Package ‘gDRimport’

July 2, 2025

Type Package

Title Package for handling the import of dose-response data

Version 1.6.0

Date 2025-04-10

Description The package is a part of the gDR suite. It helps to prepare raw drug response data for downstream processing. It mainly contains helper functions for importing/loading/validating dose-response data provided in different file formats.

License Artistic-2.0

LazyLoad yes

Depends R (>= 4.2)

Imports assertthat, BumpyMatrix, checkmate, CoreGx, PharmacoGx, data.table, futile.logger, gDRutils (>= 1.3.17), magrittr, methods, MultiAssayExperiment, readxl, rio, S4Vectors, stats, stringi, SummarizedExperiment, tibble, tools, utils, XML, yaml, openxlsx

Suggests BiocStyle, gDRtestData (>= 1.3.3), gDRstyle (>= 1.3.3), knitr, purrr, qs, testthat

URL `https://github.com/gdrplatform/gDRimport`, `https://gdrplatform.github.io/gDRimport/`

BugReports `https://github.com/gdrplatform/gDRimport/issues`

biocViews Software, Infrastructure, DataImport

VignetteBuilder knitr

ByteCompile TRUE

Roxygen list(markdown = TRUE)

RoxygenNote 7.3.2

SwitchrLibrary gDRimport

DeploySubPath gDRimport

Encoding UTF-8

git_url https://git.bioconductor.org/packages/gDRimport

git_branch RELEASE_3_21

git_last_commit ec46ae0

git_last_commit_date 2025-04-15

Repository Bioconductor 3.21

Date/Publication 2025-07-02

Author Arkadiusz Gladki [aut, cre] (ORCID: <https://orcid.org/0000-0002-7059-6378>),
Bartosz Czech [aut] (ORCID: <https://orcid.org/0000-0002-9908-3007>),
Marc Hafner [aut] (ORCID: <https://orcid.org/0000-0003-1337-7598>),
Sergiu Mocanu [aut],
Dariusz Scigocki [aut],
Allison Vuong [aut],
Luca Gerosa [aut] (ORCID: <https://orcid.org/0000-0001-6805-9410>),
Janina Smola [aut]

Maintainer Arkadiusz Gladki <gladki.arkadiusz@gmail.com>

Contents

`gDRimport-package` *gDRimport: Package for handling the import of dose-response data*

Description

The package is a part of the gDR suite. It helps to prepare raw drug response data for downstream processing. It mainly contains helper functions for importing/loading/validating dose-response data provided in different file formats.

Value

package help page

Note

To learn more about functions start with `help(package = "gDRimport")`

Author(s)

Maintainer: Arkadiusz Gladki `<gladki.arkadiusz@gmail.com>` (ORCID)

Authors:

- Bartosz Czech (ORCID)
- Marc Hafner (ORCID)
- Sergiu Mocanu
- Dariusz Scigocki
- Allison Vuong
- Luca Gerosa (ORCID)
- Janina Smola

See Also

Useful links:

- `https://github.com/gdrplatform/gDRimport`
- `https://gdrplatform.github.io/gDRimport/`
- Report bugs at `https://github.com/gdrplatform/gDRimport/issues`

`.check_against_single_template_sheet` *Evaluate if template file with single sheet is present, if the name of the sheet is correct and if it can be fixed*

Description

get sheets for given set of XLS files

Usage

```
.check_against_single_template_sheet(ts)
```

Arguments

`ts`	list with template sheets info

Value

logical flag

`.check_file_structure` *Check the structure of raw data*

Description

Check the structure of raw data

Usage

```
.check_file_structure(
  df,
  filename,
  sheet_name,
  readout_offset,
  n_row,
  n_col,
  bcode_idx,
  bcode_col
)
```

Value

`NULL` invisibly.

