# Package: aquanes.report (via r-universe)

October 12, 2024

**Type** Package

**Title** Automated Reporting Tool for Water Suppliers

**Version** 0.5.0

**Description** Collects, aggregates and visualises operational and analytical data from water suppliers (including a standardised reporting document).

**Depends** R (>= 3.3)

**Imports** plyr (>= 1.8.4), RMySQL (>= 0.10.9), dplyr (>= 0.7.1), dbplyr (>= 1.0.0), readxl (>= 1.0.0), readr (>= 1.1.0), tidyr (>= 0.6.1), ggplot2 (>= 2.2.1), ggforce (>= 0.1.1), xml2 (>= 1.1.0), rvest (>= 0.3.2), shiny (>= 1.0.0), shinythemes (>= 1.1.1), dygraphs (>= 1.1.1.4), digest (>= 0.6.11), leaflet (>= 1.1.0), rmarkdown (>= 1.3), xts (>= 0.9-7), lubridate (>= 1.6.0), fasttime (>= 1.0-2), data.table (>= 1.10.4), stringr (>= 1.2.0), fst (>= 0.8), magrittr (>= 1.5), janitor (>= 0.3.0)

**Suggests** devtools (>= 1.13.0), knitr, covr, testthat

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 6.1.1

**URL** `https://github.com/kwb-r/aquanes.report`

**BugReports** `https://github.com/kwb-r/aquanes.report/issues`

**VignetteBuilder** knitr

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

**RemoteUrl** https://github.com/KWB-R/aquanes.report

**RemoteRef** HEAD

**RemoteSha** 7df99ab71b6fbdf05a7b64024d8f7555361e3903

# Contents

add_label . . . 3
add_parameter_metadata . . . 3
add_site_metadata . . . 4
aggregate_export_fst_berlin_s . . . 5
aggregate_export_fst_berlin_t . . . 5
calculate_operational_parameters . . . 6
calculate_operational_parameters_berlin_s . . . 7
calculate_operational_parameters_berlin_t . . . 8
calenderweek_from_dates . . . 9
change_timezone . . . 9
check_thresholds . . . 10
create_monthly_selection . . . 10
create_report_batch . . . 11
create_wedeco_metafile . . . 11
dygraph_add_limits . . . 12
get_monthly_data_from_calendarweeks . . . 12
get_monthly_periods . . . 13
get_rawfilespaths_for_month . . . 13
get_thresholds . . . 14
get_valid_timezones . . . 14
group_datetime . . . 15
import_analytics_basel . . . 15
import_analytics_meta_basel . . . 16
import_data_basel . . . 17
import_data_berlin_s . . . 18
import_data_berlin_t . . . 18
import_data_haridwar . . . 19
import_lab_data_berlin_t . . . 20
import_operation . . . 20
import_operation_basel . . . 21
import_operation_meta_basel . . . 21
import_sheets . . . 22
load_fst_data . . . 23
merge_and_export_fst . . . 24
multiSubstitute . . . 24
plot_analytics . . . 25
plot_calculated_operational_timeseries . . . 25
read_fst . . . 26
read_pentair_data . . . 26
read_wedeco_data . . . 27
remove_duplicates . . . 27
report_config_template . . . 28
report_config_to_txt . . . 28
report_txt_to_config . . . 29
run_app . . . 30
set_timezone . . . 30

**Index** 32

---

`add_label` *Helper function: add label ("SiteName_ParaName_Unit_Method")*

---

## Description

Helper function: add label ("SiteName_ParaName_Unit_Method")

## Usage

```
add_label(df, col_sitename = "SiteName",
  col_parametername = "ParameterName",
  col_parameterunit = "ParameterUnit", col_method = "Method_Org")
```

## Arguments

| | |
|---|---|
| `df` | data frame containing at least a columns "SiteName", "ParameterName", "ParameterUnit" and optionally "Method_Org" (if not existent no "Method_Org" will be available!) |
| `col_sitename` | column in df containing site name (default: "SiteName") |
| `col_parametername` | column in df containing parameter name (default: "ParameterName") |
| `col_parameterunit` | column in df containing parameter unit (default: "ParameterUnit") |
| `col_method` | column in df containing method code (default: "Method_Org") |

## Value

returns input data frame with added column "SiteName_ParaName_Unit_Method"

---

`add_parameter_metadata` *Helper function: add parameter metadata*

---

## Description

Helper function: add parameter metadata

## Usage

```
add_parameter_metadata(df,
 meta_parameter_path = system.file("shiny/basel/data/metadata/meta_parameter.csv",
  package = "aquanes.report"))
```

### Arguments

| | |
|---|---|
| `df` | data frame containing at least a column "ParameterCode" |
| `meta_parameter_path` | Define path of "meta_parameter.csv" to be imported (default: system.file("shiny/basel/data/metadata/meta package = "aquanes.report")) |

### Value

returns input data frame with joined metadata (parameter codes/ methods not included in meta_parameter file will not be imported!!!!)

