How to Use CohortAlgebra R Package

Gowtham A. Rao

2022-08-07

Contents

1 Introduction

(This package is NOT part of HADES.)

The idea behind this package is to allow the construction of new cohorts from previously instantiated cohorts in the cohort table. All cohorts in OHDSI have a standard definition: "A cohort is a set of persons who satisfy one or more inclusion criteria for a duration of time."

- One person may belong to multiple cohorts
- One person may belong to the same cohort for multiple different time periods
- One person may not belong to the same cohort multiple times during the same period of time
- A cohort may have zero or more members

This is represented in a cohort table as cohort_definition_id, subject_id, cohort_start_date and cohort_end_date. For more details about the concept of a cohort please review The Book of OHDSI.

This package allows the creation of new cohorts from previously instantiated cohort table using cohort algebra (similar to temporal set algebra). The output is one or more new cohorts.

1.1 Installation

- This is an installable R-package that may be installed as follows:

```
remotes::install_github("OHDSI/CohortAlgebra")
```

```
#> Consider adding 'DATABASECONNECTOR_JAR_FOLDER='C:/Users/admin_grao9/Documents\RtmpKaUYmB\jdbcDrivers
#> DatabaseConnector postgresql JDBC driver downloaded to 'C:/Users/admin_grao9/Documents\RtmpKaUYmB\jd
```

1.2 Cohort UNION

- Given two or more cohorts, an UNION operator on these cohorts creates a new cohort with continuous days the persons was present in any of the cohorts.

```
cohort <- dplyr::tibble(
  cohortDefinitionId = c(1, 2, 2),
  subjectId = c(1, 1, 1),
  cohortStartDate = c(
    as.Date("2022-01-01"),
    as.Date("2022-02-10"),
    as.Date("2022-08-15")
  ),
  cohortEndDate = c(
    as.Date("2022-03-01"),
    as.Date("2022-05-10"),
    as.Date("2022-12-30")
  )
)
cohort
```

```
#> # A tibble: 3 x 4
#>   cohortDefinitionId subjectId cohortStartDate cohortEndDate
#>                <dbl>     <dbl> <date>          <date>
#> 1                  1         1 2022-01-01      2022-03-01
#> 2                  2         1 2022-02-10      2022-05-10
#> 3                  2         1 2022-08-15      2022-12-30
```

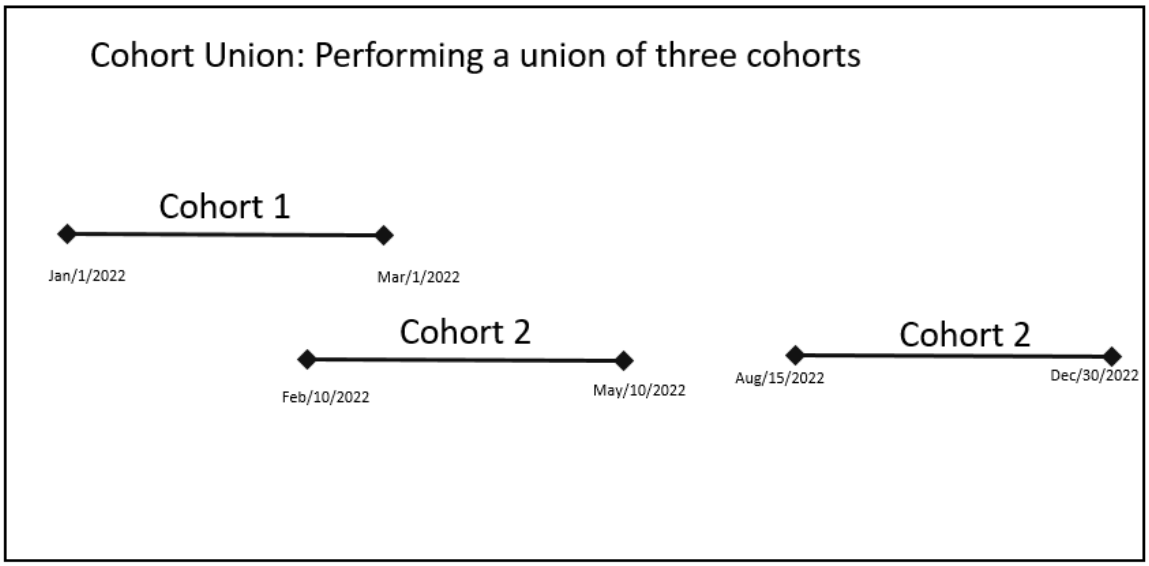
```
#> Connecting using PostgreSQL driver
```

The union of the three cohorts is expected to give us

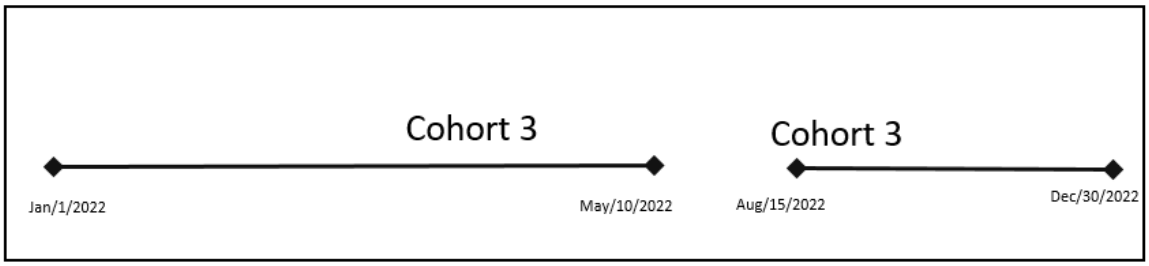
```
cohortExpected <- dplyr::tibble(
  cohortDefinitionId = c(3, 3),
  subjectId = c(1, 1),
  cohortStartDate = c(as.Date("2022-01-01"), as.Date("2022-08-15")),
  cohortEndDate = c(as.Date("2022-05-10"), as.Date("2022-12-30"))
)
cohortExpected
```

```
#> # A tibble: 2 x 4
#>   cohortDefinitionId subjectId cohortStartDate cohortEndDate
#>                <dbl>     <dbl> <date>          <date>
#> 1                  3         1 2022-01-01      2022-05-10
#> 2                  3         1 2022-08-15      2022-12-30
```

