Package 'DrugExposureDiagnostics'

July 26, 2026

Title Diagnostics for OMOP Common Data Model Drug Records

Version 1.1.10

Description Ingredient specific diagnostics for drug exposure records in the Observational Medical Outcomes Partnership (OMOP) common data model.

License Apache License (>= 2)

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.0)

Imports CDMConnector (>= 1.4.0), dplyr (>= 1.0.0), magrittr (>= 2.0.0), rlang (>= 1.0.0), tidyr (>= 1.2.0), tidyselect (>= 1.2.0), checkmate (>= 2.0.0), glue (>= 1.5.0), DrugUtilisation (>= 0.7.0), omopgenerics (>= 0.2.3), R6

Suggests testthat (>= 3.0.0), duckdb, odbc, DBI, knitr, rmarkdown, zip, lubridate, tibble, DT, graphics, SqlRender, ggplot2, plotly, tictoc, here, shiny (>= 1.6.0), shinycssloaders, shinyWidgets, shinyjs, shinytest2

Config/testthat/edition 3

URL https://darwin-eu.github.io/DrugExposureDiagnostics/, https://github.com/darwin-eu/DrugExposureDiagnostics

BugReports https://github.com/darwin-eu/DrugExposureDiagnostics/issues

VignetteBuilder knitr

Language en-US

Collate 'DrugExposureDiagnostics-package.R' 'checkDaysSupply.R' 'checkDrugDose.R' 'checkDrugExposureDuration.R' 'checkDrugQuantity.R' 'checkDrugRoutes.R' 'checkDrugSig.R' 'checkDrugSourceConcepts.R' 'checkDrugType.R' 'checkDrugsMissing.R' 'checkTimeBetween.R' 'checkVerbatimEndDate.R' 'shinyModule.R' 'dataPlotPanel.R' 'executeChecks.R' 'getDrugRecords.R' 'globals.R' 'ingredientDescendantsInDb.R' 'metaDataPanel.R' 'mockDrugExposure.R' 'obscureCounts.R' 'runBenchmark.R'

'shiny.R' 'shinyApp.R' 'summariseChecks.R' 'utils-pipe.R'
'utils.R'

NeedsCompilation no

Author Ger Inberg [aut, cre] (ORCID: <https://orcid.org/0000-0001-8993-8748>),
Edward Burn [aut] (ORCID: <https://orcid.org/0000-0002-9286-1128>),
Theresa Burkard [aut] (ORCID: <https://orcid.org/0000-0003-1313-4473>),
Yuchen Guo [ctb] (ORCID: <https://orcid.org/0000-0002-0847-4855>),
Marti Catala [ctb] (ORCID: <https://orcid.org/0000-0003-3308-9905>),
Mike Du [ctb] (ORCID: <https://orcid.org/0000-0002-9517-8834>),
Xintong Li [ctb] (ORCID: <https://orcid.org/0000-0002-6872-5804>),
Ross Williams [ctb] (ORCID: <https://orcid.org/0000-0001-7723-417X>),
Erasmus MC [cph]

Maintainer Ger Inberg <g.inberg@erasmusmc.nl>

Repository CRAN

Date/Publication 2026-07-26 16:20:02 UTC

Contents

checkDaysSupply . . . 3
checkDbType . . . 4
checkDrugDose . . . 4
checkDrugSig . . . 5
checkIngredientInTable . . . 5
checkIsIngredient . . . 6
checkLogical . . . 6
checkSampleMinCellCount . . . 7
checkTableExists . . . 7
checkVerbatimEndDate . . . 8
computeDBQuery . . . 8
dataPlotPanel . . . 9
executeChecks . . . 10
executeChecksSingleIngredient . . . 12
getAllCheckOptions . . . 13
getDrugMissings . . . 14
getDrugRecords . . . 14
getDrugRoutes . . . 15
getDrugSourceConcepts . . . 16
getDrugTypes . . . 16
getDuration . . . 17
ingredientDescendantsInDb . . . 18
mem_change . . . 18
mem_used . . . 19
metaDataPanel . . . 19
mockDrugExposure . . . 21
obscureCounts . . . 22
printDurationAndMessage . . . 23

runBenchmarkExecuteSingleIngredient 23
ShinyApp 24
ShinyModule 25
summariseChecks 27
summariseDrugExposureDuration 27
summariseQuantity 28
summariseTimeBetween 29
validateChecks 29
viewResults 30
writeFile 30
writeIngredientResultToDisk 31
writeResultToDisk 32
writeZipToDisk 33

