Using ROhdsiWebApi

Gowtham Rao

2020-07-06

Contents

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 WebApi configurations and ROhdsiWebApi

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.

To ensure reproducibility of work it is best to know the version of the WebApi (i.e. Atlas backend) being used. An easy way to do that is (and output maybe included in your study results)

```
version <- ROhdsiWebApi:::getWebApiVersion(baseUrl = baseUrl)
message1 <- paste0('This Vignette was created using WebApi version: ',
                   version,
                   ' on baseUrl: ',
                   baseUrl,
                   ". The CDM had the following source data configured: ")
cdmSources <- ROhdsiWebApi::getCdmSources(baseUrl = baseUrl)
priorityVocabulary <- ROhdsiWebApi::getPriorityVocabularyKey(baseUrl = baseUrl)
```

The object `version` will show your webApi version. Example: This Vignette was created using WebApi version: 2.7.4 on baseUrl: http://api.ohdsi.org:80/WebAPI. The CDM had the following source data configured: .

```
cdmSources
#> # A tibble: 3 x 7
#>   sourceId sourceName            sourceKey  sourceDialect cdmDatabaseSchema vocabDatabaseSchema resu
#>      <int> <chr>                 <chr>      <chr>         <chr>              <chr>                <chr
#> 1        4 Common Evidence Model CEM        postgresql    <NA>               unrestricted         <NA>
#> 2        6 SYNPUF 1K             SYNPUF1K   postgresql    synpuf1k           unrestricted         synp
#> 3        5 SYNPUF 5%             SYNPUF5PCT postgresql    synpuf5pct         unrestricted         synp
```

The priority vocabulary for the WebApi is SYNPUF5PCT.

We can also perform checks on the WebApi, example - we may want to see if the 'HCUP' & 'SYNPUF1K' is a valid SourceKey in the current webApi.

```
ROhdsiWebApi::isValidSourceKey(sourceKeys = c('HCUP', 'SYNPUF1K'), baseUrl = baseUrl)
#> [1] FALSE  TRUE
```

2.1 WebApi Analytical categories.

WebApi maybe considered to have certain modular analytic categories. ROhdsiWebApi supports the following categories:

Category	Features
ConceptSet	Functions for interfacing with ConceptSet in WebApi
Cohort	Functions for interfacing with Cohort in WebApi
IncidenceRate	Functions for interfacing with IncidenceRate in WebApi
Estimation	Functions for interfacing with Estimation in WebApi
Prediction	Functions for interfacing with Prediction in WebApi
Characterization	Functions for interfacing with Characterization in WebApi
Pathway	Functions for interfacing with Pathway in WebApi

3 Framework of ROhdsiWebApi

ROhdsiWebApi maybe better understood by having atleast a high level understanding of CRUD framework for WebApi, i.e. the GET, PUT, DELETE, POST calls to the API. See the documentation of the WebApi.

For each supported category, ROhdsiWebAPi performs GET, PUT, DELETE, POST calls to WebApi in background. The details of what calls are actually performed is less important to an analyst, but it is useful to understand the naming conventions of ROhdsiWebApi.

3.1 Naming conventions of ROhdsiWebApi

Most functions in ROhdsiWebApi start with an action oriented 'verb' - such as

Function Name	Description
cancelCharacterizationGeneration	Cancel Characterization Generation
cancelCohortGeneration	Cancel Cohort Generation
cancelGeneration	Cancel Generation
cancelIncidenceRateGeneration	Cancel Incidence Rate Generation
cancelPathwayGeneration	Cancel Pathway Generation
convertConceptSetDefinitionToTable	Convert Concept Set Definition To Table
createConceptSetWorkbook	Create Concept Set Workbook
deleteCharacterizationDefinition	Delete Characterization Definition
deleteCohortDefinition	Delete Cohort Definition
deleteConceptSetDefinition	Delete Concept Set Definition
deleteDefinition	Delete Definition
deleteEstimationDefinition	Delete Estimation Definition
deleteIncidenceRateDefinition	Delete Incidence Rate Definition
deletePathwayDefinition	Delete Pathway Definition
deletePredictionDefinition	Delete Prediction Definition
detectCharacterizationsByName	Detect Characterizations By Name
detectCohortsByName	Detect Cohorts By Name
detectConceptSetsByName	Detect Concept Sets By Name
detectEstimationsByName	Detect Estimations By Name
detectIncidenceRatesByName	Detect Incidence Rates By Name
detectPathwaysByName	Detect Pathways By Name
detectPredictionsByName	Detect Predictions By Name
existsCharacterizationName	Exists Characterization Name
existsCohortName	Exists Cohort Name
existsConceptSetName	Exists Concept Set Name
existsEstimationName	Exists Estimation Name
existsIncidenceRateName	Exists Incidence Rate Name
existsPathwayName	Exists Pathway Name

