Using CohortIncidence

Christopher Knoll

2022-04-11

Contents

1 Introduction **1**

2 Installation instructions **1**

3 Database Preparation **2**

4 A simple example **3**

4.1 Build the design 3

4.2 Using age, gender and start year strata 5

4.3 Using executeAnalysis() 5

5 Advanced Usage: Budling and Executing SQL manually **5**

5.1 Build analysis SQL from design 5

5.2 Render SQL with paramaters and execute 17

5.3 Using Temp Tables 18

```
#> Loading required package: DatabaseConnector
```

1 Introduction

This vignette describes how to use the `CohortIncidence` package to perform a single incidence rate analysis for a given target and outcome cohort, with a few settings for Time At Risk and Clean Window.

2 Installation instructions

Before installing the `CohortIncidence` package make sure you have Java available. Java can be downloaded from www.java.com. For Windows users, RTools is also necessary. RTools can be downloaded from CRAN.

The `CohortIncidence` package is currently maintained in a Github repository.

```
install.packages("remotes")
remotes::install_github("ohdsi/CohortIncidence")
```

Once installed, you can type `library(CohortIncidence)` to load the package.

3 Database Preparation

The results of the anlaysis SQL will assume a final table: `@results_database_schema.incidence_summary`. The DDL for this table can be fetched from the package via the following:

```
# Fetch DDL from package
ddl <- CohortIncidence::getResultsDdl()
cat(ddl)
CREATE TABLE @schemaName.incidence_summary
(
    ref_id int,
    database_name varchar(255),
    target_cohort_definition_id bigint,
    target_name varchar(255),
    tar_id bigint,
    tar_start_offset bigint,
    tar_start_index bigint,
    tar_end_offset bigint,
    tar_end_index bigint,
    subgroup_id bigint,
    subgroup_name varchar(255),
    outcome_id bigint,
    outcome_cohort_definition_id bigint,
    outcome_name varchar(255),
    clean_window bigint,
    age_id int,
    age_group_name varchar(255),
    gender_id int,
    gender_name varchar(255),
    start_year int,
    persons_at_risk_pe bigint,
    persons_at_risk bigint,
    person_days_pe bigint,
    person_days bigint,
    person_outcomes_pe bigint,
    person_outcomes bigint,
    outcomes_pe bigint,
    outcomes bigint,
    incidence_proportion_p100p float,
    incidence_rate_p100py float
);
```

Using `SqlRender` and `DatabaseConnector`, you can execute the above on your target database platform in order to deploy the table. Remember to replace `@schemaName` with the appropriate schema. You can also 'hack' the `@schemaName` paramater to apply a prefix by specifying the SqlRender paramater of `schemaName.incidence_summary` to a target table ie: `mySchema.prefix_incidence_summary`. Using the

same paramater name/value in `buildQuery()` will allow you to provide a prefix to the result table name instead of having to declare a separate schema.

```
connectionDetails <- DatabaseConnector::createConnectionDetails(dbms = "postgresql",server={Sys.getenv(

# to specify the target schema (the typical use case):
ddl <- SqlRender::render(CohortIncidence::getResultsDdl(), schemaName = "mySchema")

# a work-around to provide a prefix to the result table, in case creating new schema is restricted
ddlPrefix <- SqlRender::render(CohortIncidence::getResultsDdl(), "schemaName.incidence_summary" = "mySc

con <- DatabaseConnector::connect(connectionDetails)
DatabaseConnector::executeSql(ddl)
DatabaseConnector::disconnect(con)
```

4 A simple example

This example will create a CohortIncidence design containing a single target, outcome, and time at risk.

4.1 Build the design

The following script builds a single T, O and Time at Risk, and assembles those element into a design. Finally, the resulting JSON is printed.

```
t1 <- CohortIncidence::createCohortRef(id=1, name="Target cohort 1")

o1 <- CohortIncidence::createOutcomeDef(id=1,name="Outcome 1, 30d Clean",
                                        cohortId =2,
                                        cleanWindow =30)

tar1 <- CohortIncidence::createTimeAtRiskDef(id=1,
                                      startDateField="StartDate",
                                      endDateField="StartDate",
                                      endOffset=30)

# Note: c() is used when dealing with an array of numbers,
# later we use list() when dealing with an array of objects
analysis1 <- CohortIncidence::createIncidenceAnalysis(targets = c(t1$id),
                                                      outcomes = c(o1$id),
                                                      tars = c(tar1$id))

subgroup1 <- CohortIncidence::createCohortSubgroup(id=1, name="Subgroup 1", cohortRef = createCohortRef


# Create Design (note use of list() here):
irDesign <- CohortIncidence::createIncidenceDesign(targetDefs = list(t1),
                                                   outcomeDefs = list(o1),
                                                   tars=list(tar1),
                                                   analysisList = list(analysis1),
                                                   subgroups = list(subgroup1))
# Render the design as JSON
```

```
jsonlite::toJSON(irDesign,pretty = T)
#> {
#>   "cohortDefs": [],
#>   "targetDefs": [
#>     {
#>       "id": 1,
#>       "name": "Target cohort 1"
#>     }
#>   ],
#>   "outcomeDefs": [
#>     {
#>       "id": 1,
#>       "name": "Outcome 1, 30d Clean",
#>       "cohortId": 2,
#>       "cleanWindow": 30
#>     }
#>   ],
#>   "timeAtRiskDefs": [
#>     {
#>       "id": 1,
#>       "start": {
#>         "dateField": "StartDate",
#>         "offset": 0
#>       },
#>       "end": {
#>         "dateField": "StartDate",
#>         "offset": 30
#>       }
#>     }
#>   ],
#>   "analysisList": [
#>     {
#>       "targets": [1],
#>       "outcomes": [1],
#>       "tars": [1]
#>     }
#>   ],
#>   "conceptSets": [],
#>   "subgroups": [
#>     {
#>       "CohortSubgroup": {
#>         "id": 1,
#>         "name": "Subgroup 1",
#>         "cohort": {
#>           "id": 300
#>         }
#>       }
#>     }
#>   ]
#> }
```