`.createPseudoData` *Add in pseudo-data for duration and cell reference division time*

Description

Add in pseudo-data for duration and cell reference division time

Usage

```
.createPseudoData(dt)
```

Value

data.table

`.extractDoseResponse` *Get dose and viability readouts and melt into large data table*

Description

Get dose and viability readouts and melt into large data table

Usage

```
.extractDoseResponse(pset)
```

Value

data.table with dose-response data

`.extract_or_create_assay`
Extracts an assay from a SummarizedExperiment object or creates a new one if it does not exist

---

Description

This function takes a SummarizedExperiment object and an assay name as input. If the specified assay already exists in the SummarizedExperiment object, it is returned. Otherwise, a new assay with the specified name is created and added to the SummarizedExperiment object. The new assay is initialized with NA values. This is useful for when multiple Summarized Experiments in a given MAE do not have the same assays. And it is necessary to have the same assays in all Summarized Experiments in order to convert the MAE to a PSet.

Usage

```
.extract_or_create_assay(SE, assay_name)
```

Arguments

`SE`	A SummarizedExperiment object
`assay_name`	A character string specifying the name of the assay to extract or create

Value

A SummarizedExperiment object with the specified assay

`.fill_empty_wells` *Correct plates with not fully filled readout values*

Description

Correct plates with not fully filled readout values

Usage

```
.fill_empty_wells(
  df,
  plate_rows,
  data_rows,
  exp_row,
  exp_col,
  numeric_regex = "^\\d+$"
)
```

Value

data.table with corrected plates data

`.get_plate_size` *Get plate size*

Description

Get plate size

Usage

```
.get_plate_size(df)
```

Details

All plate sizes assume 1.5x nrows = ncolumns.

Value

charvec with plate dims

`.removeNegatives` *Remove negative viabilities*

Description

Remove negative viabilities

Usage

```
.removeNegatives(dataset)
```

Value

data.table with positive values in column `ReadoutValue`

`.standardize_untreated_values` *Standardize untreated values to ignore cases*

Description

Standardize untreated values to ignore cases

Usage

```
.standardize_untreated_values(df)
```

Value

data.table with standardized untreated values

`are_template_sheets_valid`
are template sheet valid?

Description

are template sheet valid?

Usage

```
are_template_sheets_valid(ts)
```

Arguments

`ts`	list with (per file) template sheets

Value

logical flag

See Also

get_xl_sheets

`check_metadata_against_spaces`
Check metadata against spaces

Description

Check metadata against spaces

Usage

```
check_metadata_against_spaces(corrected_names, df_name)
```

Arguments

`corrected_names`	a charvec with corrected colnames of df
`df_name`	a name of data.table ("" by default)

Value

a charvec with corrected colnames of df

`check_metadata_field_names`
Check metadata field names

Description

Check metadata field names

Usage

```
check_metadata_field_names(corrected_names, df_name)
```

Arguments

`corrected_names`	a charvec with corrected colnames of df
`df_name`	a name of data.table ("" by default)

Value

a charvec with corrected colnames of df

`check_metadata_headers`
Check whether metadata headers are correct and make fixes if needed

---

Description

Check whether metadata headers are correct and make fixes if needed

Usage

```
check_metadata_headers(corrected_names, df_name)
```

Arguments

`corrected_names`	a charvec with corrected colnames of df
`df_name`	a name of data.table ("" by default)

Value

a charvec with corrected colnames of df

`check_metadata_names` *check_metadata_names*

Description

Check whether all metadata names are correct

Usage

```
check_metadata_names(col_df, df_name = "", df_type = NULL)
```

Arguments

`col_df`	a character with colnames of df
`df_name`	a name of data.table ("" by default)
`df_type`	a type of a data.table (NULL by default)

Value

a charvec with corrected colnames of df

Examples

```
td <- get_test_data()
m_file <- manifest_path(td)
m_data <- read_excel_to_dt(m_file)
result <- check_metadata_names(col_df = colnames(m_data))
```

`check_metadata_req_col_names`

Check metadata for required column names

Description

Check metadata for required column names

Usage

```
check_metadata_req_col_names(col_df, df_name, df_type)
```

Arguments

`col_df`	a charvec with corrected colnames of df
`df_name`	a name of data.table ("" by default)
`df_type`	a type of a data.table (NULL by default)

Value

`NULL` invisibly.

