

# How to Use CohortAlgebra R Package

Gowtham A. Rao

2023-09-10

## Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Installation . . . . .	1
1.2	Cohort UNION . . . . .	2
1.3	InterSect Cohort . . . . .	4
1.4	Minus Cohort . . . . .	13

## 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:

```
if (length(intersect("CohortAlgebra", rownames(installed.packages()))) > 0) {  
  remotes::install_github("OHDSI/CohortAlgebra")  
}
```

```
#> Consider adding 'DATABASECONNECTOR_JAR_FOLDER='D:/windows_temp/AppData/Local/Temp/rtemp\RtmpiW7eIq\j  
#> DatabaseConnector postgresql JDBC driver downloaded to 'D:/windows_temp/AppData/Local/Temp/rtemp\Rtmp
```

## 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. For example: given a cohort table as follows

```
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
#> Inserting data took 0.031 secs
```

The union of the two cohorts is expected to give us

```
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,  
  sourceCohortDatabaseSchema = cohortDatabaseSchema,  
  sourceCohortTable = tableName,  
  targetCohortDatabaseSchema = cohortDatabaseSchema,  
  targetCohortTable = tableName,  
  oldToNewCohortId = oldToNewCohortId  
)
```

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

```
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
```

```
#> # 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
```

```
#> Inserting data took 0.034 secs
```

```
CohortAlgebra::intersectCohorts(
  connection = connection,
  sourceCohortDatabaseSchema = cohortDatabaseSchema,
  sourceCohortTable = tableName,
  targetCohortDatabaseSchema = cohortDatabaseSchema,
  targetCohortTable = tableName,
  cohortIds = c(1, 2),
  newCohortId = 3
)
```

```
#> |
```

```
#> Executing SQL took 0.0147 secs
```

```
#> Intersecting cohorts.
#> Generating eras and saving.
```

Output

## Input



## Output



Figure 1: Cohort Intersect 1

```
#> # 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
```

```
#> # 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
```

```
#> Inserting data took 0.026 secs
```

```
CohortAlgebra::intersectCohorts(
  connection = connection,
  sourceCohortDatabaseSchema = cohortDatabaseSchema,
  sourceCohortTable = tableName,
  targetCohortDatabaseSchema = cohortDatabaseSchema,
  targetCohortTable = tableName,
  cohortIds = c(1, 2),
  newCohortId = 3
)
```

```
#> |
```

```
#> Executing SQL took 0.0153 secs
```

```
#> Intersecting cohorts.
#> Generating eras and saving.
```

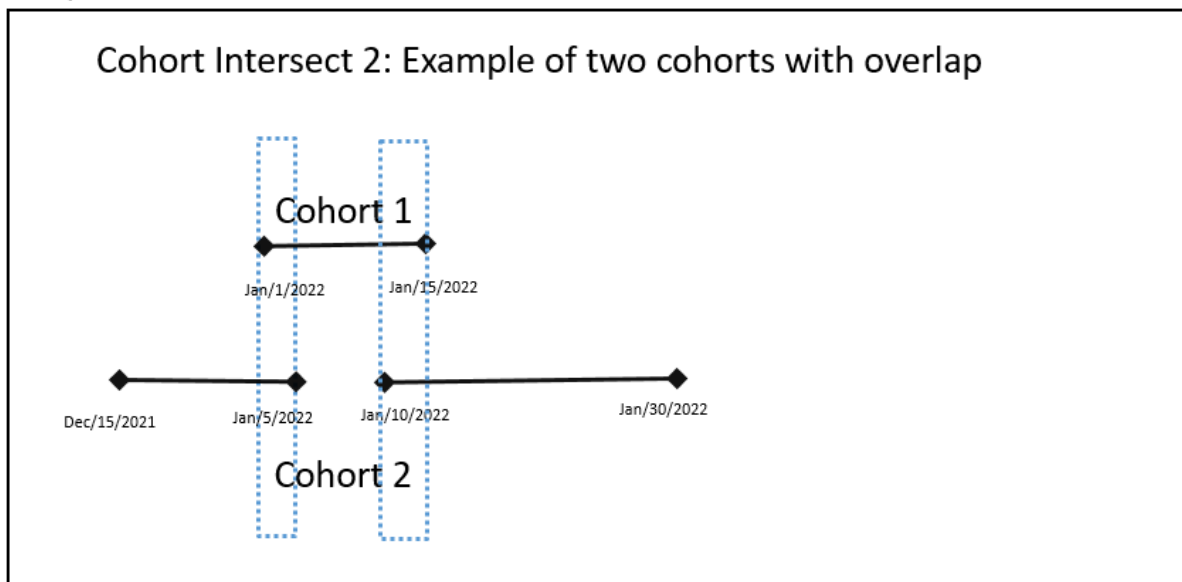
Output

```
#> # 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
```

```
#> # 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
```

```
#> Inserting data took 0.0377 secs
```

```
CohortAlgebra::intersectCohorts(
  connection = connection,
  sourceCohortDatabaseSchema = cohortDatabaseSchema,
  sourceCohortTable = tableName,
  targetCohortDatabaseSchema = cohortDatabaseSchema,
  targetCohortTable = tableName,
  cohortIds = c(1, 2, 3),
  newCohortId = 4
)
```

```
#> |
```

```
#> Executing SQL took 0.0147 secs
```

```
#> Intersecting cohorts.
#> Generating eras and saving.
```