Function Name	Description
existsPredictionName	Exists Prediction Name
getBearerDbLogin	Get Bearer Db Login
getCdmsources	Get Cdmsources
getCharacterizationDefinition	Get Characterization Definition
getCharacterizationDefinitionsMetadata	Get Characterization Definitions Metadata
getCharacterizationGenerationinformation	Get Characterization Generationinformation
getCharacterizationResults	Get Characterization Results
getCohortDefinition	Get Cohort Definition
getCohortDefinitionExpression	Get Cohort Definition Expression
getCohortDefinitionName	Get Cohort Definition Name
getCohortDefinitionSql	Get Cohort Definition Sql
getCohortDefinitionsMetadata	Get Cohort Definitions Metadata
getCohortGenerationinformation	Get Cohort Generationinformation
getCohortInclusionrulesandcounts	Get Cohort Inclusionrulesandcounts
getCohortResults	Get Cohort Results
getCohortSql	Get Cohort Sql
getConceptSetDefinition	Get Concept Set Definition
getConceptSetDefinitionsMetadata	Get Concept Set Definitions Metadata
getConcepts	Get Concepts
getDefinition	Get Definition
getDefinitionsMetadata	Get Definitions Metadata
getEstimationDefinition	Get Estimation Definition
getEstimationDefinitionsMetadata	Get Estimation Definitions Metadata
getGenerationinformation	Get Generationinformation
getIncidenceRateDefinition	Get Incidence Rate Definition
getIncidenceRateDefinitionsMetadata	Get Incidence Rate Definitions Metadata
getIncidenceRateGenerationinformation	Get Incidence Rate Generationinformation
getIncidenceRateResults	Get Incidence Rate Results
getPathwayDefinition	Get Pathway Definition
getPathwayDefinitionsMetadata	Get Pathway Definitions Metadata
getPathwayGenerationinformation	Get Pathway Generationinformation
getPathwayResults	Get Pathway Results
getPersonProfile	Get Person Profile
getPredictionDefinition	Get Prediction Definition
getPredictionDefinitionsMetadata	Get Prediction Definitions Metadata
getPriorityvocabularykey	Get Priorityvocabularykey
getResults	Get Results
getSourceconcepts	Get Sourceconcepts
getWebApiVersion	Get Web Api Version
insertCohortDefinitionInPackage	Insert Cohort Definition In Package
insertCohortDefinitionSetInPackage	Insert Cohort Definition Set In Package
invokeCharacterizationGeneration	Invoke Characterization Generation
invokeCohortGeneration	Invoke Cohort Generation
invokeGeneration	Invoke Generation
invokeIncidenceRateGeneration	Invoke Incidence Rate Generation
invokePathwayGeneration	Invoke Pathway Generation
isvalidCharacterizationId	Isvalid Characterization Id
isvalidCohortId	Isvalid Cohort Id
isvalidConceptSetId	Isvalid Concept Set Id
isvalidEstimationId	Isvalid Estimation Id
isvalidId	Isvalid Id
isvalidIncidenceRateId	Isvalid Incidence Rate Id

Function Name	Description
isvalidPathwayId	Isvalid Pathway Id
isvalidPredictionId	Isvalid Prediction Id
isvalidSourceKey	Isvalid Source Key
postCharacterizationDefinition	Post Characterization Definition
postCohortDefinition	Post Cohort Definition
postConceptSetDefinition	Post Concept Set Definition
postDefinition	Post Definition
postEstimationDefinition	Post Estimation Definition
postIncidenceRateDefinition	Post Incidence Rate Definition
postPathwayDefinition	Post Pathway Definition
postPredictionDefinition	Post Prediction Definition
resolveConceptSet	Resolve Concept Set

Most of the functions start with the following verbs:

Function Verb	Number Of Functions
Get	38
Isvalid	9
Delete	8
Post	8
Detect	7
Exists	7
Cancel	5
Invoke	5
Insert	2
Convert	1
Create	1
Resolve	1