Index **34**

`checkDaysSupply` *Check if Days_supply is the same as datediff(drug_exp_start_date,drug_exp_end_date)*

Description

Check if Days_supply is the same as datediff(drug_exp_start_date,drug_exp_end_date)

Usage

```
checkDaysSupply(
  cdm,
  drugRecordsTable = "ingredient_drug_records",
  byConcept = TRUE,
  sampleSize = 10000
)
```

Arguments

`cdm`	CDMConnector reference object
`drugRecordsTable`	a modified version of the drug exposure table, default "ingredient_drug_records"
`byConcept`	whether to get result by drug concept
`sampleSize`	the sample size given in execute checks

Value

a table with the stats of days supply compared to start and end date

checkDbType *Check the database type.*

Description

Check the database type.

Usage

```
checkDbType(cdm, type = "cdm_reference", messageStore)
```

Arguments

cdm	CDMConnector reference object
type	type of the database, default cdm_reference
messageStore	checkmate collection

checkDrugDose *Get a summary of the daily drug dose*

Description

Get a summary of the daily drug dose

Usage

```
checkDrugDose(cdm, ingredientConceptId, sampleSize = NULL, minCellCount = 5)
```

Arguments

cdm	CDMConnector reference object
ingredientConceptId	ingredient
sampleSize	Maximum number of records of an ingredient to estimate dose coverage. If an ingredient has more, a random sample equal to sampleSize will be considered. If NULL, all records will be used.
minCellCount	minimum number of events to report- results lower than this will be obscured. If NULL all results will be reported.

Value

a table with the stats about the daily dose

`checkDrugSig` *Check the drug sig field; this is the verbatim instruction for the drug as written by the provider.*

Description

Check the drug sig field; this is the verbatim instruction for the drug as written by the provider.

Usage

```
checkDrugSig(
  cdm,
  drugRecordsTable = "ingredient_drug_records",
  byConcept = TRUE,
  sampleSize = 10000
)
```

Arguments

`cdm`	CDMConnector reference object
`drugRecordsTable`	a modified version of the drug exposure table, default "ingredient_drug_records"
`byConcept`	whether to get result by drug concept
`sampleSize`	the sample size given in execute checks

Value

a table with a summary of the sig values

`checkIngredientInTable` *Check ingredient is present in given table*

Description

Check ingredient is present in given table

Usage

```
checkIngredientInTable(cdm, conceptId, tableName, messageStore)
```

Arguments

cdm	CDMConnector reference object
conceptId	ingredient concept id to check
tableName	name of the table to check
messageStore	checkmate collection

---

checkIsIngredient *Check is an ingredient*

Description

Check is an ingredient

Usage

```
checkIsIngredient(cdm, conceptId, messageStore)
```

Arguments

cdm	CDMConnector reference object
conceptId	ingredient concept id to check
messageStore	checkmate collection

---

checkLogical *Check if given object is a boolean.*

Description

Check if given object is a boolean.

Usage

```
checkLogical(input, messageStore, null.ok = TRUE)
```

Arguments

input	the input
messageStore	checkmate collection
null.ok	if value null is allowed

checkSampleMinCellCount *Check that the sample is bigger than the mincellcount*

Description

Check that the sample is bigger than the mincellcount

Usage

```
checkSampleMinCellCount(sampleSize, minCellCount, messageStore)
```

Arguments

sampleSize	sample size for sampling
minCellCount	minimum cell count below which to obsure results
messageStore	checkmate collection

checkTableExists *Check if given table exists in cdm.*

Description

Check if given table exists in cdm.

Usage

```
checkTableExists(cdm, tableName, messageStore)
```

Arguments

cdm	CDMConnector reference object
tableName	checkmate collection
messageStore	the message store

checkVerbatimEndDate *Check the verbatim_end_date field*

Description

Check the verbatim_end_date field

Usage

```
checkVerbatimEndDate(
  cdm,
  drugRecordsTable = "ingredient_drug_records",
  byConcept = TRUE,
  sampleSize = 10000
)
```

Arguments

cdm	CDMConnector reference object
drugRecordsTable	a modified version of the drug exposure table, default "ingredient_drug_records"
byConcept	whether to get result by drug concept
sampleSize	the sample size given in execute checks

Value

a table with the stats about the verbatim_end_date

computeDBQuery *Store the given input in a remote database table. It will be stored either in a permanent table or a temporary table depending on tablePrefix.*

Description

Store the given input in a remote database table. It will be stored either in a permanent table or a temporary table depending on tablePrefix.