---

## `add_site_metadata` *Helper function: add site metadata*

---

### Description

Helper function: add site metadata

### Usage

```
add_site_metadata(df, df_col_sitecode = "SiteCode",
  meta_site_path = system.file("shiny/basel/data/metadata/meta_site.csv",
  package = "aquanes.report"))
```

### Arguments

| | |
|---|---|
| `df` | data frame containing at least a column "SiteCode" |
| `df_col_sitecode` | column in df containing site code (default: "SiteCode") |
| `meta_site_path` | Define path of "meta_site.csv" to be imported (default: system.file("shiny/basel/data/metadata/meta_site.c package = "aquanes.report")) |

### Value

returns input data frame with joined metadata

## aggregate_export_fst_berlin_s

*Berlin-Schoenerlinde: aggregate and export to fst*

### Description

Berlin-Schoenerlinde: aggregate and export to fst

### Usage

```
aggregate_export_fst_berlin_s(year_month_start = "2017-04",
  year_month_end = format(Sys.Date(), "%Y-%m"), compression = 100)
```

### Arguments

| | |
|---|---|
| `year_month_start` | start year month (default: '2017-04') |
| `year_month_end` | end year month (default: current month) |
| `compression` | (default: 100) |

### Value

exports data for each month into subfolder: /data/fst/year-month

## aggregate_export_fst_berlin_t

*Berlin-Tiefwerder: aggregate and export to fst*

### Description

Berlin-Tiefwerder: aggregate and export to fst

### Usage

```
aggregate_export_fst_berlin_t(year_month_start = "2017-06",
  year_month_end = format(Sys.Date(), "%Y-%m"), compression = 100)
```

### Arguments

| | |
|---|---|
| `year_month_start` | start year month (default: '2017-06') |
| `year_month_end` | end year month (default: current month) |
| `compression` | (default: 100) |

### Value

exports data for each month into subfolder: /data/fst/year-month `calculate_operational_parameters`

*Calculate operational parameters*

## Description

Calculate operational parameters

## Usage

```
calculate_operational_parameters(df, calc_list = list(Redox_Out =
  "(Redox_Out1+Redox_Out2)/2", Redox_Diff = "Redox_Out - Redox_In",
  Power_pump = "Up*Ip", Power_cell = "Uz*Iz", Pump_WhPerCbm =
  "Power_pump/(Flux/1000)", Cell_WhPerCbm = "Power_cell/(Flux/1000)"),
  calc_list_name = c("Mean redox potential in tank",
  "Difference (outflow - inflow) of redox potential",
  "Power demand of pump", "Power demand of cell",
  "Specific energy demand of pump", "Specific energy demand of cell"),
  calc_list_unit = c("mV", "mV", "W", "W", "Wh/m3", "Wh/m3"),
  calc_paras = c("Redox_Out1", "Redox_Out2", "Redox_In", "Flux", "Up",
  "Ip", "Uz", "Iz"))
```

## Arguments

| | |
|---|---|
| `df` | a data frame as retrieved by import_data_haridwar() |
| `calc_list` | list with calculation operations to be carried out (default: list(Redox_Out = "(Redox_Out1+Redox_Out2)/2", Redox_Diff = "Redox_Out - Redox_In", Power_pump = "Up*Ip", Power_cell = "Uz*Iz", Pump_WhPerCbm = "Power_pump/Flux/1000", Cell_WhPerCbm = "Power_cell/Flux/1000")) |
| `calc_list_name` | full names of parameters to be used for plotting for each calculation specified wit 'calc_list'. default: c('Tank water: Mean redox potential", 'Difference (outflow - inflow) of redox potential', 'Power demand of pump', 'Power demand of cell', 'Specific energy demand of pump', Specific energy demand of cell') |
| `calc_list_unit` | units of parameters to be used for plotting for each calculation specified wit 'calc_list'. default: c('mV', 'mV', 'Wh', 'Wh', 'Wh/m3', 'Wh/m3') |
| `calc_paras` | a vector with parameter codes used for performing calculations defined in 'calc_list' (default: c('Redox_Out1', 'Redox_Out2', 'Redox_In', 'Flux', 'Up', 'Ip', 'Uz', 'Iz')) |

## Value

dataframe with calculated operational parameters

## Examples

```
## Not run:
haridwar_raw_list <- import_data_haridwar()
myDat <- calculate_operational_parameters(df = haridwar_raw_list)
## End(Not run)
```

---

## `calculate_operational_parameters_berlin_s`

*Calculate operational parameters for Berlin-Schoenerlinde*

---

### Description

Calculate operational parameters for Berlin-Schoenerlinde

### Usage

```
calculate_operational_parameters_berlin_s(df, calc_list = list(deltaSAK =
  "(1-SCAN_SAK_Ablauf/SCAN_SAK_Zulauf)*100", Ozoneintrag =
  "(C_O3_Zugas - C_O3_Abgas)*Q_Gas/Q_Ozonanlage"),
  calc_list_name = c("delta SAK", "Ozoneintrag"),
  calc_list_unit = c("%", "mg-O3/L"),
  calc_paras = c("SCAN_SAK_Ablauf", "SCAN_SAK_Zulauf", "C_O3_Zugas",
  "C_O3_Abgas", "Q_Gas", "Q_Ozonanlage"))
```

### Arguments

| | |
|---|---|
| `df` | a data frame as retrieved by read_wedeco_data() |
| `calc_list` | list with calculation operations to be carried out (default: list(deltaSAK = "(1-SCAN_SAK_Ablauf/SCAN_SAK_Zulauf)*100", Ozoneintrag = "(C_O3_Zugas - C_O3_Abgas)*Q_Gas/Q_Ozonanlage")) |
| `calc_list_name` | full names of parameters to be used for plotting for each calculation specified wit 'calc_list'. default: c('delta SAK', 'Ozoneintrag') |
| `calc_list_unit` | units of parameters to be used for plotting for each calculation specified wit 'calc_list'. default: c("percent", "mg-O3/L") |
| `calc_paras` | a vector with parameter codes used for performing calculations defined in 'calc_list' (default: c("SCAN_SAK_Ablauf", "SCAN_SAK_Zulauf", "C_O3_Zugas", "C_O3_Abgas", "Q_Gas", "Q_Ozonanlage")) |