---

convert_LEVEL5_prism_to_gDR_input

Load, convert and process the level 5 PRISM data into a gDR input

---

Description

Load, convert and process the level 5 PRISM data into a gDR input

Usage

```
convert_LEVEL5_prism_to_gDR_input(
  prism_data_path,
  meta_data_path,
  readout_min = 1.03
)
```

Arguments

`prism_data_path`	path to PRISM LEVEL5 csv file with data
`meta_data_path`	path to metadata file describing all cancer models/cell lines which are referenced by a dataset contained within the DepMap portal
`readout_min`	minimum ReadoutValue

Value

`data.table` object with input data for gDR pipeline

Examples

```
prism_data <- system.file("testdata/prism_sa.csv", package = "gDRimport")
prism_meta <- system.file("testdata/prism_model.csv", package = "gDRimport")
convert_LEVEL5_prism_to_gDR_input(prism_data, prism_meta)
```

`convert_LEVEL6_prism_to_gDR_input`
Load, convert and process the level 6 PRISM data into a gDR input

---

Description

Load, convert and process the level 6 PRISM data into a gDR input

Usage

```
convert_LEVEL6_prism_to_gDR_input(
  prism_data_path,
  cell_line_data_path,
  treatment_data_path,
  meta_data_path,
  readout_min = 1.03
)
```

Arguments

`prism_data_path`	path to PRISM LEVEL6 csv file with collapsed log fold change data
`cell_line_data_path`	path to cell line info data
`treatment_data_path`	path to collapsed treatment info data
`meta_data_path`	path to metadata file describing all cancer models/cell lines which are referenced by a dataset contained within the DepMap portal
`readout_min`	minimum ReadoutValue

Value

`data.table` object with input data for gDR pipeline

Examples

```
prism_data_path <- system.file("testdata/prism_collapsed_LOGFC.csv", package = "gDRimport")
cell_line_data_path <- system.file("testdata/prism_cell_lines.csv", package = "gDRimport")
treatment_data_path <- system.file("testdata/prism_treatment.csv", package = "gDRimport")
 prism_meta <- system.file("testdata/prism_model.csv", package = "gDRimport")
convert_LEVEL6_prism_to_gDR_input(prism_data_path, cell_line_data_path, treatment_data_path, prism_meta)
```

`convert_MAE_to_PSet` *Convert MultiAssayExperiment to TreatmentResponseExperiment*

Description

This function converts a MultiAssayExperiment generated by gDR into a TreatmentResponseExperiment for use in the PharmacoGx package. The resulting PharmacoSet can be used for pharmacogenomic analysis of drug response.

Usage

```
convert_MAE_to_PSet(mae, pset_name)
```

Arguments

`mae`	A MultiAssayExperiment object generated by gDR.
`pset_name`	A character string specifying the name of the resulting PharmacoSet object.

Value

A PharmacoSet object.

Examples

```
# Convert a MultiAssayExperiment object to a PharmacoSet object
m <- 20
n <- 10
rnames <- LETTERS[1:m]
cnames <- letters[1:n]
ref_gr_value <-  matrix(runif(m * n), nrow = m, ncol = n, dimnames = list(rnames, cnames))
se <- SummarizedExperiment::SummarizedExperiment(assays = list(RefGRvalue = ref_gr_value),
                                                  rowData = S4Vectors::DataFrame(rnames),
                                                  colData = S4Vectors::DataFrame(cnames))
mae <- MultiAssayExperiment::MultiAssayExperiment(experiments = list("single-agent" = se))
convert_MAE_to_PSet(mae, "my_pset")
```

`convert_pset_to_df` *Convert a PharmacoSet to a data.table that is prepare for input into gDR pipeline*

Description

Convert a PharmacoSet to a data.table that is prepare for input into gDR pipeline

Usage

```
convert_pset_to_df(pharmacoset, run_parallel = TRUE, workers = 2L)
```