Usage

```
computeDBQuery(table, tablePrefix, tableName, cdm, overwrite = TRUE)
```

Arguments

table	the input table
tablePrefix	The stem for the permanent tables that will be created when running the diagnostics. Permanent tables will be created using this prefix, and any existing tables that start with this will be at risk of being dropped or overwritten. If NULL, temporary tables will be used throughout.
tableName	the input table
cdm	cdm reference object
overwrite	if the table should be overwritten (default TRUE).

Value

reference to the table

---

dataPlotPanel *dataPlotPanel*

Description

Class to view the data and plot view of a DrugExposureDiagnostics check.

Value

self

Super class

DrugExposureDiagnostics::ShinyModule -> dataPlotPanel

Methods

Public methods:

- dataPlotPanel$new()
- dataPlotPanel$uiBody()
- dataPlotPanel$clone()

Method new(): Method to handle the back-end.

Initializer method

Usage:

```
dataPlotPanel$new(
  data,
  dataByConcept = NULL,
  id,
  title,
  description,
```

```
  plotPercentage,
  byConcept,
  downloadFilename,
  selectedColumns = colnames(data)
)
```

Arguments:

`data` data from the `DrugExposureDiagnostics` package.

`dataByConcept` data by drug concept

`id` the unique reference id for the module

`title` panel title

`description` description of data table

`plotPercentage` if plot by percentage should be enabled

`byConcept` add byConcept switch

`downloadFilename` filename of the downloaded file

`selectedColumns` default selected columns

`input (input)`
: Input from the server function.

`output (output)`
: Output from the server function.

`session (session)`
: Session from the server function.

Returns: `(NULL)`

Method `uiBody()`**:** Method to include a tabPanel to include the body.

Usage:

```
dataPlotPanel$uiBody()
```

Returns: `(tabItem)`

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
dataPlotPanel$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`executeChecks` *Execute given checks on Drug Exposure.*

Description

Execute given checks on Drug Exposure.

Usage

```
executeChecks(
  cdm,
  ingredients = c(1125315),
  subsetToConceptId = NULL,
  checks = c("missing", "exposureDuration", "quantity"),
  minCellCount = 5,
  sample = 10000,
  tablePrefix = NULL,
  earliestStartDate = "2010-01-01",
  verbose = FALSE,
  byConcept = TRUE,
  exposureTypeId = NULL,
  outputFolder = NULL,
  databaseId = CDMConnector::cdmName(cdm),
  filename = NULL
)
```

Arguments

cdm	CDMConnector reference object
ingredients	vector of ingredients, by default: acetaminophen
subsetToConceptId	vector of concept IDs of the ingredients to filter. If a concept ID is positive it will be included, a negative one will be excluded. If NULL, all concept IDs for an ingredient will be considered.
checks	the checks to be executed, by default the missing values, the exposure duration and the quantity. Possible options are "missing", "exposureDuration", "type", "route", "sourceConcept", "daysSupply", "verbatimEndDate", "dose", "sig", "quantity", "daysBetween" and "diagnosticsSummary"
minCellCount	minimum number of events to report- results lower than this will be obscured. If 0 all results will be reported.
sample	the number of samples, default 10.000
tablePrefix	The stem for the permanent tables that will be created when running the diagnostics. Permanent tables will be created using this prefix, and any existing tables that start with this will be at risk of being dropped or overwritten. If NULL, temporary tables will be used throughout.
earliestStartDate	the earliest date from which a record can be included
verbose	verbose, default FALSE
byConcept	boolean argument whether to return results by Concept or overall only
exposureTypeId	id of the drug exposure type to be filtered on (e.g. only prescribed). By default all record types will be taken into account.
outputFolder	folder to write to. If NULL, results will not be written to file
databaseId	database identifier
filename	output file name, if NULL it will be equal to databaseId

Value

named list with results

Examples

```
## Not run:
db <- DBI::dbConnect(
 RPostgres::Postgres(),
 dbname = dbname,
 port = port,
 host = host,
 user = user,
 password = password,
 bigint = c("numeric")
)
cdm <- CDMConnector::cdmFromCon(
  con = db,
  cdmSchema = "cdm schema name"
)
result <- executeChecks(
  cdm = cdm,
  ingredients = c(1125315)
)

## End(Not run)
```

`executeChecksSingleIngredient` *Execute given checks on Drug Exposure for a single ingredient.*

Description

Execute given checks on Drug Exposure for a single ingredient.