4.2 Using age, gender and start year strata

The IR design can also include settings to specify if an analysis should be done at the age, gender or start year levels (or any combination of those choices). To use this function, you create the strata settings with the `CohortIncidence::createStrataSettings)` function:

```
irDesignWithStrata <- CohortIncidence::createIncidenceDesign(targetDefs = list(t1),
                                                              outcomeDefs = list(o1),
                                                              tars=list(tar1),
                                                              analysisList = list(analysis1),
                                                              subgroups = list(subgroup1),
                                                              #add by age and by gender strata, but don't do by st
                                                              strataSettings = CohortIncidence::createStrataSettin
```

4.3 Using executeAnalysis()

If there is no need to see the analysis sql or control the output of the analysis to a permenant table, the `executeAnalysis()` function can be used to perform the cohort incidence using a simple API:

```
buildOptions <- CohortIncidence::buildOptions(cohortTable = "demoCohortSchema.cohort",
                                              cdmDatabaseSchema = "mycdm",
                                              refId = 1)


executeResults <- CohortIncidence::executeAnalysis(connectionDetails = connectionDetails,
                                                    incidenceDesign = irDesign,
                                                    buildOptions = buildOptions)
```

The results will contain the same table structure as the results schema `incidence_summary`.

5 Advanced Usage: Budling and Executing SQL manually

There may be a reason (debugging or special processing steps) Where you would want to access the analysis SQL before execution.
The following sections describe how to fetch the SQL, translate and execute the statements.

