingEvents,bottleEvents`

`settings`
: list with elements `dictionary`, `outsep`, `outdec`

`sampleFile`
: name of auto sampler file, passed to `sampleLogFileToSampleName`

## `selectIntervalsForCorrelation` *Select Intervals for Correlation*

### Description

Select Intervals for Correlation

### Usage

```
selectIntervalsForCorrelation(dat.all, settings, h.threshold.max = 0.4)
```

### Arguments

`dat.all`
: data frame containing all relevant data passed to `kwb.monitoring:::H_above_threshold`

`settings`
: list of settings with elements `Hthresholds`, `"station"` and being passed to `kwb.monitoring:::H_above_threshold`

`h.threshold.max`
: maximum value for the slider

## `setPausesOfMergedEvents` *Update Pauses of Merged Events*

### Description

Update Pauses of Merged Events

### Usage

```
setPausesOfMergedEvents(hydraulicEvents, mergedEvents, dbg = FALSE)
```

### Arguments

`hydraulicEvents` data frame containing information on hydraulic events

`mergedEvents` data frame containing information on "merged" events

`dbg` logical. If `TRUE`, debug messages are shown

### Value

*mergedEvents* with updated pauses *pBefore* and *pAfter*

## `showOverview` *Show Overview*

### Description

Show Overview

### Usage

```
showOverview(dat, settings, Qmax = NULL, Hmax = NULL, to.pdf = FALSE,
  save.pdf = FALSE)
```

### Arguments

`dat` data frame with columns `DateTime`

`settings` list with elements `dictionary`, `station`

`Qmax` passed to `plotOverview`

`Hmax` passed to `plotOverview`

`to.pdf` if `TRUE`, graphical output goes to a temporary pdf file

`save.pdf` if `TRUE` PDF output is stored to a file stored and named according to `dictionary$OVERVIEW_HQ_DATA_PDF`

## updateZoomHistory *Update Zoom History*

### Description

Update Zoom History

### Usage

```
updateZoomHistory(action, zoomHistory, i, j)
```

### Arguments

| | |
|---|---|
| action | one of c("zoom.in", "zoom.out") |
| zoomHistory | list of (i, j) pairs, storing the history of zoom settings |
| i | index in range of slider "left" |
| j | index in range of slider "right" |

## validateAndFillHydraulicData *Validate and Fill Hydraulic Data*

### Description

1. remove rows with duplicate timestamps 2. fill gaps within hydraulic events given by time intervals exceeding H threshold

### Usage

```
validateAndFillHydraulicData(hydraulicData,
  tstep.fill.s = selectElements(settings, "tstep.fill.s"),
  replaceMissingQMethod = selectElements(settings,
  "replaceMissingQMethod"),
  regressionModels = selectElements(selectElements(settings,
  "regression"), "models")[[selectElements(settings, "station")]],
  regressionUsage = selectElements(selectElements(settings,
  "regression"), "usage")[[selectElements(settings, "station")]],
  hydraulicEvents = NULL, additionalColumns = NULL,
  modelDir = getOrCreatePath("REGRESSION_DIR", selectElements(settings,
  "dictionary")), settings = NULL)
```

## Arguments

| | |
|---|---|
| `hydraulicData` | data frame with column *DateTime*, ... |
| `tstep.fill.s` | target time step in seconds. Time gaps in *hydraulicData* will be filled with interpolated values |
| `replaceMissingQMethod` | one of c("interpolate", "predict"). "interpolate": linear interpolation "predict": prediction from water levels using a saved square regression |
| `regressionModels` | data frame with character columns *from*, *to* (or POSIXct columns *tBeg* and *tEnd*) and *modelFile* determining the time intervals to which the different correlation models are assigned. |
| `regressionUsage` | data frame with character columns *from*, *to* (or POSIXct columns *tBeg* and *tEnd*) defining the first and last timestamp of the time intervals in which the correlation is to be used |
| `hydraulicEvents` | hydraulic events |
| `additionalColumns` | columns additional to "DateTime", "H" and "Q" to be selected from *hydraulicData* |
| `modelDir` | full path to the directory where the model files (of the current station) are stored) |
| `settings` | settings as returned by `configure`. Will be used to lookup function parameters for which no values have been given (see defaults) |

---

# `whichAboveThresholds` *In Which Rows Are Thresholds Exceeded?*

---

## Description

Get indices of rows in *hydraulicData* in which H or Q thresholds are exceeded

## Usage

```
whichAboveThresholds(hydraulicData,
  indices = seq_len(nrow(hydraulicData)), thresholds = c(H = NA, Q =
  NA), columns = c(H = "H", Q = "Q"))
```