Usage

```
executeChecksSingleIngredient(
  cdm,
  ingredient = 1125315,
  subsetToConceptId = NULL,
  checks = c("missing", "exposureDuration", "quantity"),
  minCellCount = 5,
  sampleSize = 10000,
  tablePrefix = NULL,
  earliestStartDate = "2010-01-01",
  verbose = FALSE,
  byConcept = FALSE,
  exposureTypeId = NULL
)
```

Arguments

cdm	CDMConnector reference object
ingredient	ingredient, by default: acetaminophen
subsetToConceptId	vector of concept IDs of the ingredients to filter. If a concept ID is positive it will be included, a negative one will be excluded. If NULL, all concept IDs for an ingredient will be considered.
checks	the checks to be executed, by default the missing values, the exposure duration and the quantity.
minCellCount	minimum number of events to report- results lower than this will be obscured. If 0 all results will be reported.
sampleSize	the number of samples, default 10.000
tablePrefix	The stem for the permanent tables that will be created when running the diagnostics. Permanent tables will be created using this prefix, and any existing tables that start with this will be at risk of being dropped or overwritten. If NULL, temporary tables will be used throughout.
earliestStartDate	the earliest date from which a record can be included
verbose	verbose, default FALSE
byConcept	boolean argument whether to return results by Concept or overall only
exposureTypeId	id of the drug exposure type to be filtered on (e.g. only prescribed). By default all record types will be taken into account.

Value

named list with results

---

getAllCheckOptions *Get all options that can be passed to the "checks" parameter*

Description

Get all options that can be passed to the "checks" parameter

Usage

```
getAllCheckOptions()
```

getDrugMissings *Check missings in drug exposure records*

Description

Check missings in drug exposure records

Usage

```
getDrugMissings(
  cdm,
  drugRecordsTable = "ingredient_drug_records",
  byConcept = TRUE,
  sampleSize = 10000
)
```

Arguments

cdm	CDMConnector reference object
drugRecordsTable	a modified version of the drug exposure table, default "ingredient_drug_records"
byConcept	by individual drug Concept
sampleSize	the sample size given in execute checks

Value

a table with a summary of missing records

getDrugRecords *Drug exposure records for ingredients of interest*

Description

Drug exposure records for ingredients of interest

Usage

```
getDrugRecords(
  cdm,
  ingredient,
  includedConceptsTable,
  drugRecordsTable = "drug_exposure",
  exposureTypeId = NULL,
  tablePrefix = NULL,
  verbose = FALSE
)
```

Arguments

cdm	CDMConnector reference object
ingredient	Concept ID for ingredient of interest
includedConceptsTable	includedConceptsTable
drugRecordsTable	drugRecordsTable, default "drug_exposure"
exposureTypeId	id of the drug exposure type to be filtered on (e.g. only prescribed). By default all record types will be taken into account.
tablePrefix	The stem for the permanent tables that will be created when running the diagnostics. Permanent tables will be created using this prefix, and any existing tables that start with this will be at risk of being dropped or overwritten. If NULL, temporary tables will be used throughout.
verbose	verbose

Value

a table containing drug exposure records

---

getDrugRoutes *Get drug exposure route types*

Description

Get drug exposure route types

Usage

```
getDrugRoutes(
  cdm,
  drugRecordsTable = "ingredient_drug_records",
  byConcept = TRUE,
  sampleSize = 10000
)
```

Arguments

cdm	CDMConnector reference object
drugRecordsTable	a modified version of the drug exposure table, default "ingredient_drug_records"
byConcept	by individual drug Concept
sampleSize	the sample size given in execute checks

Value

a table with the drug exposure route types

getDrugSourceConcepts *Check drug exposure source types*

Description

Check drug exposure source types

Usage

```
getDrugSourceConcepts(
  cdm,
  drugRecordsTable = "ingredient_drug_records",
  sampleSize = 10000
)
```

Arguments

cdm	CDMConnector reference object
drugRecordsTable	modified drug exposure table
sampleSize	the sample size given in execute checks