5.1 Build analysis SQL from design

From the previous design, the `CohortIncidence::buildQuery()` method is used to generate the analysis SQL:

```
buildOptions <- CohortIncidence::buildOptions(cohortTable = "demoCohortSchema.cohort",
                                              cdmDatabaseSchema = "mycdm",
                                              resultsDatabaseSchema = "myresults",
                                              refId = 1)

analysisSql <- CohortIncidence::buildQuery(incidenceDesign =  as.character(jsonlite::toJSON(irDesign)),
                                           buildOptions = buildOptions)
cat(analysisSql)
```

```
#> select target_cohort_definition_id, target_name
#> into #target_ref
#> from (
#> select cast(1 as int) as target_cohort_definition_id,
#>  cast ('Target cohort 1' as varchar(250)) as target_name
#> ) O
#> ;
#>
#> select tar_id, tar_start_index, tar_start_offset, tar_end_index, tar_end_offset
#> into #tar_ref
#> FROM (
#> select cast(1 as int) as tar_id,
#>  cast (1 as int) as tar_start_index, cast (0 as int) as tar_start_offset,
#>  cast (1 as int) as tar_end_index, cast (30 as int) as tar_end_offset
#> ) T
#> ;
#>
#> select outcome_id, outcome_cohort_definition_id, outcome_name, clean_window, excluded_cohort_definit
#> into #outcome_ref
#> from (
#> select cast(1 as int) as outcome_id, cast(2 as int) as outcome_cohort_definition_id,
#>  cast ('Outcome 1, 30d Clean' as varchar(255)) as outcome_name,
#>  cast (30 as int) as clean_window,
#>  cast (0 as int) as excluded_cohort_definition_id
#> ) O
#> ;
#>
#> select subgroup_id, subgroup_name
#> INTO #subgroup_ref
#> FROM (
#> select cast(0 as int) as subgroup_id,
#>  cast ('All' as varchar(250)) as subgroup_name
#> UNION ALL
#> select cast(1 as int) as subgroup_id,
#>  cast ('Subgroup 1' as varchar(250)) as subgroup_name
#> ) S
#> ;
#>
#> create table #age_group
#> (
#>  age_id int NOT NULL,
#>  group_name varchar(50) NOT NULL,
#>  min_age int NULL,
#>  max_age int NULL
#>
#> );
#>
#>
#>
#> --
#> -- Begin analysis 0
#> --
#>
```

```
#> /************************************
#> code to implement calculation using the inputs above, no need to modify beyond this point
#>
#> 1) create T + TAR periods
#> 2) create table to store era-fied at-risk periods
#>   UNION all periods that don't require erafying
#>   with era-fy records that require it
#> 3) create table to store era-fied excluded at-risk periods
#>   UNION all periods that don't require erafying
#>   with era-fy records that require it
#> 4) calculate pre-exclude outcomes and outcomes
#> 5) calculate exclsion time per T/O/TAR/Subject/start_date
#> 6) generate raw result table with T/O/TAR/subject_id, start_date, pe_at_risk (datediff(d,start,end),
#>    attach age/gender/year columns
#> 7) Create analysis_ref to produce each T/O/TAR combo
#> 8) perform rollup to calculate IR at the T/O/TAR/S/[age|gender|year] inclusing distinct people and d
#>
#> ************************************/
#>
#> --three ways for entry into excluded
#> --1:  duration of outcome periods  (ex:  immortal time due to clean period)
#> --2:  other periods excluded  (ex: persons post-appendectomy for appendicitis)
#>
#>
#> -- 1) create T + TAR periods
#> --HINT DISTRIBUTE_ON_KEY(subject_id)
#> select cohort_definition_id, tar_id, subject_id, start_date, end_date
#> into #TTAR
#> FROM (
#>  select tc1.cohort_definition_id,
#>      tar1.tar_id,
#>      subject_id,
#>      case
#>          when tar1.tar_start_index = 1 then
#>              case when dateadd(dd,tar1.tar_start_offset,tc1.cohort_start_date) < op1.observation_per
#>                  when dateadd(dd,tar1.tar_start_offset,tc1.cohort_start_date) >= op1.observation_per
#>              end
#>          when tar1.tar_start_index = 0 then
#>              case when dateadd(dd,tar1.tar_start_offset,tc1.cohort_end_date) < op1.observation_perio
#>                  when dateadd(dd,tar1.tar_start_offset,tc1.cohort_end_date) >= op1.observation_perio
#>              end
#>          else null --shouldnt get here if tar set properly
#>      end as start_date,
#>      case
#>          when tar1.tar_end_index = 1 then
#>              case when dateadd(dd,tar1.tar_end_offset,tc1.cohort_start_date) < op1.observation_perio
#>                  when dateadd(dd,tar1.tar_end_offset,tc1.cohort_start_date) >= op1.observation_perio
#>              end
#>          when tar1.tar_end_index = 0 then
#>              case when dateadd(dd,tar1.tar_end_offset,tc1.cohort_end_date) < op1.observation_period_
#>                  when dateadd(dd,tar1.tar_end_offset,tc1.cohort_end_date) >= op1.observation_period_
#>              end
#>          else null --shouldnt get here if tar set properly
```

```
#>      end as end_date
#>  from (select tar_id, tar_start_index, tar_start_offset, tar_end_index, tar_end_offset  from #tar_re