Output

```
data
```

```
#> # A tibble: 0 x 4
#> # i 4 variables: cohortDefinitionId <dbl>, subjectId <dbl>, cohortStartDate <date>, cohortEndDate <date>
```

### 1.3.4 Intersect cohort example 4

Input:

```
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
```

```
#> Inserting data took 0.0249 secs
```



## Input



## Output

NULL Cohort – i.e. no cohort is created

Figure 3: Cohort Intersect 3

```
CohortAlgebra::intersectCohorts(
  connection = connection,
  sourceCohortDatabaseSchema = cohortDatabaseSchema,
  sourceCohortTable = tableName,
  targetCohortDatabaseSchema = cohortDatabaseSchema,
  targetCohortTable = tableName,
  cohortIds = c(1, 2, 3),
  newCohortId = 4
)
```

```
#> |
```

```
#> Executing SQL took 0.0143 secs
```

```
#> Intersecting cohorts.
#> Generating eras and saving.
```

Output

```
data
```

```
#> # A tibble: 0 x 4
#> # i 4 variables: cohortDefinitionId <dbl>, subjectId <dbl>, cohortStartDate <date>, cohortEndDate <date>
```

### 1.3.5 Intersect cohort example 5

Input:

```
cohort
```

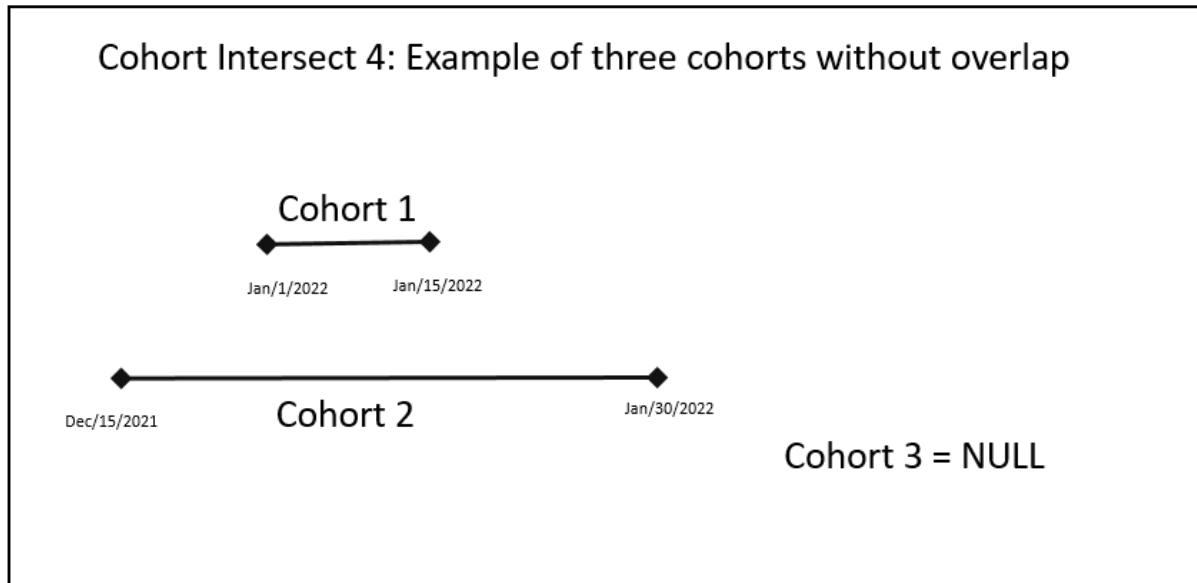
```
#> # A tibble: 2 x 4
#>   cohortDefinitionId subjectId cohortStartDate cohortEndDate
#>   <dbl>         <dbl> <date>           <date>
#> 1             1             1 2022-01-01      2022-01-01
#> 2             2             1 2022-01-01      2022-01-02
```

```
#> Inserting data took 0.0254 secs
```

```
CohortAlgebra::intersectCohorts(
  connection = connection,
  sourceCohortDatabaseSchema = cohortDatabaseSchema,
  sourceCohortTable = tableName,
  targetCohortDatabaseSchema = cohortDatabaseSchema,
  targetCohortTable = tableName,
  cohortIds = c(1, 2),
  newCohortId = 3
)
```

```
#> |
```

