

Package: gDRimport (via r-universe)

October 26, 2026

Type Package

Title Package for handling the import of dose-response data

Version 1.11.13

Date 2026-10-25

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, data.table, futile.logger, gDRutils (>= 1.7.1), magrittr, methods, MultiAssayExperiment, readxl, rio, S4Vectors, stats, stringi, SummarizedExperiment, tibble, tools, utils, XML, yaml, openxlsx, qs2

Suggests BiocStyle, CoreGx, gDRtestData (>= 1.7.1), gDRstyle (>= 1.7.1), knitr, PharmacGx, purrr, 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)

SwitchrLibrary gDRimport

DeploySubPath gDRimport

Encoding UTF-8

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.1

Config/pak/sysreqs cmake make libicu-dev libxml2-dev libssl-dev libnode-dev libx11-dev zlib1g-dev

Repository <https://bioc.r-universe.dev>

Date/Publication 2026-10-26 10:22:12 UTC

RemoteUrl <https://github.com/bioc/gDRimport>

RemoteRef HEAD

RemoteSha 4ec3d8c256935769b18aa619cadb179983d8d195

Contents

check_metadata_names	3
convert_LEVEL5_prism_to_gDR_input	3
convert_LEVEL6_prism_to_gDR_input	4
convert_MAE_to_PSet	5
convert_pset_to_df	6
detect_file_format	7
fix_typos_with_reference	7
gdr_test_data-class	8
get_df_from_raw_edited_EnVision_df	9
get_df_from_raw_unedited_EnVision_df	10
get_exception_data	10
get_expected_template_sheets	11
get_import_provenance	11
get_test_D300_data	12
get_test_data	12
get_test_EnVision_data	13
get_test_Tecan_data	13
get_test_tsv_data	14
getPSet	14
import_D300	15
is_readable_v	16
load_data	16
load_long_table	17
load_manifest	18
load_results	19
load_results_EnVision	19
load_results_Incucyte	20
load_results_Tecan	20
load_results_tsv	21
load_templates	21
load_templates_tsv	22
load_templates_xlsx	22
parse_D300_xml	23
read_excel_to_dt	23
save_drug_info_per_well	24
setEnvForPSet	24

<i>check_metadata_names</i>	3
-----------------------------	---

standardize_record_values	25
validate_gdr_input	25

Index	28
--------------	-----------

check_metadata_names	<i>check_metadata_names</i>
----------------------	-----------------------------

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))
```

convert_LEVEL5_prism_to_gDR_input
<i>Load, convert and process the level 5 PRISM data into a gDR input</i>

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	<i>Convert MultiAssayExperiment to TreatmentResponseExperiment</i>
---------------------	--

Description

This function converts a MultiAssayExperiment generated by gDR into a TreatmentResponseExperiment for use in the PharmacoSx package. The resulting PharmacoSx 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 PharmacoSx object.

Value

A PharmacoSx object.

Examples

```
# Convert a MultiAssayExperiment object to a PharmacoSx 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	<i>Convert a PharmacoSet to a data.table that is prepare for input into gDR pipeline</i>
--------------------	--

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()
```

detect_file_format	<i>Detect format of results data</i>
--------------------	--------------------------------------

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])
```

fix_typos_with_reference	<i>Fix typos using reference data</i>
--------------------------	---------------------------------------

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 identifiers?

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

`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

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

Value

data.table derived from EnVision data

get_df_from_raw_unedited_EnVision_df
<i>Get final results (as a data.table) from raw unedited EnVision data.table</i>

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_exception_data	<i>get exception data</i>
--------------------	---------------------------

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_import_provenance *Where gDR input data came from*

Description

[load_data](#) and the PRISM converters attach a record of their inputs to the table they return: the gDRimport version, the reader (instrument), every input file with its role, size and md5, and the corrections gDRimport applied on its own. gDRcore keeps the record in the provenance of the processed object (provenance\$steps\$import).

