Practical

Maarten van Kessel & Ger Inberg

2023-08-04

Prerequisits

It is assumed R, RTools, Rstudio and Java are installed accourding to the Setting up the R environment instructions.

The following packages should be installed: 1. checkmate 2. DatabaseConnector 3. remotes 4. Eunomia 5. testthat 6. knitr 7. rmarkdown 8. usethis 9. roxygen2

Assignments

1. Create an R-project

Create a new R Package project in Rstudio (File -> New Project…). Name the project Workshop. I.e. **vankesselWorkshop** or **inbergWorkshop**

2. Transform the following snippet into a function called countPersons

The function should take the following parameters: *connectionDetails* and *cdmSchema*

```r
library(DatabaseConnector)
connection <- connect(connectionDetails)
sql <- "SELECT COUNT(*) AS person_count
FROM @cdm.person;"

renderTranslateQuerySql(connection, sql, cdm = "main")
```

You can use the following function definition as a start.

```r
countPersons <- function(connectionDetails, cdmSchema) {
  # Your implementation
  result <- ...
  return(result)
}
```

3. Add function documentation to countPersons.

You can use the `roxygen2` package.

1. Add a function title.
2. Add a function description.
3. Add a parameter description for *connectionDetails* and *cdmSchema*.
4. Add the function to the exported functions.
5. Add a return description to what the function returns.
6. Add a **working** example.

4. Add parameter checking for parameters *connectionDetails* and *cdmSchema* to countPersons.

You can use the following functions: `checkmate::makeAssertCollection()`, `checkmate::assert_class()`, `checkmate::assert_character()`, `checkmate::reportAssertions()`; of the `checkmate` package.

5. Add the used dependencies to the DESCRIPTION file.

You an use the `usethis::use_package()` function, of the `usethis` package.

6. Add unit testing using.

You can setup the unit testing suite with `usethis::use_testthat()`, of the `usethis` package.

You can use functions from `testthat` to test the functionality.

7. Add package documentation vignettes.

You can use `usethis::use_vignette()` to setup the vignette.

In the vignette showcase how **countPersons** works.

Optional

C. Transform the following snippet into a function called countDrug.

Repeat steps 3-7 for this function.

```r
library(DatabaseConnector)
connection <- connect(connectionDetails)
sql <- "SELECT COUNT(DISTINCT(person_id)) AS person_count
FROM @cdm.drug_exposure
INNER JOIN @cdm.concept_ancestor
  ON drug_concept_id = descendant_concept_id
INNER JOIN @cdm.concept ingredient
  ON ancestor_concept_id = ingredient.concept_id
WHERE LOWER(ingredient.concept_name) = 'celecoxib'
  AND ingredient.concept_class_id = 'Ingredient'
  AND ingredient.standard_concept = 'S';"
```

```r
renderTranslateQuerySql(connection, sql, cdm = "main")
```

```r
countDrug <- function(connectionDetails, cdmSchema, drugName) {
  # Your implementation
  result <- ...
  return(result)
}
```

B. Transform the following snippet into a function called countDrugCondition

Repeat steps 3-7 for this function.

```r
library(DatabaseConnector)
connection <- connect(connectionDetails)

sql <- "SELECT COUNT(DISTINCT(person_id)) AS person_count
FROM @cdm.drug_era
INNER JOIN @cdm.concept ingredient
  ON drug_concept_id = ingredient.concept_id
WHERE LOWER(ingredient.concept_name) = 'celecoxib'
  AND ingredient.concept_class_id = 'Ingredient'
  AND ingredient.standard_concept = 'S';"

renderTranslateQuerySql(connection, sql, cdm = "main")
```

```r
countDrugCondition <- function(connectionDetails, cdmSchema, drugName, conditionId) {
  # Your implementation
  result <- ...
  return(result)
}
```