### Value

dataframe with calculated operational parameters

### Examples

```
## Not run:
raw_list <- read_wedeco_data()
myDat <- calculate_operational_parameters_berlin_s(df = raw_list)
## End(Not run)
```

## calculate_operational_parameters_berlin_t
*Calculate operational parameters for Berlin-Tiefwerder*

### Description

Calculate operational parameters for Berlin-Tiefwerder

### Usage

```
calculate_operational_parameters_berlin_t(df, calc_list = list(recovery =
  "100*`FY-20-01`/`FT-10-01`"), calc_list_name = c("recovery"),
  calc_list_unit = c("%"), calc_paras = c("FY-20-01", "FT-10-01"))
```

### Arguments

| | |
|---|---|
| `df` | a data frame as retrieved by read_pentair_data() |
| `calc_list` | list with calculation operations to be carried out (default: list(recovery = "100*‘FY-20-01‘/‘FT-10-01‘")) |
| `calc_list_name` | full names of parameters to be used for plotting for each calculation specified wit ’calc_list’. default: c(’recovery’) |
| `calc_list_unit` | units of parameters to be used for plotting for each calculation specified wit ’calc_list’. default: c("percent") |
| `calc_paras` | a vector with parameter codes used for performing calculations defined in ’calc_list’ (default: c("FY-20-01", "FT-10-01") |

### Value

dataframe with calculated operational parameters

### Examples

```
## Not run:
raw_list <- read_pentair_data()
myDat <- calculate_operational_parameters_berlin_t(df = raw_list)
## End(Not run)
```

## `calenderweek_from_dates` *Helper function: get calender weeks for time period*

### Description

Helper function: get calender weeks for time period

### Usage

```
calenderweek_from_dates(start = "2017-04-24", end = Sys.Date())
```

### Arguments

| | |
|---|---|
| `start` | start of period (default: '2017-04-24') |
| `end` | end of period (default: .Date()) |

### Value

data.frame with daily date sequence for and corresponding calendar week

## `change_timezone` *Timezone change: changes time zone to user defined time zone*

### Description

Timezone change: changes time zone to user defined time zone

### Usage

```
change_timezone(df, tz = "UTC", col_datetime = "DateTime",
  debug = TRUE)
```

### Arguments

| | |
|---|---|
| `df` | a dataframe containing a datetime column |
| `tz` | timezone (default: "UTC") |
| `col_datetime` | name of the datetime column (default: "DateTime") |
| `debug` | print debug messages (default: TRUE) |

### Value

returns data frame with changed time zone

## References

Check possible "tz" arguments in column "TZ*" of table https://en.wikipedia.org/wiki/List_of_tz_database_time_zones for more details.

---

## `check_thresholds` *Check thresholds*

---

### Description

Check thresholds

### Usage

```
check_thresholds(df, thresholds = aquanes.report::get_thresholds())
```

### Arguments

| | |
|---|---|
| `df` | a dataframe as retrieved by import_data_haridwar() |
| `thresholds` | thresholds dataframe as retrieved by get_thresholds() (default: "raw") |

### Value

dataframe with thresholds check results for selected time period (i.e. whether Parameters are below/above min/max thresholds defined in dataframe 'thresholds')

---

## `create_monthly_selection` *Create monthly selection*

---

### Description

Create monthly selection

### Usage

```
create_monthly_selection(startDate = "2016-09-01",
  endDate = Sys.Date())
```

### Arguments

| | |
|---|---|
| `startDate` | (default: '2016-09-01') |
| `endDate` | (default: Sys.Date()) (default: "raw") |

### Value

dataframe with first/last day for each month between 'startDate' and 'endDate' month including a column 'label' (used in shiny app for month selection)

## `create_report_batch` *Report batch: creates batch file for report*

### Description

Report batch: creates batch file for report

### Usage

```
create_report_batch(batchDir = file.path(tempdir(), "batch_report"),
  batchName = "create_report.bat", report_path = NULL,
  report_config_path = NULL, open_in_explorer = TRUE)
```

### Arguments

| | |
|---|---|
| `batchDir` | path to report batch directory (default: tempdir()) |
| `batchName` | name for report batch file(default: "create_report.bat") |
| `report_path` | (default: NULL) |
| `report_config_path` | (default: NULL) |
| `open_in_explorer` | open batchDir in Windows explorer (default: TRUE). Only working on a Windows system! |

## `create_wedeco_metafile` *Create WEDECO metafile data*

### Description

Create WEDECO metafile data

### Usage

```
create_wedeco_metafile(raw_data_file)
```

### Arguments

| | |
|---|---|
| `raw_data_file` | file path to raw data which should be used for as template for meta file creation |

### Value

data.frame with meta data file structure

## `dygraph_add_limits` *Dygraph: add (multiple) horizontal lines to plot*

### Description

Dygraph: add (multiple) horizontal lines to plot

### Usage

```
dygraph_add_limits(dygraph, limits_df, label_loc = "left",
  col_limits = "ParameterThreshold", col_label = "label", ...)
```

### Arguments

| | |
|---|---|
| `dygraph` | a dygraph object where (possibly) multiple horizontal lines should be added |
| `limits_df` | dataframe containing the limits information to be added to the dygraph (e.g. as retrieved by function get_thresholds()) |
| `label_loc` | Location for horizontal dygraph labels (left or right). (default: "left) |
| `col_limits` | column in limits_df containing the limits values (default: "ParameterThreshold") |
| `col_label` | column in limits_df containing the label values (default: "label") |
| `...` | further arguments passed to dygraphs::dyLimit() |