Arguments

`pharmacoset`	PharmacoSet object
`run_parallel`	logical, TRUE (default) if to run functions in Parallel, FALSE to run in serial
`workers`	integer, number of workers defaults to 2L if run_parallel is TRUE

Value

data.table of PharmacoSet's dose response data with column names aligned with gDR standard

Author(s)

Jermiah Joseph – collaboration with BHKLab

Examples

```
pset <- suppressMessages(getPSet(
  "Tavor_2020",
  psetDir = system.file("extdata/pset", package = "gDRimport"),
  use_local_PSets_list = TRUE
))
dt <- convert_pset_to_df(pset)
gDRutils::reset_env_identifiers()
```

`correct_template_sheets` *Correct names of the template sheets (if required)*

Description

Correct names of the template sheets (if required)

Usage

```
correct_template_sheets(tfiles)
```

Arguments

`tfiles`	charvec with paths to template files

Value

charvec with paths to corrected sheet names

`detect_file_format` *Detect format of results data*

Description

Detect format of results data

Usage

```
detect_file_format(results_file)
```

Arguments

`results_file`	path to results data

Value

string of the detected file format

Examples

```
td2 <- get_test_Tecan_data()
detect_file_format(td2$r_files[1])
```

`enhance_raw_edited_EnVision_df` *Enhance raw edited EnVision data.table*

Description

Enhance raw edited EnVision data.table

Usage

```
enhance_raw_edited_EnVision_df(df, barcode_col, headers)
```

Arguments

`df`	raw data.table
`barcode_col`	column number for barcode data
`headers`	list with the headersa

Value

data.table derived from EnVision data

`fix_typos_with_reference` *Fix typos using reference data*

Description

Fix typos using reference data Evaluate given list of ids and try to update them

Usage

```
fix_typos_with_reference(
  data,
  ref,
  method = c("exact", "grepl", "adist"),
  fix_underscores = FALSE
)
```

Arguments

`data`	list of charvec(s) or charvec with data
`ref`	charvec with reference data
`method`	charvec type of the method to be used 'exact' is used to find identical entries from 'ref' in the data (after corrections and uppercase'ing) 'grepl' is used to find entries from 'ref' that might be somehow pre- or post- fixed
`fix_underscores`	logical flag fix the issues with underscores in data identfiers?

Value

list or charvec with corrected data

`gdr_test_data-class` *gDR Test Data object*

Description

Object class `gdr_test_data` is build by function `get_test_data()`

Value

object class `gdr_test_data` with primary test data

Slots

`manifest_path`	character, path to manifest file
`result_path`	character, path(s) to results file
`template_path`	character, path(s) to data.table with template data
`ref_m_df`	character, data.table with manifest data
`ref_r1_r2`	character, path to reference file with raw data for treated & untreated
`ref_r1`	character, path to reference file with raw data for treated
`ref_t1_t2`	character, path to reference template file with treated & untreated data
`ref_t1`	character, path to reference template file with treated data

---

`getPSet` *Get PharmacoSet*

Description

Get PharmacoSet

Usage

```
getPSet(
  pset_name,
  psetDir = getwd(),
  canonical = FALSE,
  timeout = 600,
  use_local_PSets_list = FALSE
)
```

Arguments

`pset_name`	string with the name of the PharmacoSet
`psetDir`	string with the temporary directory for the PharmacoSet
`canonical`	logical flag indicating if the PSet canonical
`timeout`	maximum number of seconds allowed for PSet download
`use_local_PSets_list`	logical flag if PSets list should be used from local. If FALSE PSets list will be taken from web.

