Package: DrugExposureDiagnostics (via r-universe)

September 26, 2024

Title Diagnostics for OMOP Common Data Model Drug Records

Version 1.0.9

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 7.3.2

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), shiny (>= 1.6.0)

Suggests testthat (>= 3.0.0), duckdb, odbc, DBI, knitr, rmarkdown, zip, lubridate, tibble, DT, graphics, SqlRender, magick, DiagrammeRsvg, ggplot2, cowplot, plotly, 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

NeedsCompilation no

Author Ger Inberg [aut, cre] (<https://orcid.org/0000-0001-8993-8748>), Edward Burn [aut] (<https://orcid.org/0000-0002-9286-1128>), Theresa Burkard [aut] (<https://orcid.org/0000-0003-1313-4473>), Yuchen Guo [ctb] (<https://orcid.org/0000-0002-0847-4855>), Marti Catala [ctb] (<https://orcid.org/0000-0003-3308-9905>), Mike Du [ctb]

(<https://orcid.org/0000-0002-9517-8834>), Xintong Li [ctb]
(<https://orcid.org/0000-0002-6872-5804>), Ross Williams [ctb]
(<https://orcid.org/0000-0001-7723-417X>), Erasmus MC [cph]

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

Repository CRAN

Date/Publication 2024-09-25 12:50:02 UTC

Contents

checkDaysSupply . . . 3
checkDbType . . . 3
checkDrugDose . . . 4
checkDrugSig . . . 4
checkIngredientInTable . . . 5
checkIsIngredient . . . 5
checkLogical . . . 6
checkSampleMinCellCount . . . 6
checkTableExists . . . 7
checkVerbatimEndDate . . . 7
computeDBQuery . . . 8
executeChecks . . . 8
executeChecksSingleIngredient . . . 10
getDrugMissings . . . 11
getDrugRecords . . . 11
getDrugRoutes . . . 12
getDrugSourceConcepts . . . 13
getDrugStrength . . . 13
getDrugTypes . . . 14
getDuration . . . 15
ingredientDescendantsInDb . . . 15
mockDrugExposure . . . 16
obscureCounts . . . 17
printDurationAndMessage . . . 18
summariseChecks . . . 19
summariseDrugExposureDuration . . . 19
summariseQuantity . . . 20
viewResults . . . 20
writeResultToDisk . . . 21

Index **22**

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

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
)
```

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" 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

Value

named list with results

Examples

```
## Not run:
db <- DBI::dbConnect(" Your database connection here ")
cdm <- CDMConnector::cdm_from_con(
  con = db,
  cdm_schema = "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
)
```

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 restults by Concept or overall only

Value

named list with results

---

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",
  tablePrefix = NULL,
  verbose = FALSE
)
```

Arguments

`cdm`	CDMConnector reference object
`ingredient`	Concept ID for ingredient of interest
`includedConceptsTable`	includedConceptsTable
`drugRecordsTable`	drugRecordsTable, 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`	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

getDrugStrength *Drug strength records for ingredients of interest*

Description

Drug strength records for ingredients of interest

Usage

```
getDrugStrength(
  cdm,
  ingredient,
  includedConceptsTable = "ingredient_concepts",
  drugStrengthTable = "drug_strength",
  tablePrefix = NULL,
  verbose = FALSE
)
```

Arguments

`cdm`	CDMConnector reference object
`ingredient`	ingredient concept ID for ingredient of interest
`includedConceptsTable`	table name for the concept ids, names and units
`drugStrengthTable`	table name for drug strength, default "drug_strength"
`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 strength records

`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

`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

summariseChecks *Create a summary about the diagnostics results*

Description

Create a summary about the diagnostics results

Usage

```
summariseChecks(resultList)
```

Arguments

resultList	a list with the diagnostics results

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

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

---

`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)
```

Index

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

executeChecks, 8
executeChecksSingleIngredient, 10

getDrugMissings, 11
getDrugRecords, 11
getDrugRoutes, 12
getDrugSourceConcepts, 13
getDrugStrength, 13
getDrugTypes, 14
getDuration, 15

ingredientDescendantsInDb, 15

mockDrugExposure, 16

obscureCounts, 17

printDurationAndMessage, 18

summariseChecks, 19
summariseDrugExposureDuration, 19
summariseQuantity, 20

viewResults, 20

writeResultToDisk, 21