Usage

```
get_import_provenance(x)
```

Arguments

x the data element returned by [load_data](#), or a table returned by a PRISM converter.

Value

The record (a list), or NULL when x carries none.

Examples

```
td <- get_test_data()
l_data <- load_data(manifest_path(td), template_path(td), result_path(td))
get_import_provenance(l_data$data)$files
```

get_test_D300_data	<i>get test D300 data</i>
--------------------	---------------------------

Description

get test D300 data

Usage

```
get_test_D300_data()
```

Value

list with with input data (manifest/template/result paths) and related reference data (qs2 file paths)

Examples

```
get_test_D300_data()
```

get_test_data	<i>get primary test data</i>
---------------	------------------------------

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 (qs2 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 (.qs2 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 (qs2 file paths)

Examples

`get_test_Tecan_data()`

get_test_tsv_data	<i>get test tsv data</i>
-------------------	--------------------------

Description

get test tsv data

Usage

get_test_tsv_data()

Value

list with with input data (manifest/template/result paths) and related reference data (.qs2 file paths)

Examples

get_test_tsv_data()

getPSet	<i>Get PharmacoSet</i>
---------	------------------------

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
))
```

import_D300

Import D300

Description

This functions takes a D300 file and generates corresponding template files

Usage

```
import_D300(D300_file, destination_path, metadata_file = NULL, day0 = FALSE)
```

Arguments

D300_file	character, file path to D300 file
destination_path	character, path to folder where template files will be generated
metadata_file	character, file path to file with mapping from D300 names to Gnumbers. Defaults to NULL.
day0	logical, if TRUE, creates a template file for Day 0 data filled with vehicles in addition to the standard plates. Defaults to FALSE.

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 corresponding 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$d300, o_path, td3$Gnum)
list.files(o_path)
unlink(o_path, recursive = TRUE)
```

is_readable_v	<i>is_readable_v</i> 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	<i>Load data</i>
-----------	------------------

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_long_table	<i>Load long table</i>
-----------------	------------------------

Description

Loads a long/tidy data.table from a delimited text file (CSV/TSV) and checks that it contains the columns required by the gDR pipeline. This offers an input path that does not rely on Gene-DataScreenR: the data can come from a custom export or be provided directly by the user in tabular form, matching the shape of the data_imported object used downstream.

Usage

```
load_long_table(long_table_file)
```

Arguments

long_table_file character, path to a single CSV/TSV file with the long table. The field separator is detected automatically.

Details

Required columns follow the current gDR identifiers: Gnumber, clid, Duration and Concentration. The readout column is the fixed header ReadoutValue: there is no "readout" identifier in gDRutils, so this name is not customisable via get_env_identifiers(), matching how ReadoutValue is used across the rest of gDRimport. Combination data may add Gnumber_2 and Concentration_2. Duration, Concentration and ReadoutValue must be numeric.

Value

a `data.table` with the validated long table

Examples

```
path <- tempfile(fileext = ".csv")
dt <- data.table::data.table(Gnumber = "G1", clid = "CL1", Duration = 72,
                             Concentration = 1, ReadoutValue = 1000)
data.table::fwrite(dt, path)
load_long_table(path)
```

load_manifest	<i>Load manifest</i>
---------------	----------------------

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	<i>Load results</i>
--------------	---------------------

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; .tsv files with "EnVision" are read as "long_tsv", whether it is passed or left as the default
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	<i>Load EnVision results from xlsx</i>
-----------------------	--

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_Incucyte *Load incucyte results from plain text*

Description

This functions loads incucyte time-course cell count file

Usage

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

Arguments

results_file list of strings: file paths to result paths from individual plates
 headers list of headers identified in the manifest

Value

data.table derived from Incucyte 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	<i>Load results from tsv</i>
------------------	------------------------------

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	<i>Load templates</i>
----------------	-----------------------

