Vignette

Gowtham Rao

2020-06-11

Contents

```r
library(ROhdsiWebApi)
library(magrittr)

if (Sys.getenv("baseUrl") == '') {
  server <- "ohdsiBaseUrl"
} else {server <- "baseUrl"}
baseUrl <- Sys.getenv(server)
```

ROhdsiWebApi is part of HADES.

1 Introduction

From Package Readme

> ROhdsiWebApi is a R based interface to 'WebApi' (OHDSI RESTful services), and performs GET/PULL/POST/DELETE calls via the WebApi. All objects starting from R or output to R - are analysis ready R-objects like list and data.frame. The package handles the intermediary steps by converting R-objects to JSON and vice versa. To ensure r-objects are analysis ready, the objects are type converted where possible, e.g. date/date time are converted from string to POSIXct.
>
> This package makes reproducible research easier, by offering ability to retrieve detailed study specifications, transport study specifications from one instance to another, programmatically invoke the generation of a sequence of steps that are part of a study, manage running studies in batch mode.

This document will attempt to explain how ROhdsiWebApi maybe used to achieve reproducible research.

2 Knowing your WebApi configurations

To successfully use ROhdsiWebApi, it is necessary to have an active 'WebApi' endpoint with a known baseUrl such as "http://server.org:80/WebAPI". 'WebApi' has many functional categories.

2.0.1 Supported WebApi functional categories.

```
#> # A tibble: 7 x 2
#>   ` Category`      Features
#>   <chr>            <chr>
#> 1 ConceptSet       Functions for interfacing with ConceptSet in WebApi
#> 2 Cohort           Functions for interfacing with Cohort in WebApi
#> 3 IncidenceRate    Functions for interfacing with IncidenceRate in Web~
#> 4 Estimation       Functions for interfacing with Estimation in WebApi
#> 5 Prediction       Functions for interfacing with Prediction in WebApi
#> 6 Characterization Functions for interfacing with Characterization in ~
#> 7 Pathway          Functions for interfacing with Pathway in WebApi
```

3 General framework

One approach to understand the general framework for this package is to understand the CRUD framework for WebApi, i.e. the GET, PUT, DELETE, POST calls to the API. See the documentation of the WebApi. For each category supported the WebApi, ROhdsiWebAPi supports GET, PUT, DELETE, POST calls to the API, where appropriate. e.g. `getDefinitionMetaData` function is a general function that is able to get the Metadata for all specifications within a category.

```
ROhdsiWebApi::getDefinitionsMetadata(baseUrl = baseUrl, category = 'cohort') %>%
  dplyr::arrange(.data$id) %>%
  dplyr::rename_all(.funs = SqlRender::camelCaseToTitleCase) %>%
  head()
#> # A tibble: 6 x 7
#>      Id Name  `Created By` `Created Date`      `Modified By`
#>   <int> <chr> <chr>        <dttm>              <chr>
#> 1     2 Zhe'~ ""           2015-03-06 20:49:00 ""
#> 2     3 Fema~ ""           2015-03-06 20:49:00 ""
#> 3     4 Pret~ ""           2015-03-06 20:49:00 ""
#> 4     5 Post~ ""           2015-03-06 20:49:00 ""
#> 5     6 post~ ""           2015-03-06 20:49:00 ""
#> 6     7 test~ ""           2015-03-06 20:49:00 ""
#> # ... with 2 more variables: Description <chr>, `Modified Date` <dttm>
```

The same output may be achieved using

```
ROhdsiWebApi::getCohortDefinitionsMetaData(baseUrl = baseUrl) %>%
  dplyr::arrange(.data$id) %>%
  dplyr::rename_all(.funs = SqlRender::camelCaseToTitleCase) %>%
  head()
#> # A tibble: 6 x 7
#>      Id Name  `Created By` `Created Date`      `Modified By`
#>   <int> <chr> <chr>        <dttm>              <chr>
#> 1     2 Zhe'~ ""           2015-03-06 20:49:00 ""
#> 2     3 Fema~ ""           2015-03-06 20:49:00 ""
#> 3     4 Pret~ ""           2015-03-06 20:49:00 ""
#> 4     5 Post~ ""           2015-03-06 20:49:00 ""
#> 5     6 post~ ""           2015-03-06 20:49:00 ""
#> 6     7 test~ ""           2015-03-06 20:49:00 ""
#> # ... with 2 more variables: Description <chr>, `Modified Date` <dttm>
```