#>  (select cohort_definition_id, subject_id, cohort_start_date, cohort_end_date from demoCohortSchema.
#>  inner join mycdm.observation_period op1 on tc1.subject_id = op1.person_id
#>      and tc1.cohort_start_date >= op1.observation_period_start_date
#>      and tc1.cohort_start_date <= op1.observation_period_end_date
#> ) TAR
#> WHERE TAR.start_date <= TAR.end_date
#> ;
#>
#> /*
#> 2) create table to store era-fied at-risk periods
#>   UNION all periods that don't require erafying
#>   with era-fy records that require it
#> */
#>
#> --find the records that need to be era-fied
#> --era-building script for the 'TTAR_to_erafy' records
#> --insert records from era-building script into #TTAR_erafied
#> --HINT DISTRIBUTE_ON_KEY(subject_id)
#> select t1.cohort_definition_id, t1.tar_id, t1.subject_id, t1.start_date, t1.end_date
#> INTO #TTAR_to_erafy
#> from #TTAR t1
#> inner join #TTAR t2 on t1.cohort_definition_id = t2.cohort_definition_id
#>  and t1.tar_id = t2.tar_id
#>  and t1.subject_id = t2.subject_id
#>  and t1.start_date <= t2.end_date
#>  and t1.end_date >= t2.start_date
#>  and t1.start_date <> t2.start_date
#> ;
#>
#> --HINT DISTRIBUTE_ON_KEY(subject_id)
#> with cteEndDates (cohort_definition_id, tar_id, subject_id, end_date) AS
#> (
#>  SELECT
#>      cohort_definition_id,
#>      tar_id,
#>      subject_id,
#>      event_date as end_date
#>  FROM
#>  (
#>      SELECT cohort_definition_id,
#>          tar_id,
#>          subject_id,
#>          event_date,
#>          SUM(event_type) OVER (PARTITION BY cohort_definition_id, tar_id, subject_id ORDER BY event_
#>      FROM
#>      (
#>          SELECT
#>              cohort_definition_id,
#>              tar_id,
#>              subject_id,
#>              start_date AS event_date,
```

```
#>              -1 AS event_type
#>          FROM #TTAR_to_erafy
#>
#>          UNION ALL
#>
#>          SELECT
#>              cohort_definition_id,
#>              tar_id,
#>              subject_id,
#>              end_date AS event_date,
#>              1 AS event_type
#>          FROM #TTAR_to_erafy
#>      ) RAWDATA
#>  ) e
#>  WHERE interval_status = 0
#> ),
#> cteEnds (cohort_definition_id, tar_id, subject_id, start_date, end_date) AS
#> (
#>  SELECT c.cohort_definition_id,
#>      c.tar_id,
#>      c.subject_id,
#>      c.start_date,
#>      MIN(e.end_date) AS end_date
#>  FROM #TTAR_to_erafy c
#>  INNER JOIN cteEndDates e ON c.subject_id = e.subject_id
#>      AND c.cohort_definition_id = e.cohort_definition_id
#>      AND c.tar_id = e.tar_id
#>      AND e.end_date >= c.start_date
#>  GROUP BY  c.cohort_definition_id,
#>      c.tar_id,
#>      c.subject_id,
#>      c.start_date
#> )
#> select cohort_definition_id, tar_id, subject_id, min(start_date) as start_date, end_date
#> into #TTAR_era_overlaps
#> from cteEnds
#> group by cohort_definition_id, tar_id, subject_id, end_date
#> ;
#>
#>
#> --HINT DISTRIBUTE_ON_KEY(subject_id)
#> select cohort_definition_id, tar_id, subject_id, start_date, end_date
#> into #TTAR_erafied
#> FROM (
#>  select cohort_definition_id, tar_id, subject_id, start_date, end_date
#>  from #TTAR_era_overlaps
#>
#>  UNION ALL
#>
#>  --records that were already erafied and just need to be brought over directly
#>  select distinct t1.cohort_definition_id, t1.tar_id, t1.subject_id, t1.start_date, t1.end_date
#>  from #TTAR t1
#>  left join #TTAR t2 on t1.cohort_definition_id = t2.cohort_definition_id
```

```
#>      and t1.tar_id = t2.tar_id
#>      and t1.subject_id = t2.subject_id
#>      and t1.start_date <= t2.end_date
#>      and t1.end_date >= t2.start_date
#>      and t1.start_date <> t2.start_date
#>  where t2.subject_id IS NULL
#> ) T
#>
#> ;
#>
#>
#> create table #subgroup_person
#> (
#>  subgroup_id bigint NOT NULL,
#>  subject_id bigint NOT NULL,
#>  start_date date NOT NULL
#> );
#>
#> INSERT INTO #subgroup_person (subgroup_id, subject_id, start_date)
#> select distinct cast(1 as int) as subgroup_id, t1.subject_id, t1.start_date
#> FROM #TTAR_erafied t1
#> JOIN demoCohortSchema.cohort s1 on t1.subject_id = s1.subject_id
#>   and t1.start_date  >= s1.cohort_start_date
#>   and t1.start_date <= s1.cohort_end_date
#> WHERE s1.cohort_definition_id = 300
#> ;
#>
#>
#> /*
#> 3) create table to store era-fied excluded at-risk periods
#>   UNION all periods that don't require erafying
#>   with era-fy records that require it
#> */
#>
#> -- find excluded time from outcome cohorts and exclusion cohorts
#> -- note, clean window added to event end date
#> --HINT DISTRIBUTE_ON_KEY(subject_id)
#> select or1.outcome_id, oc1.subject_id, dateadd(dd,1,oc1.cohort_start_date) as cohort_start_date, dat
#> into #excluded_tar_cohort
#> from demoCohortSchema.cohort oc1
#> inner join (