### Value

add limits to existing dygraph object

## `get_monthly_data_from_calendarweeks` *Helper function for Berlin-S: get all calendar week files for monthy*

### Description

Helper function for Berlin-S: get all calendar week files for monthy

### Usage

```
get_monthly_data_from_calendarweeks(year_month)
```

### Arguments

| | |
|---|---|
| `year_month` | month to be imported (e.g. 2017-04') |

### Value

character vector with operational filenames with all calendar weeks that need to be imported for Berlin Schoenerlinde

## `get_monthly_periods` *Get monthly periods*

### Description

Get monthly periods

### Usage

```
get_monthly_periods(year_month_start = "2017-06",
  year_month_end = format(Sys.Date(), "%Y-%m"), tz = "CET")
```

### Arguments

| | |
|---|---|
| `year_month_start` | start year month (default: '2017-06') |
| `year_month_end` | end year month (default: current month) |
| `tz` | (default: 'CET') |

### Value

dataframe with monthly periods

## `get_rawfilespaths_for_month` *Berlin-Tiefwerder: get rawfilepaths for months*

### Description

Berlin-Tiefwerder: get rawfilepaths for months

### Usage

```
get_rawfilespaths_for_month(monthly_period = get_monthly_periods()[1, ],
  raw_data_dir = system.file("shiny/berlin_t/data/operation", package =
  "aquanes.report"), max_offset_days = 7)
```

### Arguments

| | |
|---|---|
| `monthly_period` | one row of data.frame as retrieved by function first row of get_monthly_periods(), i.e. year month is (default: '2017-06') |
| `raw_data_dir` | directory with operational raw data files for Berlin Tiefwerder (default: system.file("shiny/berlin_t/data/operation", package = "aquanes.report") |
| `max_offset_days` | number of days in previous/next month to look for beginning/ ending of month (default: 7) |

### Value

dataframe with monthly periods

---

## get_thresholds *Get thresholds for analytics/operational parameters*

---

### Description

Get thresholds for analytics/operational parameters

### Usage

```
get_thresholds(csv_path = system.file(file.path("shiny/haridwar/data",
  "thresholds.csv"), package = "aquanes.report"))
```

### Arguments

| | |
|---|---|
| csv_path | path to csv file with thresholds for Haridwar site (default: system.file(file.path("shiny/haridwar/data", thresholds.csv") |

### Value

returns data frame thresholds for operational/analytical parameters

---

## get_valid_timezones *Timezone: get valid time zones from Wikipedia*

---

### Description

Timezone: get valid time zones from Wikipedia

### Usage

```
get_valid_timezones()
```

### Value

returns data frame valid time zones (column: TZ.) from Wikipedia

### References

Check possible "tz" arguments in column "TZ*" of table `https://en.wikipedia.org/wiki/List_of_tz_database_time_zones` for more details.

## group_datetime — *Group DateTime by user defined period (year, month, day, hour, minute)*

### Description

Group DateTime by user defined period (year, month, day, hour, minute)

### Usage

```
group_datetime(df, by = 600, fun = "stats::median",
  col_datetime = "DateTime", col_datatype = "DataType", dbg = TRUE)
```

### Arguments

| | |
|---|---|
| `df` | a data frame as retrieved by import_data_haridwar() |
| `by` | an aggregation time step in seconds (default: 600 seconds) for intra- day aggregation or "day", "month" or "year" for longer time spans |
| `fun` | function to be used for grouping measurement data of column ParameterValue (default: stats::median) (default: system.file("shiny/haridwar/.my.cnf", package = "aquanes.report")) |
| `col_datetime` | column name of datetime column (default: DateTime) |
| `col_datatype` | column name of data type column (default: DataType) |
| `dbg` | print debug information |

### Value

returns data frame with data aggregated according to user defined aggregation time step

## import_analytics_basel — *Imports analytical data for Basel (without metadata)*

### Description

Imports analytical data for Basel (without metadata)

### Usage

```
import_analytics_basel(csv_dir = system.file("shiny/basel/data/analytics",
package = "aquanes.report"))
```

### Arguments

| | |
|---|---|
| `csv_dir` | Define directory with raw analytical data in CSV (.csv) format to be imported (default: system.file("shiny/basel/data/analytics", package = "aquanes.report")) |

### Value

returns data frame with imported raw analytics data

---

## `import_analytics_meta_basel`

*Imports analytical data for Basel (with metadata for both sites at once, i.e. "rhein" and "wiese")*

---

### Description

Imports analytical data for Basel (with metadata for both sites at once, i.e. "rhein" and "wiese")

### Usage

```
import_analytics_meta_basel(analytics_dir = system.file("shiny/basel/data/analytics",
 package = "aquanes.report"),
 meta_site_path = system.file("shiny/basel/data/metadata/meta_site.csv",
 package = "aquanes.report"),
meta_parameter_path = system.file("shiny/basel/data/metadata/meta_parameter.csv",
 package = "aquanes.report"))
```

### Arguments

| | |
|---|---|
| `analytics_dir` | Define directory with raw analytical data in CSV (.csv) format to be imported (default: system.file("shiny/basel/data/analytics", package = "aquanes.report")) |
| `meta_site_path` | Define path of "meta_site.csv" to be imported (default: system.file("shiny/basel/data/metadata/meta_site.c package = "aquanes.report")) |
| `meta_parameter_path` | Define path of "meta_parameter.csv" to be imported (default: system.file("shiny/basel/data/metadata/meta package = "aquanes.report")) |