Similar approach may be used for all categories as follows:

```
ROhdsiWebApi::getDefinitionsMetadata(baseUrl = baseUrl, category = 'estimation') %>%
  dplyr::arrange(.data$id) %>%
  dplyr::rename_all(.funs = SqlRender::camelCaseToTitleCase) %>%
  head()
#> # A tibble: 6 x 6
#>      Id Name  `Created Date`      `Modified Date`     Type  Description
#>   <int> <chr> <dttm>              <dttm>              <chr> <chr>
#> 1     1 Grah~ 2018-10-24 16:06:58 2019-05-01 08:59:47 Comp~ <NA>
#> 2     3 SCYo~ 2018-11-01 04:35:32 2019-02-19 20:56:20 Comp~ <NA>
#> 3     4 COPY~ 2018-11-20 18:54:33 2018-11-21 03:34:41 Comp~ <NA>
#> 4     5 COX2~ 2018-11-21 06:22:30 2019-03-21 04:17:59 Comp~ <NA>
#> 5     6 Hypo~ 2018-11-21 07:08:12 2019-03-13 22:33:41 Comp~ <NA>
#> 6     7 New ~ 2018-11-22 04:48:49 NA                  Comp~ <NA>
```

```
ROhdsiWebApi::getEstimationDefinitionsMetaData(baseUrl = baseUrl) %>%
  dplyr::arrange(.data$id) %>%
  dplyr::rename_all(.funs = SqlRender::camelCaseToTitleCase) %>%
  head()
#> # A tibble: 6 x 6
#>      Id Name  `Created Date`      `Modified Date`     Type  Description
#>   <int> <chr> <dttm>              <dttm>              <chr> <chr>
#> 1     1 Grah~ 2018-10-24 16:06:58 2019-05-01 08:59:47 Comp~ <NA>
#> 2     3 SCYo~ 2018-11-01 04:35:32 2019-02-19 20:56:20 Comp~ <NA>
#> 3     4 COPY~ 2018-11-20 18:54:33 2018-11-21 03:34:41 Comp~ <NA>
#> 4     5 COX2~ 2018-11-21 06:22:30 2019-03-21 04:17:59 Comp~ <NA>
#> 5     6 Hypo~ 2018-11-21 07:08:12 2019-03-13 22:33:41 Comp~ <NA>
#> 6     7 New ~ 2018-11-22 04:48:49 NA                  Comp~ <NA>
```

This is a generic framework that applies to most WebApi categories, and supports different types of CRUD functionalities like deleteConceptSetDefinition() vs deleteDefinition(category = 'conceptSet').

4 Concept Set

Please review 'Concept sets - The Book of OHDSI'

We commonly post concept set expression into WebApi/Atlas, or try get an expression from Atlas/WebApi based on a conceptSetDefinitionId.

Example: lets say we have concept set expression as follows, that is being used for a Rheumatoid Arthritis study.

```
jsonExpression <- '{
  "items": [
    {
      "concept": {
        "CONCEPT_ID": 81097,
        "CONCEPT_NAME": "Feltys syndrome",
        "STANDARD_CONCEPT": "S",
        "STANDARD_CONCEPT_CAPTION": "Standard",
        "INVALID_REASON": "V",
        "INVALID_REASON_CAPTION": "Valid",
        "CONCEPT_CODE": "57160007",
        "DOMAIN_ID": "Condition",
        "VOCABULARY_ID": "SNOMED",
        "CONCEPT_CLASS_ID": "Clinical Finding"
      },
      "isExcluded": true,
      "includeDescendants": false,
      "includeMapped": false
    },
    {
      "concept": {
        "CONCEPT_ID": 80809,
        "CONCEPT_NAME": "Rheumatoid arthritis",
        "STANDARD_CONCEPT": "S",
        "STANDARD_CONCEPT_CAPTION": "Standard",
        "INVALID_REASON": "V",
        "INVALID_REASON_CAPTION": "Valid",
        "CONCEPT_CODE": "69896004",
        "DOMAIN_ID": "Condition",
        "VOCABULARY_ID": "SNOMED",
        "CONCEPT_CLASS_ID": "Clinical Finding"
      },
      "isExcluded": false,
      "includeDescendants": true,
      "includeMapped": false
    },
    {
      "concept": {
        "CONCEPT_ID": 4035611,
        "CONCEPT_NAME": "Seropositive rheumatoid arthritis",
        "STANDARD_CONCEPT": "S",
        "STANDARD_CONCEPT_CAPTION": "Standard",
        "INVALID_REASON": "V",
        "INVALID_REASON_CAPTION": "Valid",
        "CONCEPT_CODE": "239791005",
        "DOMAIN_ID": "Condition",
        "VOCABULARY_ID": "SNOMED",
        "CONCEPT_CLASS_ID": "Clinical Finding"
      },
      "isExcluded": false,
      "includeDescendants": true,
      "includeMapped": false
    }
  ]
}'
```

Lets call this concept set expression - '[ROhdsiWebApi Vignette] Rheumatoid Arthritis concept set'.