#>  select outcome_id, outcome_cohort_definition_id, clean_window
#>  from #outcome_ref
#>  where outcome_id in (1)
#> ) or1 on oc1.cohort_definition_id = or1.outcome_cohort_definition_id
#> where dateadd(dd,or1.clean_window, oc1.cohort_end_date) >= dateadd(dd,1,oc1.cohort_end_date)
#>
#> union all
#>
#> SELECT or1.outcome_id, c1.subject_id, c1.cohort_start_date, c1.cohort_end_date
#> FROM demoCohortSchema.cohort c1
#> inner join (
#>  select outcome_id, excluded_cohort_definition_id
```

```
#>  from #outcome_ref
#>  where outcome_id in (1)
#> ) or1 on c1.cohort_definition_id = or1.excluded_cohort_definition_id
#> ;
#>
#> --HINT DISTRIBUTE_ON_KEY(subject_id)
#> select te1.cohort_definition_id as target_cohort_definition_id,
#>  te1.tar_id,
#>  ec1.outcome_id,
#>  ec1.subject_id,
#>  case when ec1.cohort_start_date > te1.start_date then ec1.cohort_start_date else te1.start_date end
#>  case when ec1.cohort_end_date < te1.end_date then ec1.cohort_end_date else te1.end_date end as end_
#> into #exc_TTAR_o
#> from #TTAR_erafied te1
#> inner join #excluded_tar_cohort ec1 on te1.subject_id = ec1.subject_id
#>  and ec1.cohort_start_date <= te1.end_date
#>  and ec1.cohort_end_date >= te1.start_date
#> ;
#>
#> --find the records that need to be era-fied
#>
#> --HINT DISTRIBUTE_ON_KEY(subject_id)
#> select t1.target_cohort_definition_id, t1.tar_id, t1.outcome_id, t1.subject_id, t1.start_date, t1.en
#> into #exc_TTAR_o_to_erafy
#> from #exc_TTAR_o t1
#> inner join #exc_TTAR_o t2 on t1.target_cohort_definition_id = t2.target_cohort_definition_id
#>  and t1.tar_id = t2.tar_id
#>  and t1.outcome_id = t2.outcome_id
#>  and t1.subject_id = t2.subject_id
#>  and t1.start_date < t2.end_date
#>  and t1.end_date > t2.start_date
#>  and (t1.start_date <> t2.start_date or t1.end_date <> t2.end_date)
#> ;
#>
#> --era-building script for the 'exc_TTAR_o_to_erafy ' records
#> --insert records from era-building script into #TTAR_erafied
#>
#> --HINT DISTRIBUTE_ON_KEY(subject_id)
#> with cteEndDates (target_cohort_definition_id, tar_id, outcome_id, subject_id, end_date) AS
#> (
#>  SELECT
#>      target_cohort_definition_id,
#>      tar_id,
#>      outcome_id,
#>      subject_id,
#>      event_date as end_date
#>  FROM
#>  (
#>      SELECT
#>          target_cohort_definition_id,
#>          tar_id,
#>          outcome_id,
#>          subject_id,
```

```
#>          event_date,
#>          SUM(event_type) OVER (PARTITION BY target_cohort_definition_id, tar_id, outcome_id, subject_
#>      FROM
#>      (
#>          SELECT
#>              target_cohort_definition_id,
#>              tar_id,
#>              outcome_id,
#>              subject_id,
#>              start_date AS event_date,
#>              -1 AS event_type
#>          FROM #exc_TTAR_o_to_erafy
#>
#>          UNION ALL
#>
#>          SELECT
#>              target_cohort_definition_id,
#>              tar_id,
#>              outcome_id,
#>              subject_id,
#>              end_date AS event_date,
#>              1 AS event_type
#>          FROM #exc_TTAR_o_to_erafy
#>      ) RAWDATA
#>  ) e
#>  WHERE interval_status = 0
#> ),
#> cteEnds (target_cohort_definition_id, tar_id, outcome_id, subject_id, start_date, end_date) AS
#> (
#>  SELECT c.target_cohort_definition_id,
#>   c.tar_id,
#>   c.outcome_id,
#>       c.subject_id,
#>      c.start_date,
#>      MIN(e.end_date) AS end_date
#>  FROM #exc_TTAR_o_to_erafy c
#>  INNER JOIN cteEndDates e ON c.subject_id = e.subject_id
#>      AND c.target_cohort_definition_id = e.target_cohort_definition_id
#>      AND c.tar_id = e.tar_id
#>      AND c.outcome_id = e.outcome_id
#>      AND e.end_date >= c.start_date
#>  GROUP BY  c.target_cohort_definition_id,
#>   c.tar_id,
#>   c.outcome_id,
#>       c.subject_id,
#>      c.start_date
#> )
#> select target_cohort_definition_id, tar_id, outcome_id, subject_id, min(start_date) as start_date, e
#> into #ex_TTAR_o_overlaps
#> from cteEnds
#> group by target_cohort_definition_id, tar_id, outcome_id, subject_id, end_date
#> ;
#>
```

```
#> --HINT DISTRIBUTE_ON_KEY(subject_id)
#> select target_cohort_definition_id, tar_id, outcome_id, subject_id, start_date, end_date
#> into #exc_TTAR_o_erafied
#> from #ex_TTAR_o_overlaps
#>
#> UNION ALL
#>
#> --records that were already erafied and just need to be brought over directly
#> select distinct t1.target_cohort_definition_id, t1.tar_id, t1.outcome_id, t1.subject_id, t1.start_da
#> from #exc_TTAR_o t1
#> left join #exc_TTAR_o t2 on t1.target_cohort_definition_id = t2.target_cohort_definition_id
#>  and t1.tar_id = t2.tar_id
#>  and t1.outcome_id = t2.outcome_id
#>  and t1.subject_id = t2.subject_id
#>  and t1.start_date < t2.end_date
#>  and t1.end_date > t2.start_date
#>  and (t1.start_date <> t2.start_date or t1.end_date <> t2.end_date)
#> where t2.subject_id IS NULL