## Arguments

| | |
|---|---|
| `hydraulicData` | data frame with columns as named in `columns` |
| `indices` | vector of indices of preselected rows from which to exclude those in which the thresholds are exceeded |
| `thresholds` | vector of thresholds for `H` and `Q` values, respectively |
| `columns` | vector of names containing `H` and `Q` values, respectively |

## writeCsvToPathFromDictionary

*Write CSV to Path From Dictionary*

### Description

Write CSV to Path From Dictionary

### Usage

```
writeCsvToPathFromDictionary(dataFrame, key, settings,
  open.directory = TRUE, ...)
```

### Arguments

| | |
|---|---|
| `dataFrame` | data frame containing data to save |
| `key` | key in *settings$dictionary* to be resolved to file path |
| `settings` | list of settings with elements *dictionary*, *outsep*, *outdec* |
| `open.directory` | if TRUE (default), the directory in which the file is created is opened in the Windows Explorer after the file has been written. |
| `...` | arguments passed to `getOrCreatePath` |

## writeDataAndOpenDirectory

*Write Data and Open Directory*

### Description

Write Data and Open Directory

### Usage

```
writeDataAndOpenDirectory(dataFrame, filePath, settings)
```

### Arguments

| | |
|---|---|
| `dataFrame` | data frame containing data to save |
| `filePath` | base name of file ("_<moniPoint>_<sampleName>.csv" will be appended) |
| `settings` | list of settings, as returned by `configure` |

# Index

.plotRainData, 3
.toIntervalList, 4

addFakeEntriesForDaysWithoutData, 4
addSampleTimesToPlot, 5, *27*
addStatisticsToEvents, 5
addSumRow, 6
addZoomToHistory, 6
appendColumn_samplesOk, 7
appendInterpolColumns, 7
apply_H_threshold, 8
availableAutoSamplerFiles, 8, *22*
availableAutoSamplerFiles2, 9

bottleEventsToSamplerEvents, 9

calculateVolumeCompositeSample, 10
configure, *8*, *10*, 10, *21*, *23*, *39*, *40*
createDummyEventThresholdFiles, 12

dateToDateStringInPath, 13

filterForRelevantBottles, 13
filterSampleEventsForFilename, 14
formatEvent, 14
formatEventStatistics, 15
formatEventStatisticsTable, 15
formatSettings, 16

ganttPlotEvents, *27*
getAllSamplerEvents, 16
getAllTypesOfEvents, 17
getEventStatistics, *24*
getFunctionValueOrDefault, *17*
getFunctionValueOrDefault2, 17
getHydraulicEvents, *17*, 18
getIndicesWithinEvents, 18
getMergedEvents, 19
getModelFromFile, 19
getOrCreatePath, 20, *40*
getPredictionOfQ, 20
getSampleInformation, 21, *22*
getSamplerEvents, *17*, 22
getStatisticsByDay, 22
getStatisticsByEvent, *5*, 23

H_above_threshold, 23
hsDateStr, *13*
hsEventNumber, *23*
hsEvents, *14*

indicesInIntervals, 24, *33*

mergeParallelRainEventStat, 24

pathDictionary, *9*, 25
plot_H_columns, 29
plot_hydraulic_event, 28
plot_hydraulic_events, 29
plot_sampled_event, 30
plot_variable, *30*
plotEventDistribution, 25, *29*
plotEventOverview, 26
plotMergedEventInfoForValidation, *24*
plotOverview, 26, *37*
plotRain, *3*
plotSampleInformation, 27
plotTotalDischargeVersusRainProperties, 27
printSampleInformation, 31

rainGaugesNearStation, 31
readAndJoinSamplerFiles, 32, *34*
readDictionary, *25*
removeDuplicates, 32
removeIntervals, 33
removeZoomFromHistory, 33
resolve, *20*

sampleDataToSamplingEvents, *21*, 34
sampleLogFileToSampleName, 34, *36*
samplingEventsToBottleEvents, 35

saveRegressionModel, 35
saveSampleInformation, 36
selectIntervalsForCorrelation, 36
setPausesOfMergedEvents, 37
showOverview, 37

toEvents, *9*, *19*, *35*

updateZoomHistory, 38

validateAndFillHydraulicData, 38

whichAboveThresholds, 39
writeCsvToPathFromDictionary, 40
writeDataAndOpenDirectory, 40