Input



Output



To perform Cohort Union, we use the unionCohorts function. This function requires as an input a data.frame called oldToNewCohortId. Here we specify the cohort id's of the cohorts we want to union. The newCohortId is the cohortId of the resultant cohort. The oldCohortId are cohorts that are already in the cohort table.

```
oldToNewCohortId <-
  dplyr::tibble(oldCohortId = c(1, 2, 2),
                newCohortId = c(3, 3, 3))

CohortAlgebra::unionCohorts(
  connection = connection,
  cohortDatabaseSchema = cohortDatabaseSchema,
  cohortTable = tableName,
  oldToNewCohortId = oldToNewCohortId
)
```

Now we will have a new cohortId '3' which is the union of cohortId's 1 and 2.

```
data <-
  DatabaseConnector::renderTranslateQuerySql(
    connection = connection,
    sql = paste0(
      "SELECT * FROM @cohort_database_schema.@table_name
```

```
        where cohort_definition_id = 3
        order by cohort_definition_id, subject_id, cohort_start_date;"
    ),
    cohort_database_schema = cohortDatabaseSchema,
    table_name = tableName,
    snakeCaseToCamelCase = TRUE
  ) %>%
  dplyr::tibble()

data
```

```
#> # A tibble: 2 x 4
#>   cohortDefinitionId subjectId cohortStartDate cohortEndDate
#>                <dbl>     <dbl> <date>          <date>
#> 1                  3         1 2022-01-01      2022-05-10
#> 2                  3         1 2022-08-15      2022-12-30
```

Note: if the target cohort table had a cohort with cohortId = 3, before running the union function - this would cause a conflict. In those cases, the union function would not run. We can purge all records for cohortId = 3 from the target cohort table. The parameter purgeConflicts will delete any cohort records in the cohort table where cohortId is the cohortId of the newCohort.

1.3 InterSect Cohort

- Given two or more cohorts, an INTERSECT operator on these cohorts creates a new cohort with continuous days the persons was present in ALL of the cohorts.

1.3.1 Intersect cohort example 1

Input:

```
cohort <- dplyr::tibble(
  cohortDefinitionId = c(1, 2),
  subjectId = c(1, 1),
  cohortStartDate = c(
    as.Date("2022-01-01"),
    as.Date("2021-12-15")
  ),
  cohortEndDate = c(
    as.Date("2022-01-15"),
    as.Date("2022-01-30")
  )
)
cohort
```

```
#> # A tibble: 2 x 4
#>   cohortDefinitionId subjectId cohortStartDate cohortEndDate
#>                <dbl>     <dbl> <date>          <date>
#> 1                  1         1 2022-01-01      2022-01-15
#> 2                  2         1 2021-12-15      2022-01-30
```

```
CohortAlgebra::intersectCohorts(
  connection = connection,
  cohortDatabaseSchema = cohortDatabaseSchema,
  cohortTable = tableName,
  cohortIds = c(1, 2),
  newCohortId = 3
)
```

```
#> Currently in a tryCatch or withCallingHandlers block, so unable to add global calling handlers. Paral
#> Currently in a tryCatch or withCallingHandlers block, so unable to add global calling handlers. Paral
```

