Example package results spec

James P. Gilbert

2023-09-12

Contents

1 Introduction

This guide intends to server as an example of using RMM to build and maintain a package that produces results in an end to end manner. The aspects this package will cover are as follows:

- Creating a basic package results specification
- Using this specification to create a database schema and instantiating it in an SQLite database
- Creating a database migration for this project

2 Setup R for data export

2.1 Project setup

First we will create an R package called `SimpleFeatureExtractor` a toy example that pulls a set of aggregate features for specified cohorts from an OMOP CDM and exports the result set to a relational database for further analysis.

In this example we will export a single csv file that contains the following:

table_name	column_name	data_type	is_required	primary_key	optional	empty_is_na
covariate_definition	covariate_id	int	Yes	Yes	No	No
covariate_definition	covariate_name	varchar	Yes	No	No	No
covariate_result	cohort_definition_id	int	Yes	Yes	No	No
covariate_result	covariate_id	bigint	Yes	Yes	No	No
covariate_result	covariate_mean	numeric	Yes	No	No	No

Results exported from this package are covariate prevalances related to a cohorts and given a covariate_id, these are related to names in a second table. This table should be saved to the `inst` folder of your R pacakge. Preferably, this file should be called `resultsDataModelSpecification.csv`

The package should create results csv files that correspond to these fields in terms of type and name.

2.2 Creating a results database schema

First we should load our specification

```
specification <- ResultModelManager::loadResultsDataModelSpecifications("resultsDataModelSpecification"
```

We can then create our schema from this sql:

```
sql <- ResultModelManager::generateSqlSchema(schemaDefinition = specification)
```

Viewing the sql we can see that we should add a `database_schema` parameter when executing the sql and `table_prefix` if we need it.

```
writeLines(sql)
```

```
## {DEFAULT @table_prefix = ''}
## {DEFAULT @covariate_definition = covariate_definition}
## {DEFAULT @covariate_result = covariate_result}
##
## CREATE TABLE @database_schema.@table_prefix@covariate_definition (
##       covariate_id INT NOT NULL,
##   covariate_name VARCHAR,
##  PRIMARY KEY(covariate_id)
## );
##
## CREATE TABLE @database_schema.@table_prefix@covariate_result (
##       cohort_definition_id INT NOT NULL,
##   covariate_id BIGINT NOT NULL,
##   covariate_mean NUMERIC,
##  PRIMARY KEY(cohort_definition_id,covariate_id)
## );
```

We can then easily use this to create a schema using a `QueryNamespace`:

```
connectionDetails <- DatabaseConnector::createConnectionDetails(
  dbms = "sqlite",
  server = tempfile()
)
qns <- ResultModelManager::createQueryNamespace(
  connectionDetails = connectionDetails,
  tableSpecification = specification,
  tablePrefix = "my_study_",
  database_schema = "main"
)
```

```
## Connecting using SQLite driver
```

```
# note - the table prefix and schema parameters are not neeeded when we do this
qns$executeSql(sql)
```

```
##   |
```

```
## Executing SQL took 0.0676 secs
```

Alternatively, we can just use `DatabaseConnector` functions directly.

```
connection <- DatabaseConnector::connect(connectionDetails)
DatabaseConnector::renderTranslateExecuteSql(connection,
  sql,
  table_prefix = "my_study_",
  database_schema = "main"
)
```

2.3 Uploading results

Now we have a schema we can upload results to it using the functionality exposed in this package. Using the above example, a results folder should have the following files:

`results/:`

File Name	Description
covariate_definition.csv	Covariate Definition File
covariate_result.csv	Covariate Result File

We can now use the results spec to upload these files (and validate that they conform to the specification):

```
ResultModelManager::uploadResults(connectionDetails,
  schema = "main",
  resultsFolder = "results",
  tablePrefix = "my_study_",
  specifications = specification
)
```

With the results uploaded we can now write queries inside the namespace:

```
qns$queryDb("SELECT * FROM @database_schema.@covariate_definition")
```

```
## [1] covariateId   covariateName
## <0 rows> (or 0-length row.names)
```

```
qns$queryDb("SELECT * FROM @database_schema.@covariate_result WHERE cohort_definition_id = @cohort_id",
  cohort_id = 5
)
```

```
## [1] cohortDefinitionId covariateId        covariateMean
## <0 rows> (or 0-length row.names)
```