A function to get `Definition` is `getDefinitionMetaData` function. This is a general function that is able to get the Metadata for all specifications within a category.

```
ROhdsiWebApi::getDefinitionsMetadata(baseUrl = baseUrl,
                                     category = 'cohort') %>%
  arrange(.data$id) %>%
  rename_all(.funs = SqlRender::camelCaseToTitleCase) %>%
  tail()
#> # A tibble: 6 x 7
#>        Id Name                                                `Created By` `Created Date`
#>     <int> <chr>                                               <chr>        <dttm>
#> 1 1774574 " SH_target_hypoparathyroidism_exclude_thyroidectomy (1)" ""           2015-03-06 20:49:00
#> 2 1774575 "New Cohort / Test 2"                               ""           2015-03-06 20:49:00
#> 3 1774576 "AML patients from IPOP"                            ""           2015-03-06 20:49:00
#> 4 1774577 "[TS] LEGEND Depression IPCI"                       ""           2015-03-06 20:49:00
#> 5 1774578 "diabetes (5)"                                      ""           2015-03-06 20:49:00
#> 6 1774579 "mineral_metabolism_target"                         ""           2015-03-06 20:49:00
```

The same output may be achieved using

```
ROhdsiWebApi::getCohortDefinitionsMetaData(baseUrl = baseUrl) %>%
  arrange(.data$id) %>%
  rename_all(.funs = SqlRender::camelCaseToTitleCase) %>%
  tail()
#> # A tibble: 6 x 7
#>        Id Name                                                `Created By` `Created Date`
#>     <int> <chr>                                               <chr>        <dttm>
#> 1 1774574 " SH_target_hypoparathyroidism_exclude_thyroidectomy (1)" ""           2015-03-06 20:49:00
#> 2 1774575 "New Cohort / Test 2"                               ""           2015-03-06 20:49:00
#> 3 1774576 "AML patients from IPOP"                            ""           2015-03-06 20:49:00
#> 4 1774577 "[TS] LEGEND Depression IPCI"                       ""           2015-03-06 20:49:00
#> 5 1774578 "diabetes (5)"                                      ""           2015-03-06 20:49:00
#> 6 1774579 "mineral_metabolism_target"                         ""           2015-03-06 20:49:00
```

Similar approach may be used for all categories as follows:

```
ROhdsiWebApi::getDefinitionsMetadata(baseUrl = baseUrl,
                                     category = 'estimation') %>%
  arrange(.data$id) %>%
  rename_all(.funs = SqlRender::camelCaseToTitleCase) %>%
  tail()
#> # A tibble: 6 x 6
#>      Id Name                          `Created Date`      `Modified Date`     Type          Descript
#>   <int> <chr>                         <dttm>              <dttm>              <chr>         <chr>
#> 1   361 "COPY OF: YBY_ACEi versus TH~ 2020-06-08 02:17:38 2020-06-08 06:19:50 ComparativeC~ YBY_Comp
#> 2   362 "PLE - DOAC"                  2020-06-09 14:57:28 2020-06-09 15:08:51 ComparativeC~ <NA>
#> 3   363 "New Population Level-test"   2020-06-16 03:15:00 NA                  ComparativeC~ <NA>
#> 4   364 "[ymp] Metformin vs Sulfonyl~ 2020-06-18 01:37:44 NA                  ComparativeC~ <NA>
#> 5   365 "[YM] PLE Hypertension"       2020-06-29 11:01:10 2020-06-29 15:58:25 ComparativeC~ <NA>
#> 6   366 "SH_mineral_dementia"         2020-06-30 03:18:33 2020-06-30 03:25:39 ComparativeC~ <NA>
```

```
ROhdsiWebApi::getEstimationDefinitionsMetaData(baseUrl = baseUrl) %>%
  arrange(.data$id) %>%
  rename_all(.funs = SqlRender::camelCaseToTitleCase) %>%
  tail()
#> # A tibble: 6 x 6
#>      Id Name                          `Created Date`      `Modified Date`     Type          Descript
#>   <int> <chr>                         <dttm>              <dttm>              <chr>         <chr>
#> 1   361 "COPY OF: YBY_ACEi versus TH~ 2020-06-08 02:17:38 2020-06-08 06:19:50 ComparativeC~ YBY_Comp
#> 2   362 "PLE - DOAC"                  2020-06-09 14:57:28 2020-06-09 15:08:51 ComparativeC~ <NA>
#> 3   363 "New Population Level-test"   2020-06-16 03:15:00 NA                  ComparativeC~ <NA>
#> 4   364 "[ymp] Metformin vs Sulfonyl~ 2020-06-18 01:37:44 NA                  ComparativeC~ <NA>
#> 5   365 "[YM] PLE Hypertension"       2020-06-29 11:01:10 2020-06-29 15:58:25 ComparativeC~ <NA>
#> 6   366 "SH_mineral_dementia"         2020-06-30 03:18:33 2020-06-30 03:25:39 ComparativeC~ <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'.

We will need to check if there is a concept set by this name.