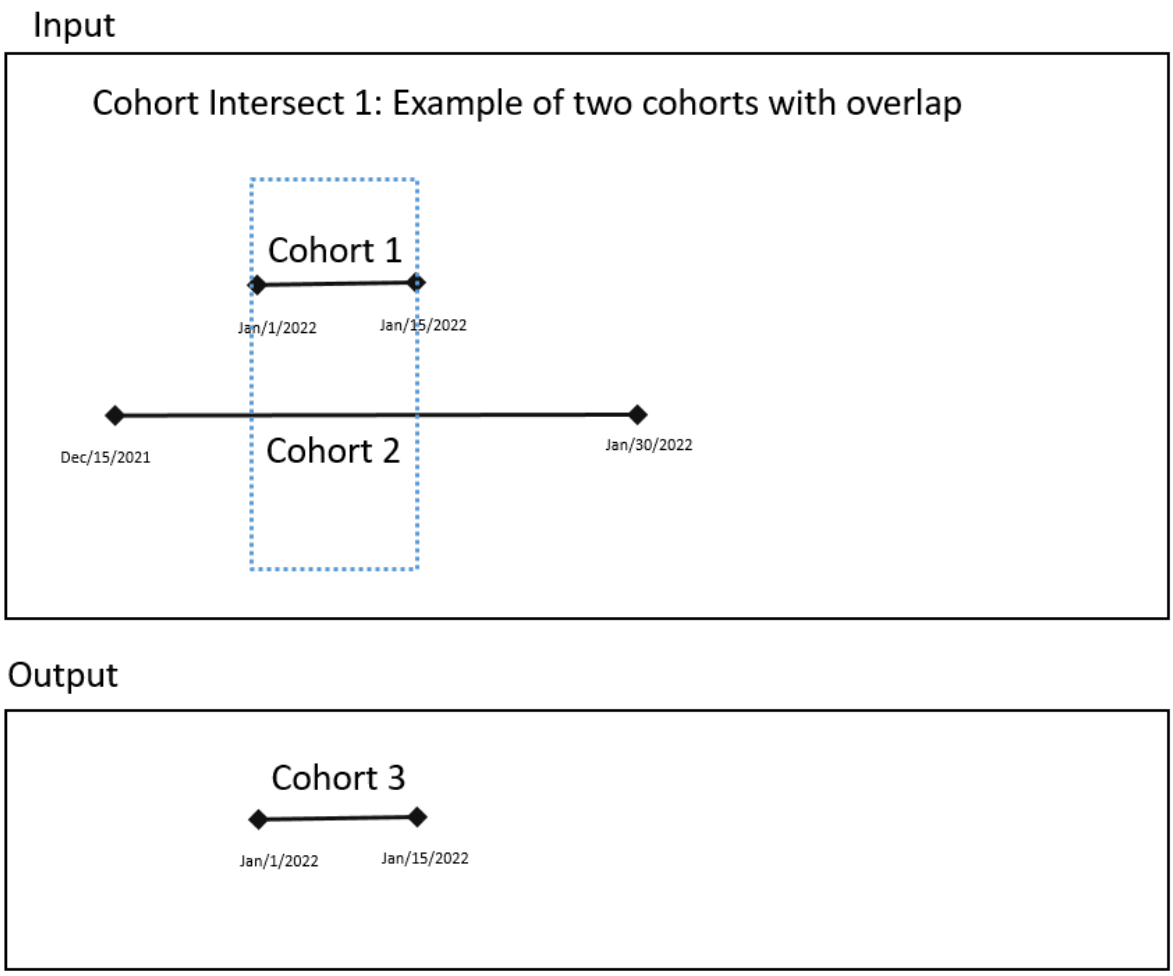


Figure 1: Cohort Intersect 1

Output

```
#> # A tibble: 1 x 4
#>   cohortDefinitionId subjectId cohortStartDate cohortEndDate
#>                <dbl>     <dbl> <date>          <date>
#> 1                  3         1 2022-01-01      2022-01-15
```

1.3.2 Intersect cohort example 2

Input:

```
cohort <- dplyr::tibble(
  cohortDefinitionId = c(1, 2, 2),
  subjectId = c(1, 1, 1),
  cohortStartDate = c(
    as.Date("2022-01-01"),
    as.Date("2021-12-15"),
    as.Date("2022-01-10")
  ),
  cohortEndDate = c(
    as.Date("2022-01-15"),
    as.Date("2022-01-05"),
    as.Date("2022-01-30")
  )
)
cohort
```

```
#> # A tibble: 3 x 4
#>   cohortDefinitionId subjectId cohortStartDate cohortEndDate
#>                <dbl>     <dbl> <date>          <date>
#> 1                  1         1 2022-01-01      2022-01-15
#> 2                  2         1 2021-12-15      2022-01-05
#> 3                  2         1 2022-01-10      2022-01-30
```

```
CohortAlgebra::intersectCohorts(
  connection = connection,
  cohortDatabaseSchema = cohortDatabaseSchema,
  cohortTable = tableName,
  cohortIds = c(1, 2),
  newCohortId = 3
)
```

```
#> Currently in a tryCatch or withCallingHandlers block, so unable to add global calling handlers. Paral
#> Currently in a tryCatch or withCallingHandlers block, so unable to add global calling handlers. Paral
```

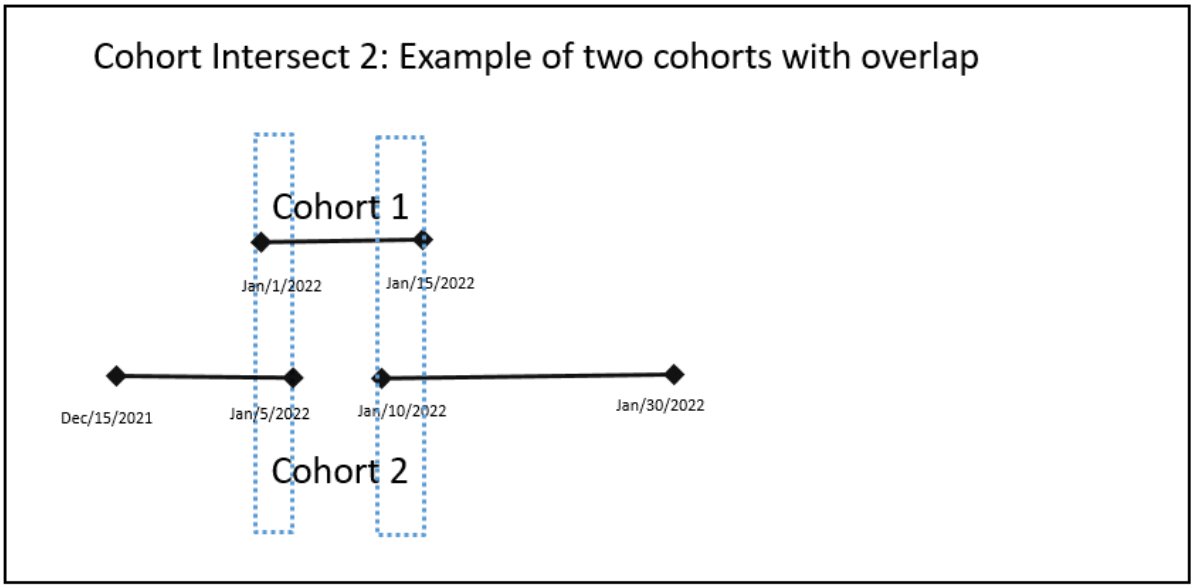
Output

```
#> # A tibble: 2 x 4
#>   cohortDefinitionId subjectId cohortStartDate cohortEndDate
#>                <dbl>     <dbl> <date>          <date>
#> 1                  3         1 2022-01-01      2022-01-05
#> 2                  3         1 2022-01-10      2022-01-15
```

1.3.3 Intersect cohort example 3

Input:

Input



Output

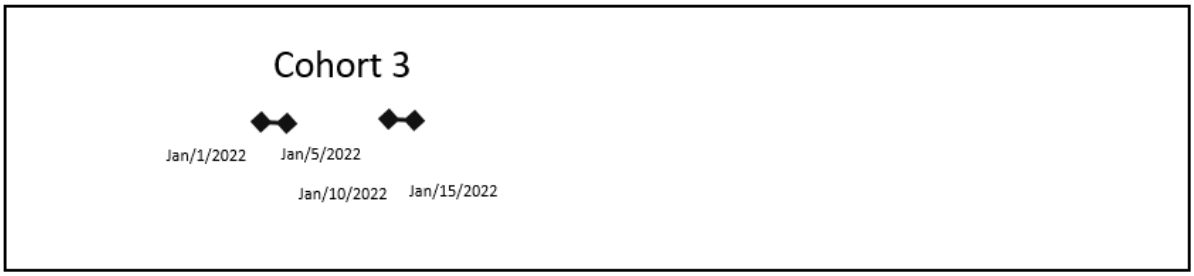


Figure 2: Cohort Intersect 2

```
cohort <- dplyr::tibble(
  cohortDefinitionId = c(1, 2, 3),
  subjectId = c(1, 1, 1),
  cohortStartDate = c(
    as.Date("2022-01-01"),
    as.Date("2021-12-15"),
    as.Date("2022-03-01")
  ),
  cohortEndDate = c(
    as.Date("2022-01-15"),
    as.Date("2022-01-30"),
    as.Date("2022-03-15")
  )
)
cohort
```

```
#> # A tibble: 3 x 4
#>   cohortDefinitionId subjectId cohortStartDate cohortEndDate
#>                <dbl>     <dbl> <date>          <date>
#> 1                  1         1 2022-01-01      2022-01-15
#> 2                  2         1 2021-12-15      2022-01-30
#> 3                  3         1 2022-03-01      2022-03-15
```

```
CohortAlgebra::intersectCohorts(
  connection = connection,
  cohortDatabaseSchema = cohortDatabaseSchema,
  cohortTable = tableName,
  cohortIds = c(1, 2, 3),
  newCohortId = 4
)
```

```
#> Currently in a tryCatch or withCallingHandlers block, so unable to add global calling handlers. Paral
#> Currently in a tryCatch or withCallingHandlers block, so unable to add global calling handlers. Paral
```

Output

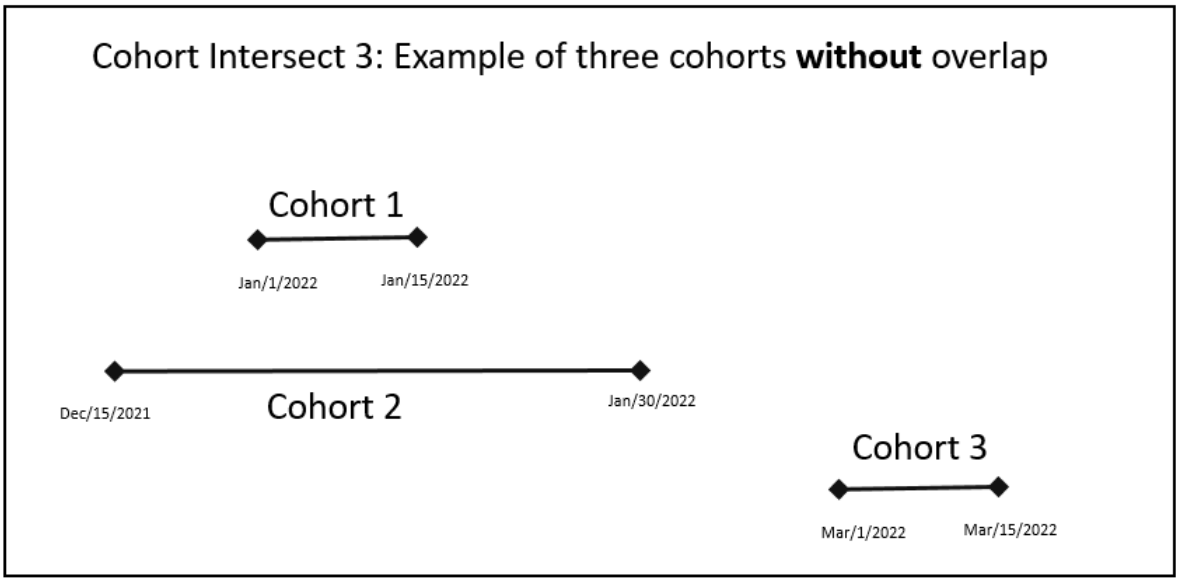
```
#> # A tibble: 0 x 4
#> # ... with 4 variables: cohortDefinitionId <dbl>, subjectId <dbl>, cohortStartDate <date>, cohortEndl
#> # i Use ‘colnames()‘ to see all variable names
```

1.3.4 Intersect cohort example 4

Input:

```
cohort <- dplyr::tibble(
  cohortDefinitionId = c(1, 2),
  subjectId = c(1, 1),
  cohortStartDate = c(
    as.Date("2022-01-01"),
    as.Date("2021-12-15")
  ),
```

Input



Output

NULL Cohort – i.e. no cohort is created

Figure 3: Cohort Intersect 3

```
  cohortEndDate = c(
    as.Date("2022-01-15"),
    as.Date("2022-01-30")
  )
)
cohort
```

```
#> # A tibble: 2 x 4
#>   cohortDefinitionId subjectId cohortStartDate cohortEndDate
#>                <dbl>     <dbl> <date>          <date>
#> 1                  1         1 2022-01-01      2022-01-15
#> 2                  2         1 2021-12-15      2022-01-30
```

```
CohortAlgebra::intersectCohorts(
  connection = connection,
  cohortDatabaseSchema = cohortDatabaseSchema,
  cohortTable = tableName,
  cohortIds = c(1, 2, 3),
  newCohortId = 4
)
```

```
#> Currently in a tryCatch or withCallingHandlers block, so unable to add global calling handlers. Paral
#> Currently in a tryCatch or withCallingHandlers block, so unable to add global calling handlers. Paral
```