Value

PharmacoSet object

Examples

```
suppressMessages(getPSet(
  "Tavor_2020",
  psetDir = system.file("extdata/pset", package = "gDRimport"),
  use_local_PSets_list = TRUE
))
```

`get_df_from_raw_edited_EnVision_df`

Get final results (as a data.table) from raw edited EnVision data.table

---

Description

Get final results (as a data.table) from raw edited EnVision data.table

Usage

```
get_df_from_raw_edited_EnVision_df(
  df,
  barcode_idx,
  barcode_col,
  n_row,
  n_col,
  fname,
  sheet_name,
  headers
)
```

Arguments

`df`	raw data.table
`barcode_idx`	numeric vector with barcode indices
`barcode_col`	column number for barcode data
`n_row`	number of rows
`n_col`	number of columns
`fname`	file name
`sheet_name`	name of the Excel sheet
`headers`	list with the headersa

Value

data.table derived from EnVision data

`get_df_from_raw_unedited_EnVision_df` *Get final results (as a data.table) from raw unedited EnVision data.table*

Description

Get final results (as a data.table) from raw unedited EnVision data.table

Usage

```
get_df_from_raw_unedited_EnVision_df(df, n_row, n_col, barcode_col)
```

Arguments

`df`	raw data.table
`n_row`	number of rows
`n_col`	number of columns
`barcode_col`	column number for barcode data

Value

data.table derived from EnVision data

`get_EnVision_properties` *Get properties of EnVision data*

Description

This function return properties of EnVision data

Usage

```
get_EnVision_properties(results.list, fname)
```

Arguments

`results.list`	list with EnVision data
`fname`	name of the input file

Value

list with EnVision propertiesa

`get_excel_sheet_names` *get Excel sheets names for a charvec of files for non-Excel files return 0*

Description

get Excel sheets names for a charvec of files for non-Excel files return 0

Usage

```
get_excel_sheet_names(fls)
```

Arguments

`fls`	charvec with file pathsa

Value

list with one element per file with sheet names or 0 (for non-Excel file)

`get_exception_data` *get exception data*

Description

get exception data

Usage

```
get_exception_data(status_code = NULL)
```

Arguments

`status_code`	A numeric value

Value

A data.table row with exception data or all exceptions

Examples

```
get_exception_data(1)
get_exception_data()
```

`get_expected_template_sheets` *Get names of the sheets expected in templates xlsx*

Description

Get names of the sheets expected in templates xlsx

Usage

```
get_expected_template_sheets(type = c("all", "core", "optional"))
```

Arguments

`type`	charvec type of the sheets

Value

string with type of the sheets

`get_plate_info_from_template_xlsx` *Get plate info from template xlsx*

Description

Get plate info from template xlsx

Usage

```
get_plate_info_from_template_xlsx(template_file, Gnumber_idx, idx)
```

Arguments

`template_file`	character, file path(s) to template(s)
`Gnumber_idx`	index with Gnumber data
`idx`	template file index

Value

list with plate info

`get_test_D300_data` *get test D300 data*

Description

get test D300 data

Usage

```
get_test_D300_data()
```

Value

list with with input data (manifest/template/result paths) and related reference data (qs file paths)

Examples

```
get_test_D300_data()
```

`get_test_data` *get primary test data*

Description

get primary test data

Usage

```
get_test_data()
```

Value

object class "gdr_test_data" with with input data (manifest/template/result paths) and related reference data (qs file paths)

Examples

```
get_test_data()
```

`get_test_EnVision_data` *get test EnVision data*

Description

get test EnVision data

Usage

```
get_test_EnVision_data()
```

Value

list with with input data (manifest/template/result paths) and related reference data (.qs file paths)

Examples

```
get_test_EnVision_data()
```

`get_test_Tecan_data` *get test Tecan data*

Description

get test Tecan data

Usage

```
get_test_Tecan_data()
```

Value

list with with input data (manifest/template/result paths) and related reference data (qs file paths)

Examples

```
get_test_Tecan_data()
```

`get_test_tsv_data` *get test tsv data*

Description

get test tsv data

Usage

```
get_test_tsv_data()
```

Value

list with with input data (manifest/template/result paths) and related reference data (.qs file paths)

Examples

```
get_test_tsv_data()
```

`get_xl_sheets` *Get Excel sheets*

Description

get sheets for given set of XLS files

Usage

```
get_xl_sheets(files)
```

Arguments

`files`	charvec with file paths

Value

named list where names are the excel filenames and the values are the sheets within each file `import_D300` *Import D300*

Description

This functions takes a D300 file and generates corresponding template files

Usage

```
import_D300(D300_file, metadata_file, destination_path)
```

Arguments

`D300_file`	character, file path to D300 file
`metadata_file`	character, file path to file with mapping from D300 names to Gnumbers
`destination_path`	character, path to folder where template files will be generated

Details

For example, wells treated with 2 drugs in combination will result in 4 sheets per plate.