#> ;
#>
#>
#> -- 4) calculate pre-exclude outcomes and outcomes
#> -- calculate pe_outcomes and outcomes by T, TAR, O, Subject, TAR start
#>
#> --HINT DISTRIBUTE_ON_KEY(subject_id)
#> select t1.cohort_definition_id as target_cohort_definition_id,
#>  t1.tar_id,
#>  t1.subject_id,
#>  t1.start_date,
#>  o1.outcome_id,
#>  count_big(o1.subject_id) as pe_outcomes,
#>  SUM(case when eo.tar_id is null then 1 else 0 end) as num_outcomes
#> into #outcome_smry
#> from #TTAR_erafied t1
#> inner join (
#>  select oref.outcome_id, oc.subject_id, oc.cohort_start_date
#>  from demoCohortSchema.cohort oc
#>  JOIN #outcome_ref oref on oc.cohort_definition_id = oref.outcome_cohort_definition_id
#>  where oref.outcome_id in (1)
#> ) o1 on t1.subject_id = o1.subject_id
#>  and t1.start_date <= o1.cohort_start_date
#>  and t1.end_date >= o1.cohort_start_date
#> left join #exc_TTAR_o_erafied eo on t1.cohort_definition_id = eo.target_cohort_definition_id
#>  and t1.tar_id = eo.tar_id
#>  and o1.outcome_id = eo.outcome_id
#>  and o1.subject_id = eo.subject_id
#>  and eo.start_date <= o1.cohort_start_date
#>  and eo.end_date >= o1.cohort_start_date
#> group by t1.cohort_definition_id, t1.tar_id, t1.subject_id, t1.start_date, o1.outcome_id
#> ;
#>
#> -- 5) calculate exclsion time per T/O/TAR/Subject/start_date
#>
```

```
#> --HINT DISTRIBUTE_ON_KEY(subject_id)
#> select t1.cohort_definition_id as target_cohort_definition_id,
#>  t1.tar_id,
#>  t1.subject_id,
#>  t1.start_date,
#>  et1.outcome_id,
#>  sum(datediff(dd,et1.start_date, et1.end_date) + 1) as person_days
#> INTO #excluded_person_days
#> from #TTAR_erafied t1
#> inner join #exc_TTAR_o_erafied et1 on t1.cohort_definition_id = et1.target_cohort_definition_id
#>  and t1.tar_id = et1.tar_id
#>  and t1.subject_id = et1.subject_id
#>  and t1.start_date <= et1.start_date
#>  and t1.end_date >= et1.end_date
#> group by t1.cohort_definition_id,
#>   t1.tar_id,
#>   t1.subject_id,
#>  t1.start_date,
#>  et1.outcome_id
#> ;
#>
#> /*
#> 6) generate raw result table with T/O/TAR/subject_id,start_date, pe_at_risk (datediff(d,start,end),
#>    and attach age/gender/year columns
#> */
#>
#> --HINT DISTRIBUTE_ON_KEY(subject_id)
#> select t1.target_cohort_definition_id,
#>  o1.outcome_id,
#>  t1.tar_id,
#>  t1.subject_id,
#>  t1.start_date,
#>  ag.age_id,
#>  t1.gender_id,
#>  t1.start_year,
#>  datediff(dd,t1.start_date, t1.end_date) + 1 as pe_person_days,
#>  datediff(dd,t1.start_date, t1.end_date) + 1 - coalesce(te1.person_days,0) as person_days,
#>  coalesce(os1.pe_outcomes,0) as pe_outcomes,
#>  coalesce(os1.num_outcomes,0) as outcomes
#> into #incidence_raw
#> from (
#>  select te.cohort_definition_id as target_cohort_definition_id,
#>      te.tar_id,
#>      te.subject_id,
#>      te.start_date,
#>      te.end_date,
#>      YEAR(te.start_date) - p.year_of_birth as age,
#>      p.gender_concept_id as gender_id,
#>      YEAR(te.start_date) as start_year
#>  from #TTAR_erafied te
#>  join mycdm.person p on te.subject_id = p.person_id
#> ) t1
#> cross join (select outcome_id from #outcome_ref where outcome_id in (1)) o1
```

```
#> left join #excluded_person_days te1 on t1.target_cohort_definition_id = te1.target_cohort_definition
#>  and t1.tar_id = te1.tar_id
#>  and t1.subject_id = te1.subject_id
#>  and t1.start_date = te1.start_date
#>  and o1.outcome_id = te1.outcome_id
#> left join #outcome_smry os1 on t1.target_cohort_definition_id = os1.target_cohort_definition_id
#>  and t1.tar_id = os1.tar_id
#>  and t1.subject_id = os1.subject_id
#>  and t1.start_date = os1.start_date
#>  and o1.outcome_id = os1.outcome_id
#> left join #age_group ag on t1.age >= coalesce(ag.min_age, -999) and t1.age < coalesce(ag.max_age, 99
#> ;
#>
#> -- 7) Create analysis_ref to produce each T/O/TAR/S combo
#>
#> select t1.target_cohort_definition_id,
#>   t1.target_name,
#>  tar1.tar_id,
#>  tar1.tar_start_offset,
#>  tar1.tar_start_index,
#>  tar1.tar_end_offset,
#>  tar1.tar_end_index,
#>  s1.subgroup_id,
#>  s1.subgroup_name,
#>  o1.outcome_id,
#>  o1.outcome_cohort_definition_id,
#>  o1.outcome_name,
#>  o1.clean_window
#> into #tscotar_ref
#> from (select target_cohort_definition_id, target_name from #target_ref where target_cohort_definitio
#>  (select tar_id, tar_start_offset, tar_start_index, tar_end_offset, tar_end_index from #tar_ref wher
#>  (select subgroup_id, subgroup_name from #subgroup_ref) s1,
#>  (select outcome_id, outcome_cohort_definition_id, outcome_name, clean_window from #outcome_ref wher
#> ;
#>
#> -- 8) perform rollup to calculate IR / IP at the T/O/TAR/S/[age|gender|year] level for 'all' and eac
#> -- and aggregate to the selected levels
#> with incidence_w_subgroup (subgroup_id, target_cohort_definition_id, outcome_id, tar_id, subject_id,
#> (
#>  -- the 'all' group