Value

a table with the drug source concepts

getDrugTypes *Get drug exposure record types*

Description

Get drug exposure record types

Usage

```
getDrugTypes(
  cdm,
  drugRecordsTable = "ingredient_drug_records",
  byConcept = TRUE,
  sampleSize = 10000
)
```

Arguments

cdm	CDMConnector reference object
drugRecordsTable	a modified version of the drug exposure table, default "ingredient_drug_records"
byConcept	by individual drug Concept
sampleSize	the sample size given in execute checks

Value

a table with the drug exposure record types

---

getDuration *Compute the difference in days between 2 variables in a database table.*

Description

Compute the difference in days between 2 variables in a database table.

Usage

```
getDuration(
  cdm,
  tableName = "drug_exposure",
  startDateCol = "drug_exposure_start_date",
  endDateCol = "drug_exposure_end_date",
  colName = "duration"
)
```

Arguments

cdm	CDMConnector reference object
tableName	the table name
startDateCol	the start date column name
endDateCol	the end date column name
colName	the result column name

Value

the table with as new column the duration

ingredientDescendantsInDb

Get the descendants for the given ingredients

Description

Get the descendants for the given ingredients

Usage

```
ingredientDescendantsInDb(
  cdm,
  ingredient,
  drugRecordsTable = "drug_exposure",
  tablePrefix = NULL,
  verbose = FALSE
)
```

Arguments

cdm	CDMConnector reference object
ingredient	ingredient concept id for ingredient of interest
drugRecordsTable	table name of the drug exposure records, default "drug_exposure"
tablePrefix	The stem for the permanent tables that will be created when running the diagnostics. Permanent tables will be created using this prefix, and any existing tables that start with this will be at risk of being dropped or overwritten. If NULL, temporary tables will be used throughout.
verbose	if verbose set to TRUE, the function will output extra messages

Value

temp table with concepts used

mem_change

Determine change in memory from running code

Description

Determine change in memory from running code

Usage

```
mem_change(code)
```

Arguments

code	Code to evaluate.

Value

Change in memory (in megabytes) before and after running code.

---

mem_used *How much memory is currently used by R?*

Description

R breaks down memory usage into Vcells (memory used by vectors) and Ncells (memory used by everything else). However, neither this distinction nor the "gc trigger" and "max used" columns are typically important. What we're usually most interested in is the the first column: the total memory used. This function wraps around `gc()` to return the total amount of memory (in megabytes) currently used by R.

Usage

```
mem_used()
```

Value

Megabytes of ram used by R objects.

---

metaDataPanel *metaDataPanel*

Description

Class to view the metadata of a DrugExposureDiagnostics execution.

Value

`self`

Super class

`DrugExposureDiagnostics::ShinyModule` -> `metaDataPanel`

Methods

Public methods:

- metaDataPanel$new()
- metaDataPanel$uiBody()
- metaDataPanel$clone()

Method `new()`**:** Method to handle the back-end.

Initializer method

Usage:

```
metaDataPanel$new(data, id, title, description, downloadFilename)
```

Arguments:

`data` data from the `DrugExposureDiagnostics` package.

`id` the unique reference id for the module

`title` panel title

`description` description of data table

`downloadFilename` filename of the downloaded file

`input (input)`
Input from the server function.

`output (output)`
Output from the server function.

`session (session)`
Session from the server function.

Returns: (`NULL`)

Method `uiBody()`**:** Method to include a tabPanel to include the body.

Usage:

```
metaDataPanel$uiBody()
```

Returns: (`tabItem`)

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
metaDataPanel$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`mockDrugExposure` *Mock Drug exposure tables for ingredients of interest*

Description

Mock Drug exposure tables for ingredients of interest

Usage

```
mockDrugExposure(
  drug_exposure = NULL,
  concept_ancestor = NULL,
  concept_relationship = NULL,
  concept = NULL,
  drug_strength = NULL,
  ingredient_drug_records = NULL,
  drug_exposure_size = 100,
  patient_size = 50,
  person = NULL,
  observation_period = NULL,
  amount_val = c(NA, 100, 200, 300),
  den_val = c(1, 10, 100),
  amount_unit = c(8587, 8576, 9655),
  num_unit = c(8587, 8576, 9655),
  denom_unit = c(8587, 8576, 8505),
  num_val = c(1, 2, 3),
  seed = 1
)
```