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	<i>Load templates from tsv</i>
--------------------	--------------------------------

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	<i>Load templates from xlsx</i>
---------------------	---------------------------------

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

parse_D300_xml	<i>Parse D300</i>
----------------	-------------------

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_excel_to_dt	<i>Read excel file and transform it into data.table object</i>
------------------	--

Description

Read excel file and transform 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)
```

```
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
<i>standardize_record_values</i>

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"))

validate_gdr_input	<i>Validate gDR input data before import</i>
--------------------	--

Description

Checks a set of input files and reports every problem it finds, from unreadable files to plate-level anomalies in the readouts. Bench input (a manifest, treatment templates and raw plate-reader files) is loaded with [load_data](#), so a failure reported for load_data is the error the import itself would raise. The checks also flag problems the loaders let through without an error: values in a Concentration sheet that are not numbers (those wells are dropped), raw plates without a barcode, duplicated wells, plates missing from the manifest or from the data, and Duration labels that do not fit the readouts of the control wells. PRISM input, processed MultiAssayExperiments and D300 files are recognised and checked too.

Usage

```
validate_gdr_input(
  paths,
  type = c("auto", "bench", "prism", "mae", "pre_d300"),
  instrument = NULL,
  id_patterns = list(drug = "^G[0-9]{8}", cellline = NULL),
  fix = FALSE,
  output_dir = NULL
)

## S3 method for class 'gdr_validation'
print(x, ...)
```

Arguments

paths	character, paths to input files and/or directories. For a directory, the files directly inside it are used, not those in its subdirectories.
type	character, the kind of input: "auto" (detected from the files), "bench" (manifest, templates and raw plate-reader files), "prism", "mae" (a processed gDR MultiAssayExperiment saved as .qs2) or "pre_d300" (D300 .tdd files that still need <code>import_D300</code>).
instrument	character, the plate-reader format passed to <code>load_data</code> . When NULL, it is detected from the first raw file with <code>detect_file_format</code> .
id_patterns	list of regular expressions for identifier formats. drug is the format of a drug identifier once its batch suffix is removed (the default is the one <code>gDRutils::remove_drug_batch</code> uses), and cellline the format of a cell line identifier (no default). An element set to NULL switches its check off; elements left out keep their default.
fix	logical, whether to repair copies of the files in output_dir.
output_dir	character, a new or empty directory for the repaired copies; required with fix = TRUE.
x	a <code>gdr_validation</code> object.
...	not used.

Details

The input files are only read, never modified. With `fix = TRUE` they are copied to `output_dir`, the mechanical problems (fix tier "A") are repaired in the copies, and the copies are validated again: files re-encoded as UTF-8, spaces trimmed from file names, long tables saved as .tsv, Gnumber_1 and Concentration_1 sheets renamed, n/a and -99 cleared from templates, and manifest references that only differ from a template file by spaces or the extension rewritten. In `xlsx` files only the repaired cells change.

Value

An object of class `gdr_validation`, a list with

verdict "PASS", "WARN" or "FAIL", the most severe finding ("INFO" counts as "PASS").

data_type the kind of input that was validated.

files a `data.table` with one row per input file: file, path, role and format (the plate-reader format of raw files).

findings a `data.table` with one row per finding: check, severity ("PASS", "INFO", "WARN" or "FAIL"), detail, fix_tier ("A" a mechanical fix, "B" a fix to confirm with whoever made the files, "C" a decision about the experiment, NA when there is nothing to fix) and status_code, the matching row of `get_exception_data` (NA when there is none).

fixes character, one note per repair (empty without fix).

output_dir the directory of the repaired copies, NA without fix. With `fix = TRUE`, files and findings describe the copies after the repairs.