#>  select cast(0 as int) as subgroup_id, ir.target_cohort_definition_id, ir.outcome_id, ir.tar_id,
#>      ir.subject_id, ir.age_id, ir.gender_id, ir.start_year,
#>      ir.pe_person_days, ir.person_days, ir.pe_outcomes, ir.outcomes
#>  from #incidence_raw ir
#>
#>  UNION ALL
#>
#>  -- select the individual subgroup members using the subgruop_person table
#>  select s.subgroup_id as subgroup_id, ir.target_cohort_definition_id, ir.outcome_id, ir.tar_id,
#>      ir.subject_id, ir.age_id, ir.gender_id, ir.start_year,
#>      ir.pe_person_days, ir.person_days, ir.pe_outcomes, ir.outcomes
#>  from #incidence_raw ir
#>  join #subgroup_person s on ir.subject_id = s.subject_id and ir.start_date = s.start_date
```

```
#> )
#> select target_cohort_definition_id, tar_id, subgroup_id, outcome_id, age_id, gender_id, start_year,
#>  persons_at_risk_pe, persons_at_risk, person_days_pe, person_days, person_outcomes_pe, person_outcom
#> into #incidence_subgroups
#> from (
#>  select irs.target_cohort_definition_id,
#>      irs.tar_id,
#>      irs.subgroup_id,
#>      irs.outcome_id,
#>      cast (null as int) as age_id,
#>      cast (null as int) as gender_id,
#>      cast (null as int) as start_year,
#>      count_big(distinct irs.subject_id) as persons_at_risk_pe,
#>      count_big(distinct case when irs.person_days > 0 then irs.subject_id end) as persons_at_risk,
#>      sum(cast(irs.pe_person_days as bigint)) as person_days_pe,
#>      sum(cast(irs.person_days as bigint)) as person_days,
#>      count_big(distinct case when irs.pe_outcomes > 0 then irs.subject_id end) as person_outcomes_pe
#>      count_big(distinct case when irs.outcomes > 0 then irs.subject_id end) as person_outcomes,
#>      sum(cast(irs.pe_outcomes as bigint)) as outcomes_pe,
#>      sum(cast(irs.outcomes as bigint)) as outcomes
#>  from incidence_w_subgroup irs
#>  group by irs.target_cohort_definition_id, irs.tar_id, irs.subgroup_id, irs.outcome_id
#>
#> ) IR;
#>
#> insert into myresults.incidence_summary (ref_id, database_name, target_cohort_definition_id, target_
#>  tar_id, tar_start_offset, tar_start_index, tar_end_offset, tar_end_index,
#>  subgroup_id, subgroup_name,
#>  outcome_id, outcome_cohort_definition_id, outcome_name, clean_window,
#>  age_id, age_group_name, gender_id, gender_name, start_year,
#>  persons_at_risk_pe, persons_at_risk, person_days_pe, person_days,
#>  person_outcomes_pe, person_outcomes, outcomes_pe, outcomes,
#>  incidence_proportion_p100p, incidence_rate_p100py)
#> select CAST(1 as int) as ref_id, '@databaseName' as database_name, tref.target_cohort_definition_id,
#>  tref.tar_id, tref.tar_start_offset, tref.tar_start_index, tref.tar_end_offset, tref.tar_end_index,
#>  tref.subgroup_id, tref.subgroup_name,
#>  tref.outcome_id, tref.outcome_cohort_definition_id, tref.outcome_name, tref.clean_window,
#>  irs.age_id, ag.group_name, irs.gender_id, c.concept_name as gender_name, irs.start_year,
#>  coalesce(irs.persons_at_risk_pe, 0) as persons_at_risk_pe,
#>  coalesce(irs.persons_at_risk, 0) as persons_at_risk,
#>  coalesce(irs.person_days_pe, 0) as  person_days_pe,
#>  coalesce(irs.person_days, 0) as person_days,
#>  coalesce(irs.person_outcomes_pe, 0) as person_outcomes_pe,
#>  coalesce(irs.person_outcomes, 0) as person_outcomes,
#>  coalesce(irs.outcomes_pe, 0) as outcomes_pe,
#>  coalesce(irs.outcomes, 0) as outcomes,
#>  case when coalesce(irs.persons_at_risk, 0) > 0 then
#>      (100.0 * cast(coalesce(irs.person_outcomes,0) as float) / (cast(coalesce(irs.persons_at_risk, 0
#>  end as incidence_proportion_p100p,
#>  case when coalesce(irs.person_days,0) > 0 then
#>      (100.0 * cast(coalesce(irs.outcomes,0) as float) / (cast(coalesce(irs.person_days,0) as float)
#>  end AS incidence_rate_p100py
#> from #tscotar_ref tref
```

```
#> left join #incidence_subgroups irs on tref.target_cohort_definition_id = irs.target_cohort_definitio
#>  and tref.tar_id = irs.tar_id
#>  and tref.subgroup_id = irs.subgroup_id
#>  and tref.outcome_id = irs.outcome_id
#> left join #age_group ag on ag.age_id = irs.age_id
#> left join mycdm.concept c on c.concept_id = irs.gender_id
#> ;
#>
#>
#> -- CLEANUP TEMP TABLES
#> DROP TABLE #TTAR;
#> DROP TABLE #TTAR_to_erafy;
#> DROP TABLE #TTAR_era_overlaps;
#> DROP TABLE #TTAR_erafied;
#> DROP TABLE #subgroup_person;
#> DROP TABLE #excluded_tar_cohort;
#> DROP TABLE #exc_TTAR_o;
#> DROP TABLE #ex_TTAR_o_overlaps;
#> DROP TABLE #exc_TTAR_o_to_erafy;
#> DROP TABLE #exc_TTAR_o_erafied;
#> DROP TABLE #outcome_smry;
#> DROP TABLE #excluded_person_days;
#> DROP TABLE #incidence_raw;
#> DROP TABLE #tscotar_ref;
#> DROP TABLE #incidence_subgroups;
#>
#> --
#> -- End analysis 0
#> --
#>
#> DROP TABLE #target_ref;
#> DROP TABLE #tar_ref;
#> DROP TABLE #outcome_ref;
#> DROP TABLE #subgroup_ref;
#> DROP TABLE #age_group;
```