- Sheet 1: Drug 1
- Sheet 2: Conc of Drug 1
- Sheet 3: Drug 2
- Sheet 4: Conc of Drug 2

Value

Create one Excel file per plate. Each sheet in each plate file describes the drugs and corrresponding concentrations of what was tested in each well.

Examples

```
td3 <- get_test_D300_data()[["f_96w"]]
o_path <- file.path(tempdir(), "td3")
dir.create(o_path)
import_D300(td3$d300, td3$Gnum, o_path)
list.files(o_path)
unlink(o_path, recursive = TRUE)
```

`is_readable_v` *is_readable_v Check if all paths in vector are readable*

Description

is_readable_v Check if all paths in vector are readable

Usage

```
is_readable_v(paths)
```

Arguments

`paths`	a character with path(s)

Value

`NULL` invisibly.

Examples

```
td2 <- get_test_Tecan_data()
is_readable_v(td2$r_files)
```

`load_data` *Load data*

Description

This functions loads and checks the data file(s)

Usage

```
load_data(
  manifest_file,
  df_template_files,
  results_file,
  instrument = "EnVision"
)
```

Arguments

`manifest_file`	character, file path(s) to manifest(s)
`df_template_files`	data.table, with datapaths and names of results file(s) or character with file path of templates file(s)
`results_file`	data.table, with datapaths and names of results file(s) or character with file path of results file(s)
`instrument`	character

Value

a list with three data.tables for manifest/treatment and results

Examples

```
td <- get_test_data()
l_tbl <- load_data(manifest_path(td), template_path(td), result_path(td))
```

`load_manifest` *Load manifest*

Description

This functions loads and checks the manifest file(s)

Usage

```
load_manifest(manifest_file)
```

Arguments

`manifest_file`	character, file path(s) to manifest(s)

Value

list with manifest data.table and headers

Examples

```
td <- get_test_data()
ml <- load_manifest(manifest_path(td))
```

`load_results` *Load results*

Description

This functions loads and checks the results file(s)

Usage

```
load_results(
  df_results_files,
  instrument = "EnVision",
  headers = gDRutils::get_env_identifiers()
)
```

Arguments

`df_results_files`	data.table, with datapaths and names of results file(s) or character with file path of results file(s)
`instrument`	character
`headers`	list of headers identified in the manifest file

Value

data.table with results' data

Examples

```
td <- get_test_data()
r_df <- load_results(result_path(td))
```

`load_results_EnVision` *Load EnVision results from xlsx*

Description

This functions loads and checks the results file(s)

Usage

```
load_results_EnVision(results_file, headers = gDRutils::get_env_identifiers())
```

Arguments

`results_file`	character vector containing file path(s) to results file(s)
`headers`	list of headers identified in the manifest

Value

data.table with results data

`load_results_Tecan` *Load tecan results from xlsx*

Description

This functions loads and checks the results file

Usage

```
load_results_Tecan(results_file, headers = gDRutils::get_env_identifiers())
```

Arguments

`results_file`	string, file path to a result file
`headers`	list of headers identified in the manifest

Value

data.table derived from Tecan data

`load_results_tsv` *Load results from tsv*

Description

This functions loads and checks the results file(s)

Usage

```
load_results_tsv(results_file, headers)
```

Arguments

`results_file`	character, file path(s) to template(s)
`headers`	list of headers identified in the manifest

Value

data.table with results data

`load_templates` *Load templates*

Description

This functions loads and checks the template file(s)

Usage

```
load_templates(df_template_files)
```

Arguments

`df_template_files`	data.table, with datapaths and names of results file(s) or character with file path of templates file(s)