### Value

data.frame with analytics data for Basel sites including metadata `import_data_basel` *Imports operational & analytical data for Basel (with metadata for both sites at once, i.e. "rhein" and "wiese")*

## Description

Imports operational & analytical data for Basel (with metadata for both sites at once, i.e. "rhein" and "wiese")

## Usage

```
import_data_basel(analytics_dir = system.file("shiny/basel/data/analytics",
  package = "aquanes.report"),
  raw_dir_rhein = system.file("shiny/basel/data/operation/rhein", package
  = "aquanes.report"),
  raw_dir_wiese = system.file("shiny/basel/data/operation/wiese", package
  = "aquanes.report"),
  meta_online_path = system.file("shiny/basel/data/metadata/meta_online.csv",
  package = "aquanes.report"),
 meta_parameter_path = system.file("shiny/basel/data/metadata/meta_parameter.csv",
  package = "aquanes.report"),
  meta_site_path = system.file("shiny/basel/data/metadata/meta_site.csv",
  package = "aquanes.report"))
```

## Arguments

| | |
|---|---|
| `analytics_dir` | Define directory with raw analytical data in CSV (.csv) format to be imported (default: system.file("shiny/basel/data/analytics", package = "aquanes.report")) |
| `raw_dir_rhein` | Define directory for site "rhein" with raw data in EXCEL spreadsheet format (.xlsx) to be imported (default: system.file("shiny/basel/data/operation/rhein", package = "aquanes.report")) |
| `raw_dir_wiese` | Define directory for site "rhein" with raw data in EXCEL spreadsheet format (.xlsx) to be imported (default: system.file("shiny/basel/data/operation/wiese", package = "aquanes.report")) |
| `meta_online_path` | path to file containing metadata for online data (default: system.file("shiny/basel/data/metadata/meta_onli package = "aquanes.report")) |
| `meta_parameter_path` | Define path of "meta_parameter.csv" to be imported (default: system.file("shiny/basel/data/metadata/meta package = "aquanes.report")) |
| `meta_site_path` | Define path of "meta_site.csv" to be imported (default: system.file("shiny/basel/data/metadata/meta_site.c package = "aquanes.report")) |

## Value

data.frame with analytical & operational data for Basel

## `import_data_berlin_s` *Import data for Berlin Schoenerlinde*

### Description

Import data for Berlin Schoenerlinde

### Usage

```
import_data_berlin_s(raw_data_dir = system.file("shiny/berlin_s/data/operation",
  package = "aquanes.report"), raw_data_files = NULL,
 meta_file_path = system.file("shiny/berlin_s/data/parameter_site_metadata.csv",
  package = "aquanes.report"))
```

### Arguments

| | |
|---|---|
| `raw_data_dir` | path of directory containing WEDECO CSV files (default: (default: system.file("shiny/berlin_s/data/opera package = "aquanes.report")))) |
| `raw_data_files` | vector with full path to operational raw data files that allows to limit import to specific files (default: NULL). If specified parameter "raw_data_dir" will not be used |
| `meta_file_path` | path to metadata file (default: system.file("shiny/berlin_s/data/parameter_site_metadata.csv", package = "aquanes.report"))) |

### Value

list with "df": data.frame with imported operational data (analytics data to be added as soon as available) and "added_data_points": number of added data points in case of existing fst file was updated with new operational data

## `import_data_berlin_t` *Import data for Berlin Tiefwerder*

### Description

Import data for Berlin Tiefwerder

### Usage

```
import_data_berlin_t(raw_data_dir = system.file("shiny/berlin_t/data/operation",
  package = "aquanes.report"), raw_data_files = NULL,
  analytics_path = system.file("shiny/berlin_t/data/analytics.xlsx",
  package = "aquanes.report"),
 meta_file_path = system.file("shiny/berlin_t/data/parameter_site_metadata.csv",
  package = "aquanes.report"))
```

### Arguments

| | |
|---|---|
| `raw_data_dir` | path of directory containing PENTAIR xls files (default: (default: system.file("shiny/berlin_t/data/operatio package = "aquanes.report")))) |
| `raw_data_files` | vector with full path to operational raw data files that allows to limit import to specific files (default: NULL). If specified parameter "raw_data_dir" will not be used |
| `analytics_path` | full path to lab data EXCEL file in xlsx format (default: (default: system.file("shiny/berlin_t/data/analytics package = "aquanes.report")))) |
| `meta_file_path` | path to metadata file (default: system.file("shiny/berlin_t/data/parameter_site_metadata.csv", package = "aquanes.report"))) |

### Value

data.frame with imported operational data (analytics´data to be added as soon as available)

---

## `import_data_haridwar` *Imports Haridwar data*

---

### Description

Imports Haridwar data

### Usage

```
import_data_haridwar(analytics_path = system.file(file.path("shiny",
  "haridwar/data/analytics.xlsx"), package = "aquanes.report"),
  operation_mySQL_conf = system.file("shiny/haridwar/.my.cnf", package =
  "aquanes.report"),
  operation_meta_path = system.file(file.path("shiny/haridwar/data",
  "operation_parameters.csv"), package = "aquanes.report"),
  excludedSheets = c("Parameters", "Location", "Sites", "#Summary",
  "Site_and_Parameter", "Observations", "dP", "ORP", "Flow",
  "Current_Voltage", "As_total_Arsenator"), skip = 69, debug = TRUE)
```