Note: function does not accept JSON. It needs to be converted to R (list) expression

We can post the concept set expression into WebApi as follows:

```
#> Post ConceptSet definition was successful
```

If successful, we will get a return object as follows into R.

```
#> # A tibble: 1 x 6
#>   createdBy modifiedBy createdDate         modifiedDate            id
#>   <lgl>     <lgl>      <dttm>              <dttm>               <int>
#> 1 NA        NA         2020-06-11 06:28:58 2020-06-11 06:28:58 1.86e6
#> # ... with 1 more variable: name <chr>
```

The id of the newly posted concept-set definition is 1864152. We can now use this concept-set for many concept set queries eg.,

4.1 if want to print ready expression of the concept set definition

```
conceptSetDefinition = getConceptSetDefinition(conceptSetId = returnFromPostRequest$id, baseUrl = baseU
conceptTbl <- convertConceptSetDefinitionToTable(conceptSetDefinition)
names(conceptTbl) <- SqlRender::camelCaseToTitleCase(names(conceptTbl))
conceptTbl
#> # A tibble: 3 x 13
#>   `Is Excluded` `Include Descen~ `Include Mapped` `Concept Id`
#>   <lgl>         <lgl>            <lgl>                   <int>
#> 1 TRUE          FALSE            FALSE                   81097
#> 2 FALSE         TRUE             FALSE                   80809
#> 3 FALSE         TRUE             FALSE                 4035611
#> # ... with 9 more variables: `Concept Name` <chr>, `Standard
#> #   Concept` <chr>, `Standard Concept Caption` <chr>, `Invalid
#> #   Reason` <chr>, `Invalid Reason Caption` <chr>, `Concept
#> #   Code` <chr>, `Domain Id` <chr>, `Vocabulary Id` <chr>, `Concept
#> #   Class Id` <chr>
```

`createConceptSetWorkbook` maybe used to create an Excel workbook of the concept set.

If we want a list of all conceptId's (including descendants) from the concept set expression

```
resolvedConcepts = resolveConceptSet(conceptSetDefinition = conceptSetDefinition, baseUrl = baseUrl)
resolvedConcepts
#>  [1]    80809  4035427  4035611  4103516  4114439  4114440  4114441
#>  [8]  4114442  4114444  4115050  4115051  4115161  4116148  4116149
#> [15]  4116150  4116151  4116152  4116153  4116440  4116441  4116442
#> [22]  4116443  4116444  4116445  4116446  4117687  4147418  4162539
#> [29]  4179378  4179536  4200987  4269880  4296152  4311391 35609009
#> [36] 35609010 36684997 36684998 36685017 36685018 36685019 36685021
#> [43] 36685022 36685023 36686999 36687000 36687001 36687002 36687003
#> [50] 36687005 36687006 37108590 37108591 37108714 37117421 37207804
#> [57] 37207805 37207806 37207807 37209321 37209322 37209323 37209329
#> [64] 37395590 42534834 42534835 42534836 42534837 42536657 42539550
```

The concept set expression json expression can be recaptured from WebApi as follows

```
json <- getConceptSetDefinition(baseUrl = baseUrl, conceptSetId = returnFromPostRequest$id)$expression
```

5 Cohorts/Characterization/Incidence rate

Please review ‘What is a cohort - The Book of OHDSI’.

> We define a cohort as a set of persons who satisfy one or more inclusion criteria for a duration of time. The term cohort is often interchanged with the term phenotype. Cohorts are used throughout OHDSI analytical tools and network studies as the primary building blocks for executing a research question.
>
> A cohort is defined as the set of persons satisfying one or more inclusion criteria for a duration of time. One person may qualify for one cohort multiple times during non-overlapping time intervals. Cohorts are constructed in ATLAS by specifying cohort entry criteria and cohort exit criteria. Cohort entry criteria involve selecting one or more initial events, which determine the start date for cohort entry, and optionally specifying additional inclusion criteria which filter to the qualifying events. Cohort exit criteria are applied to each cohort entry record to determine the end date when the person’s episode no longer qualifies for the cohort.

Cohorts/Characterization/Incidence Rate are WebApi categories, where WebApi manages the execution of generations.

Example: We may want to know if a certain cohort specification has been generated by checking cohort generation status `getCohortGenerationInformation(baseUrl = baseUrl, cohortId= 4234)`. If a cohort is not previously generated, it may be generated using `invokeCohortSetGeneration(baseUrl = baseUrl, cohortId = 4234, sourceKey = 'HCUP')` . If it is already generated, we can extract its output of cohort generation using `getCohortResults(baseUrl, cohortId = 4234)`.

6 Characterization (TO Do)

Please review ‘Characterization - The Book of OHDSI’.

7 Population Level Effect Estimation (TO Do)

Please review ‘Population Level Effect Estimation - The Book of OHDSI’.

8 Patient Level Prediction (TO Do)

Please review ‘Patient Level Prediction - The Book of OHDSI’.