Arguments

`drug_exposure`	drug exposure table
`concept_ancestor`	concept_ancestor table
`concept_relationship`	concept_relationship table
`concept`	concept table
`drug_strength`	drug strength table
`ingredient_drug_records`	modified drug exposure table having drug name
`drug_exposure_size`	the sample size of the drug exposure table
`patient_size`	the number of unique patients in the drug exposure table
`person`	person table

observation_period	observation_period table
amount_val	vector of possible numeric amount value for the drug in the drug strength table
den_val	vector of possible numeric denominator value for the drug in drug strength table
amount_unit	vector of possible amount unit type drug strength table representing milligram, milliliter and microgram
num_unit	vector of possible numerator unit type drug strength table representing milligram, milliliter and microgram
denom_unit	vector of possible numerator unit type drug strength table representing milligram, milliliter and hour
num_val	vector of possible numeric numerator denominator value drug strength table
seed	seed to make results reproducible

Value

CDMConnector CDM reference object to duckdb database with mock data include concept_ancestor, concept, drug_strength, drug_exposure tables

---

obscureCounts *Obscure the small number of counts*

Description

Obscure the small number of counts

Usage

```
obscureCounts(table, tableName, minCellCount = 5, substitute = NA)
```

Arguments

table	the table as a tibble
tableName	the table name
minCellCount	the minimum number of counts that will be displayed. If 0 all results will be reported.
substitute	the substitute value if values will be obscured

Value

the input table with results obscured if minCellCount applies

printDurationAndMessage

Print duration from start to now and print it as well as new status message

Description

Print duration from start to now and print it as well as new status message

Usage

```
printDurationAndMessage(message, start)
```

Arguments

message	the message
start	the start time

Value

the current time

runBenchmarkExecuteSingleIngredient

Run benchmark for ExecuteSingleIngredient

Description

Run benchmark for ExecuteSingleIngredient

Usage

```
runBenchmarkExecuteSingleIngredient(
  cdm,
  ingredients = c(1125315),
  subsetToConceptId = NULL,
  checks = c("missing", "exposureDuration", "quantity"),
  minCellCount = 5,
  sampleSize = 10000,
  tablePrefix = NULL,
  earliestStartDate = "2010-01-01",
  verbose = FALSE,
  byConcept = FALSE
)
```

Arguments

cdm	CDMConnector reference object
ingredients	vector of ingredients, by default: acetaminophen
subsetToConceptId	vector of concept IDs of the ingredients to filter. If a concept ID is positive it will be included, a negative one will be excluded. If NULL (default), all concept IDs for an ingredient will be considered.
checks	the checks to be executed, by default the missing values, the exposure duration and the quantity. Possible options are "missing", "exposureDuration", "type", "route", "sourceConcept", "daysSupply", "verbatimEndDate", "dose", "sig", "quantity" and "diagnosticsSummary"
minCellCount	minimum number of events to report- results lower than this will be obscured. If 0 all results will be reported.
sampleSize	the number of samples, default 10.000
tablePrefix	The stem for the permanent tables that will be created when running the diagnostics. Permanent tables will be created using this prefix, and any existing tables that start with this will be at risk of being dropped or overwritten. If NULL, temporary tables will be
earliestStartDate	the earliest date from which a record can be included
verbose	verbose, default FALSE
byConcept	boolean argument whether to return results by Concept or overall only

Value

a tibble with the time taken and memory usage for different analysis per ingredient

Examples

```
## Not run:
cdm <- mockDrugExposure()

benchmarkResults <- runBenchmarkExecuteSingleIngredient(cdm)

## End(Not run)
```

ShinyApp *DrugExposureDiagnostics ShinyApp*

Description

DrugExposureDiagnostics shiny app that shows tables and plots

Details

The module consists of the following:

"dataPlotPanel" Table and a plot (bar or box) for each check.

"metaDataPanel" Table containing the metadata.