```
# check if there is a concept set by this name, if yes, delete it
exists <- ROhdsiWebApi::existsConceptSetName(conceptSetName = conceptSetName, baseUrl = baseUrl)
exists
#> # A tibble: 1 x 6
#>   createdBy modifiedBy createdDate         modifiedDate             id name
#>   <chr>     <chr>      <dttm>              <dttm>                <int> <chr>
#> 1 ""        ""         2020-07-06 22:15:08 2020-07-06 22:15:08 1864626 [ROhdsiWebApi Vignette] Rheum
```

If there is a concept set with this name, then we could either choose another name - or delete the old concept set. For this vignette we have chosen to delete any matching concept set as follows:

```
if (!isFALSE(exists)) {
  ROhdsiWebApi::deleteConceptSetDefinition(conceptSetId = exists$id, baseUrl = baseUrl)
}
#> Successfully deleted conceptSet definition id 1864626. Request status code: Success: (204) No Conten
#> NULL
```

Now we need to ensure the JSON expression above is converted to R-data object. Note: By design, ROhdsiWebAPi does not accept JSON. It needs to be converted to R (list) expression

We can now post this R-object into WebApi as follows:

```
returnFromPostRequest <- ROhdsiWebApi::postConceptSetDefinition(baseUrl = baseUrl,
                                                                conceptSetDefinition = rExpression,
                                                                name = conceptSetName)
#> 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 name
#>   <lgl>     <lgl>      <dttm>              <dttm>                <int> <chr>
#> 1 NA        NA         2020-07-06 22:54:59 2020-07-06 22:54:59 1864627 [ROhdsiWebApi Vignette] Rheuma
```

The id of the newly posted concept-set definition is 1864627. 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 = baseUrl)
conceptTbl <-
  convertConceptSetDefinitionToTable(conceptSetDefinition)
names(conceptTbl) <-
  SqlRender::camelCaseToTitleCase(names(conceptTbl))
conceptTbl
#> # A tibble: 3 x 13
#>   `Is Excluded` `Include Descen~ `Include Mapped` `Concept Id` `Concept Name` `Standard Conce~ `Stan
#>   <lgl>         <lgl>            <lgl>                  <int> <chr>          <chr>            <chr>
#> 1 TRUE          FALSE            FALSE                  81097 Felty's syndr~ S                Stand
#> 2 FALSE         TRUE             FALSE                  80809 Rheumatoid ar~ S                Stand
#> 3 FALSE         TRUE             FALSE                4035611 Seropositive ~ S                Stand
#> # ... with 4 more variables: `Concept Code` <chr>, `Domain Id` <chr>, `Vocabulary Id` <chr>, `Concep
```

`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)
print("Note: Showing only the first 10 concept id's")
#> [1] "Note: Showing only the first 10 concept id's"
resolvedConcepts[1:10]
#>  [1]   80809 4035427 4035611 4103516 4114439 4114440 4114441 4114442 4114444 4115050
```

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

```
json <-
  getConceptSetDefinition(baseUrl = baseUrl,
                          conceptSetId = returnFromPostRequest$id
                          )$expression %>%
  RJSONIO::toJSON(pretty = TRUE)
```

Similar framework maybe used with other WebApi categories such as Cohorts/Characterization/Incidence Rate.

5 Applications of ROhdsiWebAPi

A valuable feature of ROhdsiWebApi is that it is able to get full result set into R, as a data frame object. Results of Cohort, Characterization, Incidence Rate, Pathway maybe obtained. This data frame may then be converted to publication ready material by using packages like Officer, flextable. The functions in ROhdsiWebAPi maybe used to create dynamic R-shiny apps that allow user to interact with WebApi and select cohort definitions, concept sets for review or modifications. ROhdsiWebApi may be used to build 'mini versions' of Atlas that is project specific - by directly interacting with WebApi using R.

5.1 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)`.

5.2 Characterization

Please review 'Characterization - The Book of OHDSI'.

5.3 Population Level Effect Estimation

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

6 Patient Level Prediction

Please review 'Patient Level Prediction - The Book of OHDSI'.