### Arguments

| | |
|---|---|
| `analytics_path` | Define path of analytics EXCEL spreadsheet to be imported (default: system.file(file.path("shiny/haridwa "analytics.xlsx"), package = "aquanes.report")) |
| `operation_mySQL_conf` | column name pattern for identifying raw data (default: system.file("shiny/haridwar/.my.cnf", package = "aquanes.report")) |
| `operation_meta_path` | path to table with meta data for operational parameters (default: system.file(file.path("shiny/haridwar/data "operation_parameters.csv"), package = "aquanes.report")) |

| | |
|---|---|
| `excludedSheets` | all sheets, which are not listed here will be imported as lab data sheets (default: c("Parameters", "Location", "Sites", "#Summary", "Site_and_Parameter", "Observations", "dP", "ORP", "Flow", "Current_Voltage", "As_total_Arsenator")) |
| `skip` | number of rows to skip for each lab data sheet (default: 69), i.e. for all sheets which are not explictly excluded with parameter "excludedSheets" |
| `debug` | if TRUE print debug messages (default: TRUE) |

### Value

returns data frame with Haridwar raw data (operation & analytics)

---

## `import_lab_data_berlin_t` *BerlinTiefwerder: import lab data*

---

### Description

BerlinTiefwerder: import lab data

### Usage

```
import_lab_data_berlin_t(xlsx_path = system.file("shiny/berlin_t/data/analytics.xlsx",
 package = "aquanes.report"))
```

### Arguments

| | |
|---|---|
| `xlsx_path` | full path to lab data EXCEL file in xlsx format (default: (default: system.file("shiny/berlin_t/data/analytics package = "aquanes.report")))) |

### Value

a list of imported lab data for Berlin-Tiefwerder

---

## `import_operation` *Imports operational data*

---

### Description

Imports operational data

### Usage

```
import_operation(mysql_conf = file.path(getwd(), ".my.cnf"))
```

### Arguments

| | |
|---|---|
| `mysql_conf` | path to the MySQL configuration file |

### Value

returns data frame operational data from MySQL db

---

## `import_operation_basel` *Imports operational data for Basel (without metadata and only for one site at once, e.g. "rhein" or "wiese")*

---

### Description

Imports operational data for Basel (without metadata and only for one site at once, e.g. "rhein" or "wiese")

### Usage

```
import_operation_basel(xlsx_dir = system.file("shiny/basel/data/operation/wiese",
 package = "aquanes.report"))
```

### Arguments

| | |
|---|---|
| `xlsx_dir` | Define directory with raw data in EXCEL spreadsheet (.xlsx) to be imported (default: system.file("shiny/basel/data/operation/wiese", package = "aquanes.report")) |

### Value

returns data frame with imported raw operational data

---

## `import_operation_meta_basel` *Imports operational data for Basel (with metadata for both sites at once, i.e. "rhein" and "wiese")*

---

### Description

Imports operational data for Basel (with metadata for both sites at once, i.e. "rhein" and "wiese")

## Usage

```
import_operation_meta_basel(raw_dir_rhein = system.file(file.path("shiny",
  "basel/data/operation/rhein"), package = "aquanes.report"),
  raw_dir_wiese = system.file("shiny/basel/data/operation/wiese", package
  = "aquanes.report"),
  meta_online_path = system.file("shiny/basel/data/metadata/meta_online.csv",
  package = "aquanes.report"),
  meta_site_path = system.file("shiny/basel/data/metadata/meta_site.csv",
  package = "aquanes.report"),
 meta_parameter_path = system.file("shiny/basel/data/metadata/meta_parameter.csv",
  package = "aquanes.report"))
```

## Arguments

| | |
|---|---|
| `raw_dir_rhein` | Define directory for site "rhein" with raw data in EXCEL spreadsheet format (.xlsx) to be imported (default: system.file("shiny/basel/data/operation/rhein", package = "aquanes.report")) |
| `raw_dir_wiese` | Define directory for site "rhein" with raw data in EXCEL spreadsheet format (.xlsx) to be imported (default: system.file("shiny/basel/data/operation/wiese", package = "aquanes.report")) |
| `meta_online_path` | path to file containing metadata for online data (default: system.file("shiny/basel/data/metadata/meta_onli package = "aquanes.report")) |
| `meta_site_path` | Define path of "meta_site.csv" to be imported (default: system.file("shiny/basel/data/metadata/meta_site.c package = "aquanes.report")) |
| `meta_parameter_path` | Define path of "meta_parameter.csv" to be imported (default: system.file("shiny/basel/data/metadata/meta package = "aquanes.report")) |

## Value

returns data frame with imported raw operational data with metadata for both sites (i.e."rhein" and "wiese")

data.frame with operational data for Basel sites including metadata

---

`import_sheets` *Imports multiple analytics sheets from an EXCEL spreadsheet*

---

## Description

Imports multiple analytics sheets from an EXCEL spreadsheet

### Usage

```
import_sheets(xlsPath, sheets_analytics, sheet_parameters = "Parameters",
  sheet_sites = "Sites", sheet_location = "Location",
  col_rawData_pattern = "raw",
  col_ignore_pattern = "mean|empty|X_|RX|not_used", skip = 69,
  tz_org = NULL, tz_export = "UTC", dbg = TRUE)
```