Output

```
#> # A tibble: 0 x 4
#> # ... with 4 variables: cohortDefinitionId <dbl>, subjectId <dbl>, cohortStartDate <date>, cohortEndl
#> # i Use `colnames()` to see all variable names
```

1.3.5 Intersect cohort example 5

Input:

```
cohort <- dplyr::tibble(
  cohortDefinitionId = c(1, 2),
  subjectId = c(1, 1),
  cohortStartDate = c(
    as.Date("2022-01-01"),
    as.Date("2022-01-01")
  ),
  cohortEndDate = c(
    as.Date("2022-01-01"),
    as.Date("2022-01-02")
  )
)
cohort
```

```
#> # A tibble: 2 x 4
#>   cohortDefinitionId subjectId cohortStartDate cohortEndDate
#>                <dbl>     <dbl> <date>          <date>
#> 1                  1         1 2022-01-01      2022-01-01
#> 2                  2         1 2022-01-01      2022-01-02
```

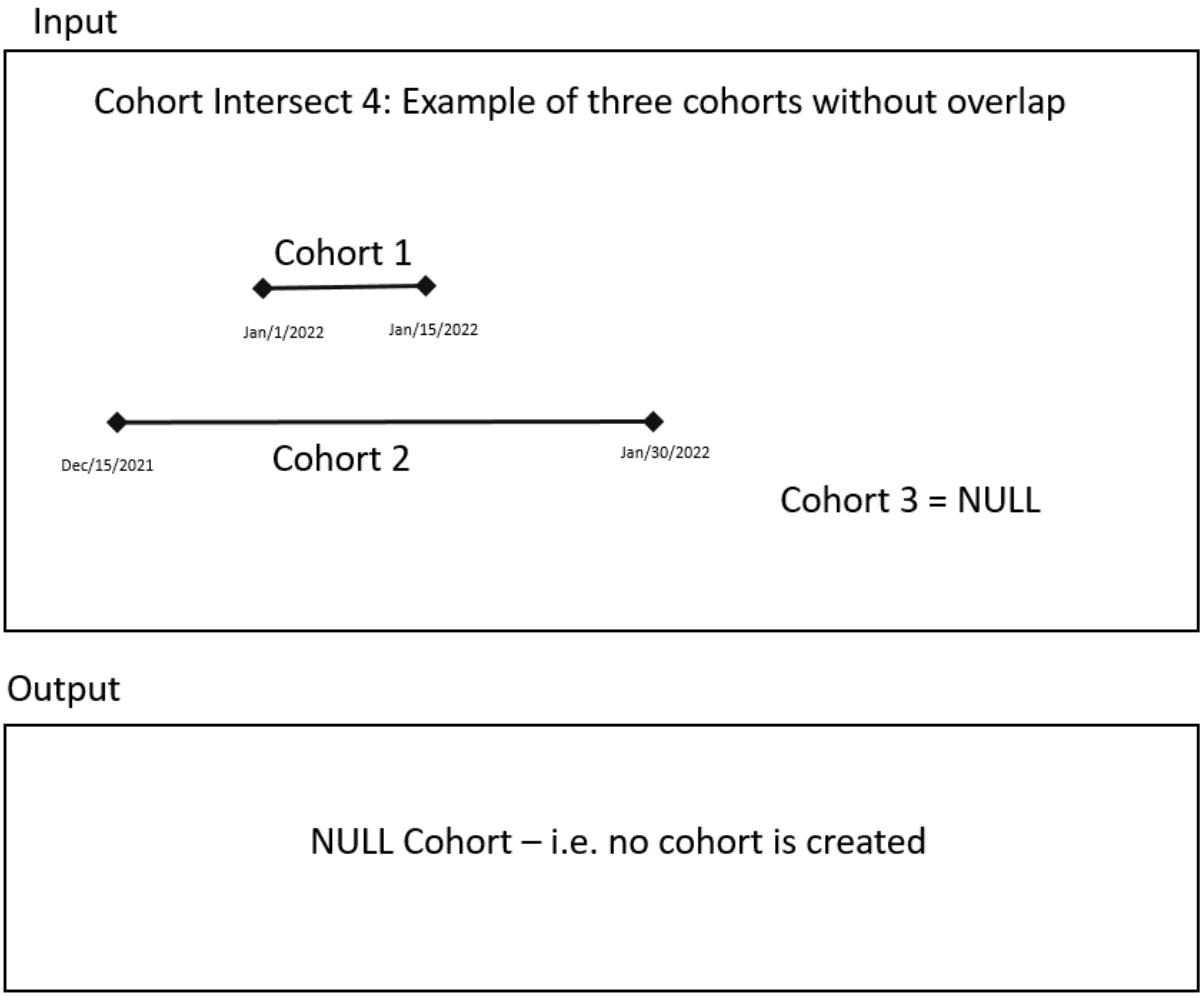


Figure 4: Cohort Intersect 4

```
CohortAlgebra::intersectCohorts(
  connection = connection,
  cohortDatabaseSchema = cohortDatabaseSchema,
  cohortTable = tableName,
  cohortIds = c(1, 2),
  newCohortId = 3
)
```

```
#> Currently in a tryCatch or withCallingHandlers block, so unable to add global calling handlers. Paral
#> Currently in a tryCatch or withCallingHandlers block, so unable to add global calling handlers. Paral
```

