Practical

Maarten van Kessel & Ger Inberg

2023-08-10

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. *knitr*
3. *rmarkdown*
4. *Hades*
    a. *Eunomia*
    b. *DatabaseConnector*
5. *devtools*
    a. *testthat*
    b. *roxygen2*
    c. *remotes*
    d. *usethis*

```
# Install packages from CRAN
install.packages(c("devtools", "checkmate", "knitr", "rmarkdown", "DatabaseConnector"))

# Install packages from OHDSI
remotes::install_github("OHDSI/Eunomia")
```

Assignments

1. Create an R-project

1. Create a new R Package project in Rstudio (File -> New Project...). Name the project Workshop. I.e. **mVankesselWorkshop** or **gInbergWorkshop**.
2. Remove the **hello.R** and **man/hello.Rd** files.
3. Create a new R-file named **getCounts.R**.

2. Transform the following snippet into a function called countPersons

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

```
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.

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

Make sure you close the database connection regardless if the function throws an error or not.

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. Import `DatabaseConnector` and `checkmate` in your package.
5. Add the function to the exported functions.
6. Add a return description to what the function returns.
7. Add a **working** example.
8. Use `roxygen2::roxygenise()` to add the documentation files.

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

You can use various functions of `checkmate`.

5. The DESCRIPTION file.

1. Add the used dependencies
2. Add a title in *Title Case*.
3. Add an author.
4. Add an maintainer.
5. Add an encoding.

You can use various functions of `usethis`.

6. Add unit testing.

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, and what the results mean.

8. R-Check

To verify that the package is installable and use able use `devtools::check()`. This may take 1-5 minutes. Investigate, and resolve any *Error*, *Warning*, and *NOTE* messages that you encounter.

Optional

A. Code review

Review and compare your implementation with the *answers*.

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

Repeat steps 3-8 for this function.

Use the following snippet to create a function that counts the number of people that are taking a drug by name.

```
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';"

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

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

C. Transform the following snippet into a function called countDrugCondition

Repeat steps 3-7 for this function.

Use the following snippet to count the number of people, with a condition (by concept ID), taking a drug (by name).

```
library(DatabaseConnector)

connection <- connect(connectionDetails)

sql <- "SELECT COUNT(*) AS diagnose_count
FROM @cdm.drug_era
INNER JOIN @cdm.concept ingredient
  ON drug_concept_id = ingredient.concept_id
INNER JOIN @cdm.condition_occurrence
  ON condition_start_date >= drug_era_start_date
    AND condition_start_date <= drug_era_end_date
INNER JOIN @cdm.concept_ancestor
  ON condition_concept_id = descendant_concept_id
WHERE LOWER(ingredient.concept_name) = 'celecoxib'
  AND ingredient.concept_class_id = 'Ingredient'
  AND ingredient.standard_concept = 'S'
  AND ancestor_concept_id = 192671;"

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

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