Super class

`DrugExposureDiagnostics::ShinyModule` -> `ShinyApp`

Methods

Public methods:

- `ShinyApp$new()`
- `ShinyApp$clone()`

Method `new()`**:** Initializer method

Usage:

```
ShinyApp$new(resultList, database_id = NULL)
```

Arguments:

`resultList` (`list`) List containing the output of the checks

`database_id` (`character`) Database identifier (optional)

Returns: (`invisible(self)`)

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ShinyApp$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`ShinyModule` *Module Decorator Class*

Description

Module Decorator Class

Module Decorator Class

Active bindings

`instanceId` (`character(1)`) Random ID

`parentNamespace` (`character(1)`) Namespace parent module

`moduleName` (`character(1)`) Module name

`moduleId` (`character(1)`) Module id `moduleName-instanceId`

`namespace` (`character(1)`) Namespace, composed like: `[parentNamespace-]moduleName-instanceId` where `parentNamespace` is optional

`reactiveValues` (`reactivevalues`) Reactive values. use `shiny::isolate()` to get a non-reactive item from the reactive environment.

Methods

Public methods:

- `ShinyModule$new()`
- `ShinyModule$validate()`
- `ShinyModule$UI()`
- `ShinyModule$server()`
- `ShinyModule$clone()`

Method `new()`**:** Initializer method

Usage:

```
ShinyModule$new()
```

Returns: (`self`)

Method `validate()`**:** Validator method

Usage:

```
ShinyModule$validate()
```

Returns: (`self`)

Method `UI()`**:** Method to include a tagList to include the body.

Usage:

```
ShinyModule$UI()
```

Returns: (`tagList`)

Method `server()`**:** Method to handle the back-end.

Usage:

```
ShinyModule$server(input, output, session)
```

Arguments:

`input` (`input`) Input from the server function.

`output` (`output`) Output from the server function.

`session` (`session`) Session from the server function.

Returns: (`NULL`)

Method `clone()`**:** The objects of this class are cloneable with this method.

Usage:

```
ShinyModule$clone(deep = FALSE)
```

Arguments:

`deep` Whether to make a deep clone.

`summariseChecks` *Create a summary about the diagnostics results*

Description

Create a summary about the diagnostics results

Usage

```
summariseChecks(resultList, minCellCount = 5)
```

Arguments

`resultList`	a list with the diagnostics results
`minCellCount`	minimum number of events to report, results lower than this will be obscured. If 0 all results will be reported.

Value

a table containing the diagnostics summary

`summariseDrugExposureDuration`
Summarise drug exposure record durations

Description

Summarise drug exposure record durations

Usage

```
summariseDrugExposureDuration(
  cdm,
  drugRecordsTable = "ingredient_drug_records",
  byConcept = TRUE,
  sampleSize = 10000
)
```

Arguments

cdm	CDMConnector reference object
drugRecordsTable	a modified version of the drug exposure table, default "ingredient_drug_records"
byConcept	by individual drug Concept
sampleSize	the sample size given in execute checks

Value

a table with the drug exposure record durations

---

summariseQuantity *Summarise the quantity column of the drug_exposure table*

Description

Summarise the quantity column of the drug_exposure table

Usage

```
summariseQuantity(
  cdm,
  drugRecordsTable = "ingredient_drug_records",
  byConcept = TRUE,
  sampleSize = sampleSize
)
```

Arguments

cdm	CDMConnector reference object
drugRecordsTable	a modified version of the drug exposure table, default "ingredient_drug_records"
byConcept	whether to get result by drug concept
sampleSize	the sample size given in execute checks

Value

a table with the summarized quantity result

summariseTimeBetween *Check time in between drug records per person and report the summary*

Description

Check time in between drug records per person and report the summary

Usage

```
summariseTimeBetween(
  cdm,
  drugRecordsTable = "ingredient_drug_records",
  byConcept = TRUE,
  sampleSize = 10000
)
```

Arguments

cdm	CDMConnector reference object
drugRecordsTable	a modified version of the drug exposure table, default "ingredient_drug_records"
byConcept	whether to get result by drug concept
sampleSize	the sample size given in execute checks

Value

a table with the stats about the time between

validateChecks *Validate the "checks" parameter*

Description

Validate the "checks" parameter

Usage

```
validateChecks(checks, messageStore)
```

Arguments

checks	the checks that have been passed to executeChecks
messageStore	checkmate collection

viewResults *View the results in the Shiny app*

Description

View the results in the Shiny app

Usage

```
viewResults(
  dataFolder,
  makePublishable = FALSE,
  publishDir = file.path(getwd(), "ResultsExplorer"),
  overwritePublishDir = FALSE,
  launch.browser = FALSE
)
```

Arguments

dataFolder	A folder where the exported zip files with the results are stored. Zip files containing results from multiple databases can be placed in the same folder.
makePublishable	(Optional) copy data files to make app publishable to posit connect/shinyapp.io
publishDir	If make publishable is true - the directory that the shiny app is copied to
overwritePublishDir	(Optional) If make publishable is true - overwrite the directory for publishing
launch.browser	Should the app be launched in your default browser, or in a Shiny window. Note: copying to clipboard will not work in a Shiny window.