Value

data.table with templates data

Examples

```
td <- get_test_data()
t_df <- load_templates(template_path(td))
```

`load_templates_tsv` *Load templates from tsv*

Description

This functions loads and checks the template file(s)

Usage

```
load_templates_tsv(template_file, template_filename = NULL)
```

Arguments

`template_file`	character, file path(s) to template(s)
`template_filename`	character, file name(s)

Value

data.table with template data

`load_templates_xlsx` *Load templates from xlsx*

Description

This functions loads and checks the template file(s)

Usage

```
load_templates_xlsx(template_file, template_filename = NULL)
```

Arguments

`template_file`	character, file path(s) to template(s)
`template_filename`	character, file name(s)

Value

data.table with templates data

`manifest_path` *Method manifest_path*

Description

Method for object gdr_test_data - access to slot `manifest_path`

Usage

```
manifest_path(x)

## S4 method for signature 'gdr_test_data'
manifest_path(x)
```

Arguments

`x`	object class gdr_test_data

Value

value of slot `manifest_path`

Examples

```
td <- get_test_data()
manifest_file_path <- manifest_path(td)
```

`mgrepl` *grep wrapper to support multiple patterns*

Description

grep wrapper to support multiple patterns

Usage

```
mgrepl(patterns, x, do_unlist = TRUE, ...)
```

Arguments

`patterns`	charvec with patterns to be checked
`x`	charvec with data
`do_unlist`	logical_flag unlist the final results?
`...`	additional argument

Value

list of charvec with grep output

`parse_D300_xml` *Parse D300*

Description

This function parses a D300 *.tdd file (XML format) into a data.table

Usage

```
parse_D300_xml(D300_file)
```

Arguments

`D300_file`	string, file path to D300 .tdd file

Value

data.table representing input `D300_file`.

Examples

```
td3 <- get_test_D300_data()
fs <- td3[["f_96w"]]
dose_df <- parse_D300_xml(fs[["d300"]])
```

`read_EnVision_delim` *Read EnVision delimited text files*

Description

This function reads file from the EnVision Workstation

Usage

```
read_EnVision_delim(file, nrows = 10000, seps = c(",", "\t"))
```

Arguments

`file`	string to path of input file from EnVision scanner
`nrows`	maximum number of file rows to be processed
`seps`	potential field separators of the input file

Value

a list containing the data table, n_col, n_row, and if is edited

---

`read_EnVision_xlsx` *Read in single xlsx data from EnVision*

Description

Read in single xlsx data from EnVision

Usage

```
read_EnVision_xlsx(results_file, results_sheet)
```

Arguments

`results_file`	character, file path(s) to results file(s)
`results_sheet`	results sheet names

Value

data.table with results data

`read_excel_to_dt` *Read excel file and transorm it into data.table object*

Description

Read excel file and transorm it into data.table object

Usage

```
read_excel_to_dt(path, ...)
```

Arguments

`path`	path to excel file
`...`	other arguments that should be passed into `readxl::read_excel`

Value

data.table object with read excel file

Examples

```
datasets <- readxl::readxl_example("datasets.xlsx")
read_excel_to_dt(datasets)
```

`read_in_EnVision_file` *Read EnVision file*

Description

This function reads file from the EnVision Workstation

Usage

```
read_in_EnVision_file(file, nrows, seps)
```

Arguments

`file`	input file from EnVision
`nrows`	maximum number of file rows to be processed
`seps`	potential field separators of the input file

Value

list with one element per EnVisoin input file

`read_in_manifest_file` *read manifest files*

Description

read manifest files

Usage

```
read_in_manifest_file(manifest_file, available_formats)
```

Arguments

`manifest_file`	character, file path(s) to manifest(s)
`available_formats`	charvec with available file formats

Value

a data.table with manifest data

`read_in_results_Tecan` *read in Tecan data*

Description

read in Tecan data

Usage

```
read_in_results_Tecan(results_file, results_sheets, headers)
```

Arguments

`results_file`	string, file path to a result file
`results_sheets`	template sheet names
`headers`	list of headers identified in the manifest