### Arguments

| | |
|---|---|
| `xlsPath` | path to xls file with analytics data |
| `sheets_analytics` | a character vector with the names of the sheets with analytics data (check with: readxl::excel_sheets(xlsPath)) |
| `sheet_parameters` | sheet name containing parameter metadata (default: "Parameters") |
| `sheet_sites` | sheet name containing sites metadata (default: "Sites") |
| `sheet_location` | sheet name containing location metadata (default: "Location") |
| `col_rawData_pattern` | specify pattenr of columns containing raw data (default: "raw") |
| `col_ignore_pattern` | specify pattern of columns that should be ignored of importing (default: "mean\|empty\|X_\|RX\|not_used") |
| `skip` | number of rows in sheet to skip (default: 69), |
| `tz_org` | specify timezone of samples (default: "UTC") |
| `tz_export` | specify timezone for data export (default: "UTC") |
| `dbg` | print debug messages (default: TRUE) |

### Value

returns data frame with normalised analytics data in list form

---

## `load_fst_data` *Load fst data for shiny app*

---

### Description

Load fst data for shiny app

### Usage

```
load_fst_data(fst_dir)
```

### Arguments

| | |
|---|---|
| `fst_dir` | directory of fst files to be loaded |

## merge_and_export_fst *Helper function: merge and export fst files into main shiny data folder*

### Description

Helper function: merge and export fst files into main shiny data folder

### Usage

```
merge_and_export_fst(time_pattern = NULL, compression = 100,
  import_dir = system.file("shiny/berlin_t/data/fst", package =
  "aquanes.report"), export_dir = system.file("shiny/berlin_t/data",
  package = "aquanes.report"))
```

### Arguments

| | |
|---|---|
| `time_pattern` | optional pattern to filter months to be imported (default: NULL), for using it do e.g. "2017-06\|2017-07" or c("2017-06", "2017-07") |
| `compression` | compression for fst export (default: 100) |
| `import_dir` | directory with fst files or subdirs to be imported (default: system.file("shiny/berlin_t/data/fst",package = "aquanes.report")) |
| `export_dir` | directory with fst directory for export (default: system.file("shiny/berlin_t/data",package = "aquanes.report")) |

### Value

imports multiple fst files and exports them to be used for app

## multiSubstitute *Multiple Substitutions*

### Description

apply multiple substitutions on a vector of character. For each element in *replacements* gsub is called with the element name being the pattern and the element value being the replacement.

### Usage

```
multiSubstitute(strings, replacements, ..., dbg = FALSE)
```

### Arguments

| | |
|---|---|
| `strings` | vector of character |
| `replacements` | list of pattern = replacement pairs. |
| `...` | additional arguments passed to gsub |
| `dbg` | if `TRUE` (the default is `FALSE`) it is shown which strings were replaced |

## `plot_analytics` *Plot analytics data (in PDF)*

### Description

Plot analytics data (in PDF)

### Usage

```
plot_analytics(df)
```

### Arguments

| | |
|---|---|
| `df` | dataframe as retrieved by import_sheets() |

### Value

creates new subdirectory "/report" in current working directory and stores pdf plots there

## `plot_calculated_operational_timeseries` *Plot calculate operational time series*

### Description

Plot calculate operational time series

### Usage

```
plot_calculated_operational_timeseries(df)
```

### Arguments

| | |
|---|---|
| `df` | a data frame as retrieved by calculate_operational_parameters() |

### Value

plots time series for calculated operational parameters

### Examples

```
## Not run:
haridwar_raw_list <- import_data_haridwar()
myDat <- calculate_operational_parameters(df = haridwar_raw_list)
plot_calculated_operational_timeseries(myDat)
## End(Not run)
```

## `read_fst` *Wrapper for fst::read.fst to read DateTime column in POSIXct format*

### Description

Wrapper for fst::read.fst to read DateTime column in POSIXct format

### Usage

```
read_fst(path, tz = "CET", col_datetime = "DateTime", ...)
```

### Arguments

| | |
|---|---|
| `path` | path to fst file |
| `tz` | timezone of DateTime to be imported (default: "CET") |
| `col_datetime` | column name containing numeric values in nanoseconds since 1970-01-01 (default: "DateTime") |
| `...` | further arguments passed to fst::read.fst |

### Value

data.frame with formatting of DateTime column POSIXct

## `read_pentair_data` *Read PENTAIR operational data*

### Description

Read PENTAIR operational data

### Usage

```
  read_pentair_data(raw_data_dir = system.file("shiny/berlin_t/data/operation",
  package = "aquanes.report"), raw_data_files = NULL,
 meta_file_path = system.file("shiny/berlin_t/data/parameter_site_metadata.csv",
  package = "aquanes.report"))
```

### Arguments

| | |
|---|---|
| `raw_data_dir` | path of directory containing PENTAIR xls files (default: (default: system.file("shiny/berlin_t/data/operatio package = "aquanes.report")))) |
| `raw_data_files` | vector with full path to operational raw data files that allows to limit import to specific files (default: NULL). If specified parameter "raw_data_dir" will not be used |
| `meta_file_path` | path to metadata file (default: system.file("shiny/berlin_t/data/parameter_site_metadata.csv", package = "aquanes.report"))) |

### Value

data.frame with imported PENTAIR operational data

---

## `read_wedeco_data` *Import WEDECO raw data*

---

### Description

Import WEDECO raw data

### Usage

```
read_wedeco_data(raw_data_dir = system.file("shiny/berlin_s/data/operation",
 package = "aquanes.report"), raw_data_files = NULL,
meta_file_path = system.file("shiny/berlin_s/data/parameter_site_metadata.csv",
 package = "aquanes.report"))
```