Examples

```
td <- get_test_data()
res <- validate_gdr_input(c(manifest_path(td), template_path(td), result_path(td)))
res$verdict
res$findings

# repair copies of the files instead of only reporting
fixed <- validate_gdr_input(c(manifest_path(td), template_path(td), result_path(td)),
                           fix = TRUE, output_dir = tempfile())
fixed$fixes
```

Index

- * **D300**
 - import_D300, [15](#)
 - parse_D300_xml, [23](#)
 - save_drug_info_per_well, [24](#)
- * **classes**
 - gdr_test_data-class, [8](#)
- * **correction_exception**
 - check_metadata_names, [3](#)
 - fix_typos_with_reference, [7](#)
 - get_exception_data, [10](#)
 - get_expected_template_sheets, [11](#)
 - validate_gdr_input, [25](#)
- * **load_files**
 - get_df_from_raw_edited_EnVision_df, [9](#)
 - get_df_from_raw_unedited_EnVision_df, [10](#)
 - get_import_provenance, [11](#)
 - load_data, [16](#)
 - load_long_table, [17](#)
 - load_manifest, [18](#)
 - load_results, [19](#)
 - load_results_EnVision, [19](#)
 - load_results_Incucyte, [20](#)
 - load_results_Tecan, [20](#)
 - load_results_tsv, [21](#)
 - load_templates, [21](#)
 - load_templates_tsv, [22](#)
 - load_templates_xlsx, [22](#)
- * **prism_conversion**
 - convert_LEVEL5_prism_to_gDR_input, [3](#)
 - convert_LEVEL6_prism_to_gDR_input, [4](#)
- * **pset_conversion**
 - convert_MAE_to_PSet, [5](#)
 - convert_pset_to_df, [6](#)
 - getPSet, [14](#)
 - setEnvForPSet, [24](#)
- * **test_data_class**
 - gdr_test_data-class, [8](#)
 - get_test_data, [12](#)
- * **test_data**
 - get_test_D300_data, [12](#)
 - get_test_EnVision_data, [13](#)
 - get_test_Tecan_data, [13](#)
 - get_test_tsv_data, [14](#)
- * **utils**
 - detect_file_format, [7](#)
 - is_readable_v, [16](#)
 - read_excel_to_dt, [23](#)
 - standardize_record_values, [25](#)
- check_metadata_names, [3](#)
- convert_LEVEL5_prism_to_gDR_input, [3](#)
- convert_LEVEL6_prism_to_gDR_input, [4](#)
- convert_MAE_to_PSet, [5](#)
- convert_pset_to_df, [6](#)
- detect_file_format, [7](#), [26](#)
- fix_typos_with_reference, [7](#)
- gdr_test_data-class, [8](#)
- get_df_from_raw_edited_EnVision_df, [9](#)
- get_df_from_raw_unedited_EnVision_df, [10](#)
- get_exception_data, [10](#), [27](#)
- get_expected_template_sheets, [11](#)
- get_import_provenance, [11](#)
- get_test_D300_data, [12](#)
- get_test_data, [12](#)
- get_test_data(), [8](#)
- get_test_EnVision_data, [13](#)
- get_test_Tecan_data, [13](#)
- get_test_tsv_data, [14](#)
- getPSet, [14](#)
- import_D300, [15](#), [26](#)
- is_readable_v, [16](#)

- load_data, [11](#), [16](#), [25](#), [26](#)
- load_long_table, [17](#)
- load_manifest, [18](#)
- load_results, [19](#)
- load_results_EnVision, [19](#)
- load_results_Incucyte, [20](#)
- load_results_Tecan, [20](#)
- load_results_tsv, [21](#)
- load_templates, [21](#)
- load_templates_tsv, [22](#)
- load_templates_xlsx, [22](#)

- parse_D300_xml, [23](#)
- print.gdr_validation
 - (validate_gdr_input), [25](#)

- read_excel_to_dt, [23](#)

- save_drug_info_per_well, [24](#)
- setEnvForPSet, [24](#)
- standardize_record_values, [25](#)

- validate_gdr_input, [25](#)