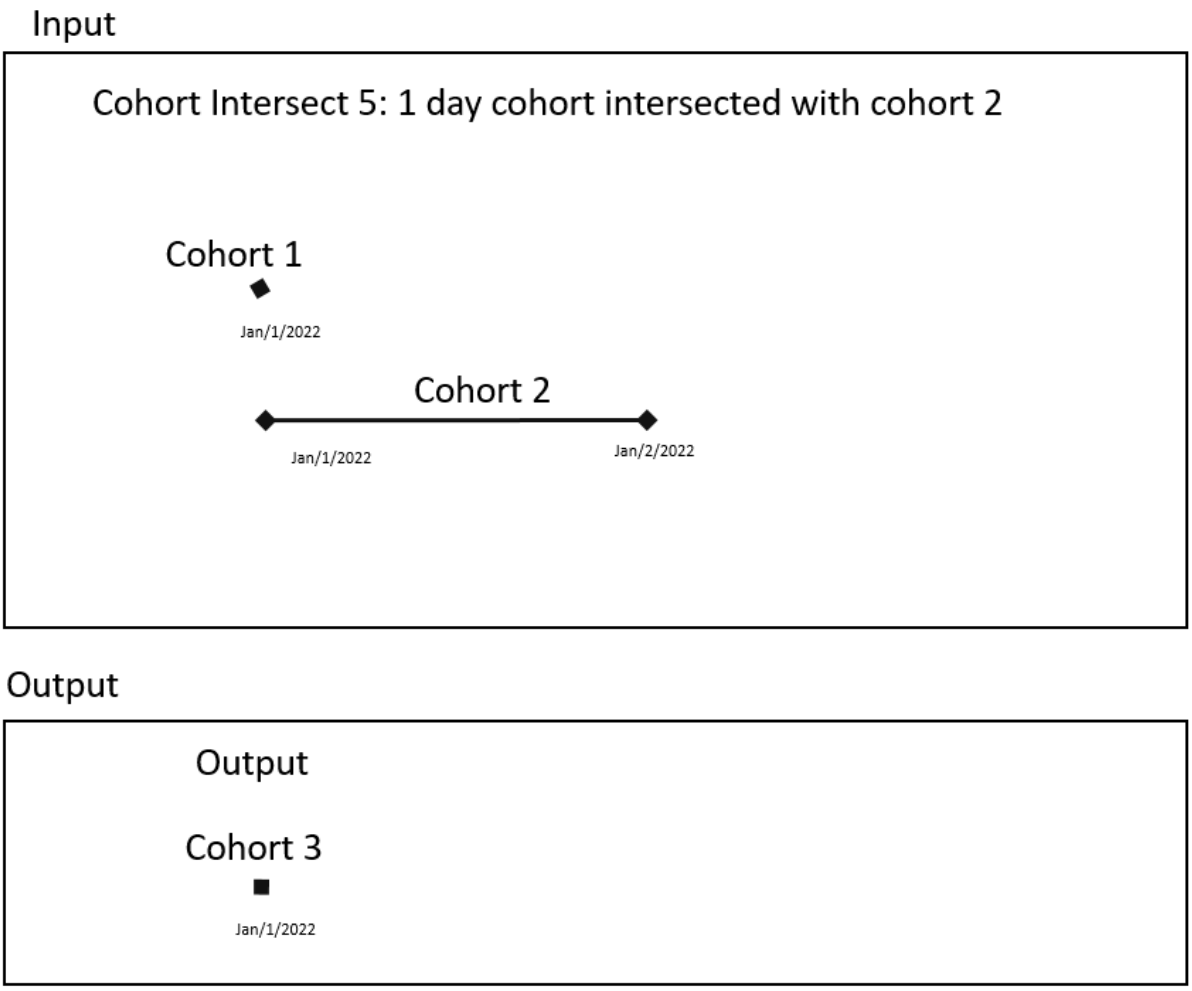


Figure 5: Cohort Intersect 5

Output

```
#> # A tibble: 1 x 4
#>   cohortDefinitionId subjectId cohortStartDate cohortEndDate
#>                <dbl>     <dbl> <date>          <date>
#> 1                  3         1 2022-01-01      2022-01-01
```

1.4 Minus Cohort

Input:

```
cohort <- dplyr::tibble(
  cohortDefinitionId = c(1, 2),
  subjectId = c(1, 1),
  cohortStartDate = c(
    as.Date("2022-01-01"),
    as.Date("2022-02-10")
  ),
  cohortEndDate = c(
    as.Date("2022-03-01"),
    as.Date("2022-05-10")
  )
)
cohort
```

```
#> # A tibble: 2 x 4
#>   cohortDefinitionId subjectId cohortStartDate cohortEndDate
#>                <dbl>     <dbl> <date>          <date>
#> 1                  1         1 2022-01-01      2022-03-01
#> 2                  2         1 2022-02-10      2022-05-10
```

```
CohortAlgebra::minusCohorts(
  connection = connection,
  cohortDatabaseSchema = cohortDatabaseSchema,
  cohortTable = tableName,
  firstCohortId = 1,
  secondCohortId = 2,
  newCohortId = 3
)
```

```
#> Currently in a tryCatch or withCallingHandlers block, so unable to add global calling handlers. Paral
#> Currently in a tryCatch or withCallingHandlers block, so unable to add global calling handlers. Paral
```

Output for example 1

```
#> # A tibble: 1 x 4
#>   cohortDefinitionId subjectId cohortStartDate cohortEndDate
#>                <dbl>     <dbl> <date>          <date>
#> 1                  3         1 2022-01-01      2022-02-09
```

But if the cohorts are switched, i.e. minus cohort 1 from Cohort 2

```
CohortAlgebra::minusCohorts(
  connection = connection,
  cohortDatabaseSchema = cohortDatabaseSchema,
  cohortTable = tableName,
  firstCohortId = 2,
  secondCohortId = 1,
  newCohortId = 4
)
```