5.2 Render SQL with paramaters and execute

With the previous analysis design and options used to generate the analysisSql, the next step is to render the SQL to provide any remaining parameters, translate, and execute on the database:

```
analysisSql <- SqlRender::render(analysisSql, "sourceName" = "OptumDOD")
analysisSql <- SqlRender::translate(analysisSql, "postgresql")

cat(analysisSql)

conn <- DatabaseConnector::connect(connectionDetails)
DatabaseConnector::executeSql(conn, paste0("DELETE FROM myresults.incidence_summary WHERE ref_id = ", bu
DatabaseConnector::executeSql(conn, analysisSql)
DatabaseConnector::disconnect(conn)
```

5.3 Using Temp Tables

Sometimes, it is not convenient or possible to create dedicated tables to store the results. Instead, the useTempTables option can be used to place the incidence results into a temp table 'incidence_summary', where they can be ETL'd to another table or exported to a CSV.

The following example demonstrates the additional steps that are necessary if you want to use temp tables:

```
# given the prior irDesign constructed from the previous example
buildOptions <- CohortIncidence::buildOptions(cohortTable = "demoCohortSchema.cohort",
                                              cdmDatabaseSchema = "mycdm",
                                              useTempTables = T,
                                              refId = 2)

analysisSql <- CohortIncidence::buildQuery(incidenceDesign = as.character(jsonlite::toJSON(irDesign)),
                                           buildOptions = buildOptions)
analysisSql <- SqlRender::render(analysisSql, "sourceName" = "OptumDOD")
analysisSql <- SqlRender::translate(analysisSql, "postgresql")

# if we are using temp tables, the steps to execute the analysis are
#   1) create result temp tables
#   2) execute the analysis query, placing the results into the temp table incidence_summary
#   3) Extract/copy the results from the temp tables
#   4) clean up temp tables

conn <- DatabaseConnector::connect(connectionDetails)

tempDDL <- SqlRender::translate(CohortIncidence::getResultsDdl(useTempTables=T), "postgresql")
DatabaseConnector::executeSql(conn, tempDDL)
DatabaseConnector::executeSql(conn, analysisSql)

# In this example, copy to a permanent table from the temp table, but the results could be downloaded t
exportSql <- SqlRender::translate("insert into mySchema.prefix_incidence_summary select * from #inciden
DatabaseConnector::executeSql(conn, exportSql)
# or download the results to a dataframe
results <- DatabaseConnector::querySql(conn, SqlRender::translate("select * from #incidence_summary", "

# use the getCleanupSql to fetch the DROP TABLE expressions for the tables that were created in tempDDL
cleanupSql <- SqlRender::translate(CohortIncidence::getCleanupSql(useTempTables=T), "postgresql")
DatabaseConnector::executeSql(conn, cleanupSql)

DatabaseConnector::dbDisconnect(conn)
```