Details

Launches a Shiny app that allows the user to explore the diagnostics

writeFile *Write a result to a file on disk.*

Description

Write a result to a file on disk.

Usage

```
writeFile(result, resultName, databaseId, dbDir)
```

Arguments

result	check result
resultName	name of the result
databaseId	database identifier
dbDir	output directory for current db

Value

No return value, called for side effects

Examples

```
## Not run:
resultList <- list("mtcars" = mtcars)
result <- writeZipToDisk(
  metadata = metadata,
  databaseId = "mtcars",
  outputFolder = here::here()
)

## End(Not run)
```

---

writeIngredientResultToDisk

Write (ingredient) diagnostics results on disk in given output folder.

Description

Write (ingredient) diagnostics results on disk in given output folder.

Usage

```
writeIngredientResultToDisk(
  resultList,
  databaseId,
  outputFolder,
  clearDBDir = FALSE
)
```

Arguments

resultList	named list with results
databaseId	database identifier
outputFolder	folder to write to
clearDBDir	if database directory should be cleared

Value

No return value, called for side effects

Examples

```
## Not run:
resultList <- list("mtcars" = mtcars)
result <- writeIngredientResultToDisk(
  resultList = resultList,
  databaseId = "mtcars",
  outputFolder = here::here()
)

## End(Not run)
```

---

writeResultToDisk *Write diagnostics results to a zip file on disk in given output folder.*

Description

Write diagnostics results to a zip file on disk in given output folder.

Usage

```
writeResultToDisk(resultList, databaseId, outputFolder, filename = NULL)
```

Arguments

resultList	named list with results
databaseId	database identifier
outputFolder	folder to write to
filename	output filename, if NULL it will be equal to databaseId

Value

No return value, called for side effects

Examples

```
## Not run:
resultList <- list("mtcars" = mtcars)
result <- writeResultToDisk(
  resultList = resultList,
  databaseId = "mtcars",
  outputFolder = here::here()
)

## End(Not run)
```

writeZipToDisk *Write (ingredient) diagnostics results on disk in given output folder.*

Description

Write (ingredient) diagnostics results on disk in given output folder.

Usage

```
writeZipToDisk(metadata, databaseId, outputFolder, filename = NULL)
```

Arguments

metadata	metadata results
databaseId	database identifier
outputFolder	folder to write to
filename	output filename for the zip file

Value

No return value, called for side effects

Examples

```
## Not run:
resultList <- list("mtcars" = mtcars)
result <- writeZipToDisk(
  metadata = metadata,
  databaseId = "mtcars",
  outputFolder = here::here()
)

## End(Not run)
```

Index

checkDaysSupply, 3
checkDbType, 4
checkDrugDose, 4
checkDrugSig, 5
checkIngredientInTable, 5
checkIsIngredient, 6
checkLogical, 6
checkSampleMinCellCount, 7
checkTableExists, 7
checkVerbatimEndDate, 8
computeDBQuery, 8

dataPlotPanel, 9
DrugExposureDiagnostics::ShinyModule, *9*, *19*, *25*

executeChecks, 10
executeChecksSingleIngredient, 12

getAllCheckOptions, 13
getDrugMissings, 14
getDrugRecords, 14
getDrugRoutes, 15
getDrugSourceConcepts, 16
getDrugTypes, 16
getDuration, 17

ingredientDescendantsInDb, 18

mem_change, 18
mem_used, 19
metaDataPanel, 19
mockDrugExposure, 21

obscureCounts, 22

printDurationAndMessage, 23

runBenchmarkExecuteSingleIngredient, 23

ShinyApp, 24
ShinyModule, 25
summariseChecks, 27
summariseDrugExposureDuration, 27
summariseQuantity, 28
summariseTimeBetween, 29

tabPanel, *10*, *20*
tagList, *26*

validateChecks, 29
viewResults, 30

writeFile, 30
writeIngredientResultToDisk, 31
writeResultToDisk, 32
writeZipToDisk, 33