### Arguments

| | |
|---|---|
| `raw_data_dir` | path to raw data directory |
| `raw_data_files` | vector with full path to operational raw data files that allows to limit import to specific files (default: NULL). If specified parameter "raw_data_dir" will not be used |
| `meta_file_path` | path to meta data file |

---

## `remove_duplicates` *Remove duplicates in data.frame*

---

### Description

Remove duplicates in data.frame

### Usage

```
remove_duplicates(df, col_names = names(df))
```

### Arguments

| | |
|---|---|
| `df` | data.frame to be checked for duplicates |
| `col_names` | column names to be used for duplicate checking (default: names(df)). can be defined by providing: c("col_name1", "col_name2") |

### Value

data.frame without duplicates

## report_config_template *Report config: generate template*

### Description

Report config: generate template

### Usage

```
report_config_template(df = NULL, temporal_aggregation = "raw",
  output_timezone = "UTC")
```

### Arguments

| | |
|---|---|
| df | a dataframe as retrieved by import_data_haridwar() |
| temporal_aggregation | Set the following values if data should be summarised to e.g. 10 minutes (600) or hourly (3600), daily ("day") or monthly ("month") median values (default: "raw") |
| output_timezone | into which timezone should the data be converted for the report? (default: "UTC") |

### Value

default list for report configuration template

## report_config_to_txt *Report config: saves config to text file*

### Description

Report config: saves config to text file

### Usage

```
report_config_to_txt(config_list, output_file = "report_config.txt")
```

### Arguments

| | |
|---|---|
| config_list | a report configuration list e.g. as retrieved by report_config_template() |
| output_file | absolute or relative path of where to save output file (default: "report_config.txt") |

## Value

saves report configuration list as text file

## Examples

```
## Not run:
### Creates a configuration template
config <- report_config_template()
### Saves list config in text
report_config_to_txt(config_list = config,
output_file = "report_config.txt")

## End(Not run)
```

---

`report_txt_to_config` *Report config: imports text file to list*

---

## Description

Report config: imports text file to list

## Usage

```
report_txt_to_config(config_txt = "report_config.txt")
```

## Arguments

| | |
|---|---|
| `config_txt` | path to report configuration text file created by a report configuration list e.g. as retrieved by function report_config_to_txt() |

## Value

saves report configuration list as text file

## Examples

```
## Not run:
### Creates a configuration template
config <- report_config_template()
### Saves list config in text
report_config_to_txt(config_list = config, output_file = "report_config.txt")
### Reads config list from text file to
config_imported <- report_txt_to_config(config_txt = "report_config.txt")
### Check whether both are identical
identical(x = config, y = config_imported)

## End(Not run)
```

## run_app — *Runs Shiny app for an AQUANES site*

### Description

Runs Shiny app for an AQUANES site

### Usage

```
run_app(siteName = "haridwar", use_live_data = FALSE,
  mySQL_conf = NULL, launch.browser = TRUE, ...)
```

### Arguments

| | |
|---|---|
| `siteName` | site name for shiny app (default: "haridwar") |
| `use_live_data` | should live data be used (default: FALSE) |
| `mySQL_conf` | file path to mySQL config file (.my.cnf). Only used if parameter use_live_data is TRUE and there is no .my.cnf in the app folder for the selected site (default: NULL) |
| `launch.browser` | If true, the system's default web browser will be launched automatically after the app is started (default: TRUE) |
| `...` | further arguments passed to shiny::runApp() |

## set_timezone — *Timezone set: sets a user defined time zone*

### Description

Timezone set: sets a user defined time zone

### Usage

```
set_timezone(df, tz = "UTC", col_datetime = "DateTime")
```

### Arguments

| | |
|---|---|
| `df` | a dataframe containing a datetime column |
| `tz` | timezone (default: "UTC") |
| `col_datetime` | name of the datetime column (default: "DateTime") |

### Value

returns data frame with specified time zone

## References

Check possible "tz" arguments in column "TZ*" of table https://en.wikipedia.org/wiki/List_of_tz_database_time_zones for more details.

# Index

add_label, 3
add_parameter_metadata, 3
add_site_metadata, 4
aggregate_export_fst_berlin_s, 5
aggregate_export_fst_berlin_t, 5

calculate_operational_parameters, 6
calculate_operational_parameters_berlin_s, 7
calculate_operational_parameters_berlin_t, 8
calenderweek_from_dates, 9
change_timezone, 9
check_thresholds, 10
create_monthly_selection, 10
create_report_batch, 11
create_wedeco_metafile, 11

dygraph_add_limits, 12

get_monthly_data_from_calendarweeks, 12
get_monthly_periods, 13
get_rawfilespaths_for_month, 13
get_thresholds, 14
get_valid_timezones, 14
group_datetime, 15

import_analytics_basel, 15
import_analytics_meta_basel, 16
import_data_basel, 17
import_data_berlin_s, 18
import_data_berlin_t, 18
import_data_haridwar, 19
import_lab_data_berlin_t, 20
import_operation, 20
import_operation_basel, 21
import_operation_meta_basel, 21
import_sheets, 22

load_fst_data, 23

merge_and_export_fst, 24
multiSubstitute, 24

plot_analytics, 25
plot_calculated_operational_timeseries, 25

read_fst, 26
read_pentair_data, 26
read_wedeco_data, 27
remove_duplicates, 27
report_config_template, 28
report_config_to_txt, 28
report_txt_to_config, 29
run_app, 30

set_timezone, 30