Value

data.table derived from Tecan data

`read_in_result_files` *Read in results files*

Description

Read in results files

Usage

```
read_in_result_files(results_file, results_filename, headers)
```

Arguments

`results_file`	data.table, with datapaths and names of results file(s) or character with file path of results file(s)
`results_filename`	charvect with file names
`headers`	list of headers identified in the result files

Value

data.table with results data

`read_in_template_sheet_xlsx` *Read in data from xlsx template sheet*

Description

Read in data from xlsx template sheet

Usage

```
read_in_template_sheet_xlsx(template_file, template_sheets, idx, plate_info)
```

Arguments

`template_file`	character, file path(s) to template(s)
`template_sheets`	template sheet names
`idx`	template file index
`plate_info`	list with plate info

Value

data.table with template data

`read_in_template_xlsx` *Read in xlsx template files*

Description

Read in xlsx template files

Usage

```
read_in_template_xlsx(template_file, template_filename, template_sheets)
```

Arguments

`template_file`	character, file path(s) to template(s)
`template_filename`	character, file name(s)
`template_sheets`	template sheet names

Value

data.table with templates data

`read_in_tsv_template_files` *read in tsv template files*

Description

read in tsv template files

Usage

```
read_in_tsv_template_files(template_file, template_filename, templates)
```

Arguments

`template_file`	character, file path(s) to template(s)
`template_filename`	character, file name(s)
`templates`	list with templates data

Value

data.table with templates data

`read_ref_data` *read_ref_data*

Description

Read reference data

Usage

```
read_ref_data(inDir, prefix = "ref")
```

Arguments

`inDir`	a directory path of reference data
`prefix`	a prefix of reference file names ('ref' by default)

Value

a list of reference data

`result_path` *Method result_path*

Description

Method for object gdr_test_data - access to slot `result_path`

Usage

```
result_path(x)

## S4 method for signature 'gdr_test_data'
result_path(x)
```

Arguments

`x`	object class gdr_test_data

Value

value of slot `result_path`

Examples

```
td <- get_test_data()
result_file_path <- result_path(td)
```

`save_drug_info_per_well` *for each drug create a Gnumber and Concentration information for each well*

Description

for each drug create a Gnumber and Concentration information for each well

Usage

```
save_drug_info_per_well(trt_info, trt_gnumber_conc, wb, idfs)
```

Arguments

`trt_info`	list with treatment info
`trt_gnumber_conc`	list with treatment data
`wb`	pointer to xlsx workbook
`idfs`	charvec with identifiers

Value

`NULL` invisibly.

`setEnvForPSet` *Adjust environment variables to meet gDR standards*

Description

Adjust environment variables to meet gDR standards

Usage

```
setEnvForPSet()
```

Value

`NULL`

Examples

```
setEnvForPSet()
gDRutils::reset_env_identifiers()
```

`standardize_record_values` *standardize_record_values*

Description

map values to a dictionary

Usage

```
standardize_record_values(x, dictionary = DICTIONARY)
```

Arguments

`x`	a named array
`dictionary`	a named array

Value

a named array with updated names

Examples

```
standardize_record_values(c("Vehicle", "vehcle"))
```

`template_path` *Method template_path*

Description

Method for object gdr_test_data - access to slot `template_path`

Usage

```
template_path(x)

## S4 method for signature 'gdr_test_data'
template_path(x)
```

Arguments

`x`	object class gdr_test_data

Value

value of slot `template_path`

Examples

```
td <- get_test_data()
template_file_path <- template_path(td)
```

`validate_template_xlsx` *Validate template xlsx data*

Description

Validate template xlsx data

Usage

```
validate_template_xlsx(template_file, template_filename, template_sheets, idx)
```

Arguments

`template_file`	character, file path(s) to template(s)
`template_filename`	character, file name(s)
`template_sheets`	template sheet names
`idx`	template file index

Value

`NULL` invisibly.

Index