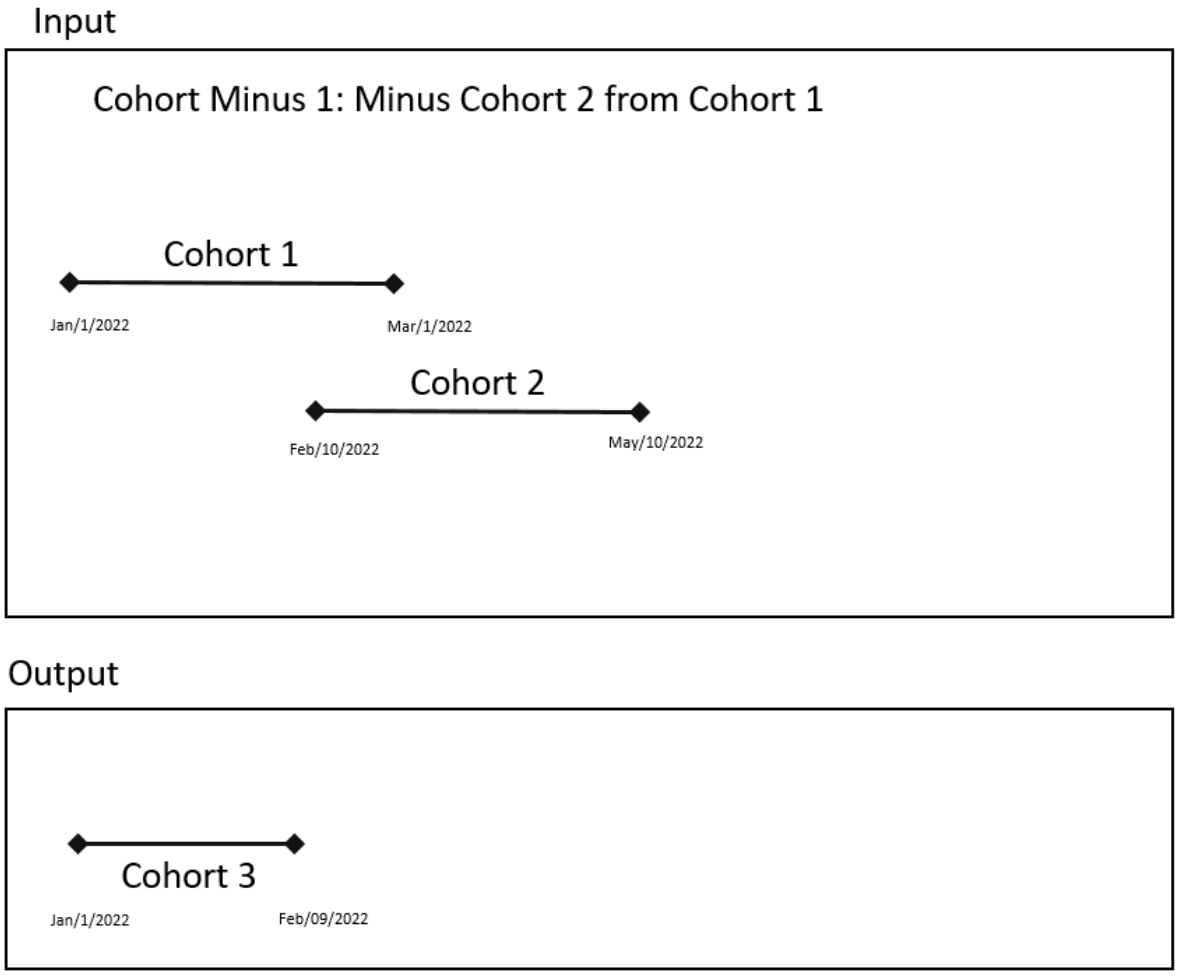


Figure 6: Cohort Minus

```
#> Currently in a tryCatch or withCallingHandlers block, so unable to add global calling handlers. Paral
#> Currently in a tryCatch or withCallingHandlers block, so unable to add global calling handlers. Paral
```

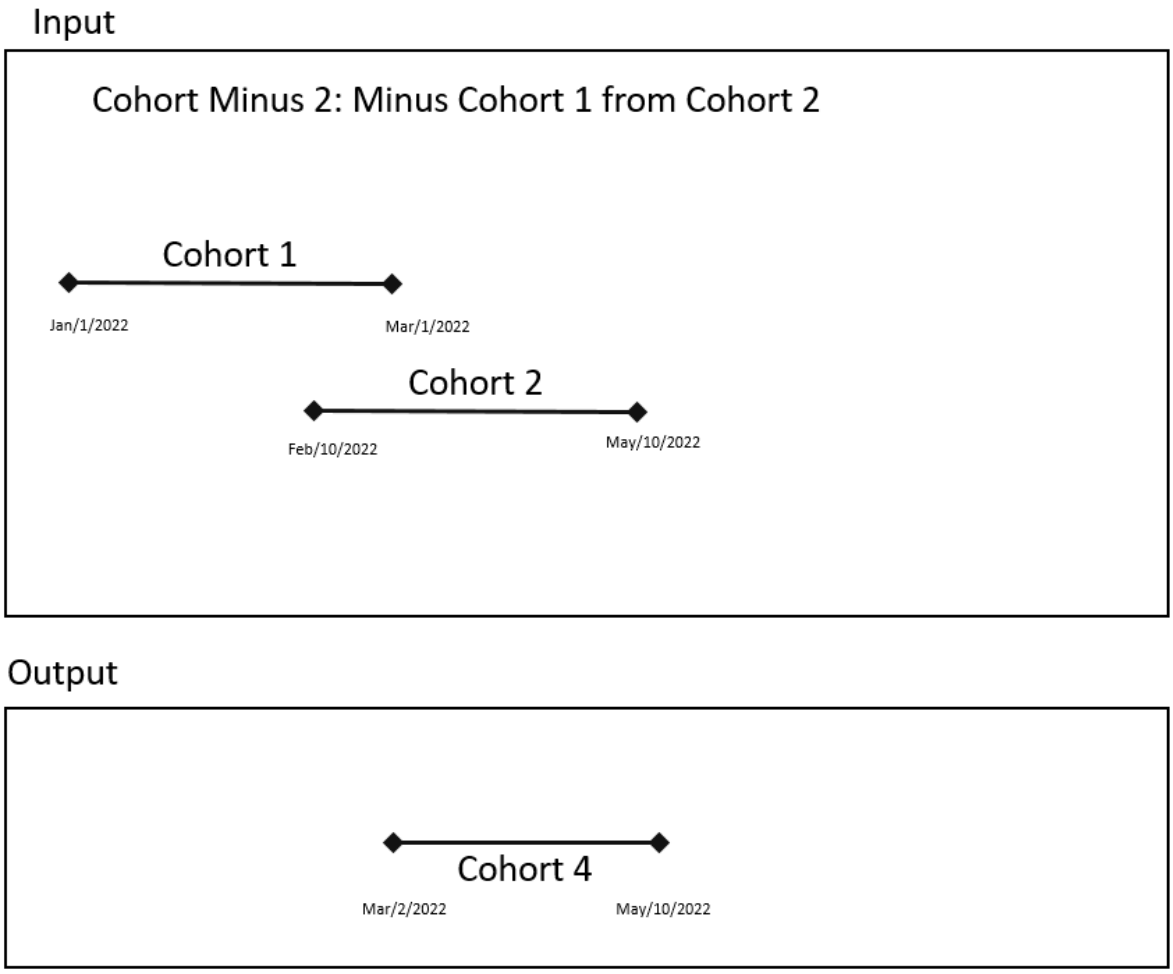


Figure 7: Cohort Minus

Output

```
#> # A tibble: 1 x 4
#>   cohortDefinitionId subjectId cohortStartDate cohortEndDate
#>                <dbl>     <dbl> <date>          <date>
#> 1                  4         1 2022-03-02      2022-05-10
```

Sequence of cohorts are important for minusCohort

1.5 Modify Cohort

Sometimes there is a need to modify a previously instantiated cohorts. Modification may be censoring (left or right) based on calendar date. This may be done using modifyCohort as follows:

```
#> Connecting using PostgreSQL driver
```

1.5.1 Example 1: Modify cohort based on calendar date censoring.

Let us say we have a simple cohort as follows:

```
#> # A tibble: 1 x 4
#>   cohortDefinitionId subjectId cohortStartDate cohortEndDate
#>                <dbl>     <dbl> <date>          <date>
#> 1                  1         1 1999-01-01      1999-01-31
```

We can censor the dates (left, right, or both). Parameter `cohortStartCensorDate` performs left censor, while `cohortEndEndCensorDate` performs right censor. This function is useful if you need to force exit based on calendar dates.

```
CohortAlgebra:::modifyCohort(
  connection = connection,
  cohortDatabaseSchema = cohortDatabaseSchema,
  cohortTable = tableName,
  oldCohortId = 1,
  newCohortId = 2,
  cohortStartCensorDate = as.Date("1999-01-05"),  # for left censor
  cohortEndCensorDate = as.Date("1999-01-25")     # for right censor
)
```

```
#> Currently in a tryCatch or withCallingHandlers block, so unable to add global calling handlers. Para
```

Gives the following output

```
cohortObserved
```

```
#> # A tibble: 1 x 4
#>   cohortDefinitionId subjectId cohortStartDate cohortEndDate
#>                <dbl>     <dbl> <date>          <date>
#> 1                  2         1 1999-01-05      1999-01-25
```

1.5.2 Example 2: Modify cohort by filtering cohort records by date range.

We can also filter an instantiated cohort and create a new cohort by filtering the original cohort by date ranges. This may be applied to cohort_start_date, cohort_end_date or both. Lets take this cohort as an example:

```
#> # A tibble: 2 x 4
#>   cohortDefinitionId subjectId cohortStartDate cohortEndDate
#>                <dbl>     <dbl> <date>          <date>
#> 1                  3         3 2010-01-01      2010-01-05
#> 2                  3         3 1999-01-15      1999-01-25
```

We can filter by cohort_start_date using the parameter cohortStartFilterRange as follows:

```
CohortAlgebra::modifyCohort(
  connection = connection,
  cohortDatabaseSchema = cohortDatabaseSchema,
  cohortTable = tableName,
  oldCohortId = 3,
  newCohortId = 2,
  cohortStartFilterRange = c(as.Date("1998-01-01"), as.Date("1999-12-31")),
  purgeConflicts = TRUE
)
```

```
#> Currently in a tryCatch or withCallingHandlers block, so unable to add global calling handlers. Paral
```

Gives the following output. Note only records where the cohort_start_date was within the given date range became cohortId 2.

```
cohortObserved
```

```
#> # A tibble: 1 x 4
#>   cohortDefinitionId subjectId cohortStartDate cohortEndDate
#>                <dbl>     <dbl> <date>          <date>
#> 1                  2         3 1999-01-15      1999-01-25
```

1.5.3 Example 3: Pad cohorts by adding/substracting days from cohort start or end date.

We can add or substracts days to an instantiated cohort and create a new cohort. Lets take this cohort as an example:

```
#> # A tibble: 1 x 4
#>   cohortDefinitionId subjectId cohortStartDate cohortEndDate
#>                <dbl>     <dbl> <date>          <date>
#> 1                  1         1 1999-01-01      1999-01-31
```

We can pad days as follows:

```
CohortAlgebra::modifyCohort(
  connection = connection,
  cohortDatabaseSchema = cohortDatabaseSchema,
  cdmDatabaseSchema = cdmDatabaseSchema,
  cohortTable = tableName,
  oldCohortId = 1,
  newCohortId = 2,
  purgeConflicts = TRUE,
  cohortStartPadDays = -10,
  cohortEndPadDays = 5
)
```

```
#> Currently in a tryCatch or withCallingHandlers block, so unable to add global calling handlers. Paral
#> Currently in a tryCatch or withCallingHandlers block, so unable to add global calling handlers. Paral
```

Notes: because padding days may extend cohort period outside the observation_period (which is not allowed), this function will require access to the cdmDatabaseSchema and the observation_period table.

Gives the following output.

```
  cohortObserved
```

```
#> # A tibble: 1 x 4
#>   cohortDefinitionId subjectId cohortStartDate cohortEndDate
#>                <dbl>     <dbl> <date>          <date>
#> 1                  2         1 1998-12-22      1999-02-05
```