## Input

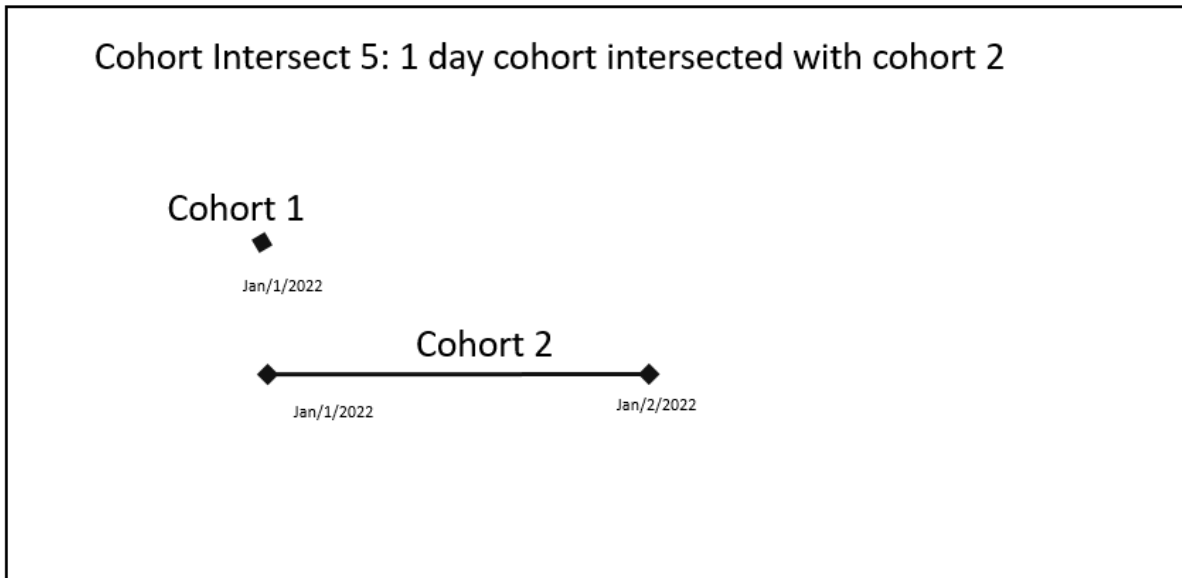


## Output

NULL Cohort – i.e. no cohort is created

Figure 4: Cohort Intersect 4

## Input



## Output



Figure 5: Cohort Intersect 5

```
#> Executing SQL took 0.0144 secs
```

```
#> Intersecting cohorts.  
#> Generating eras and saving.
```

Output

```
data
```

```
#> # 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
```

```
#> # 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
```

```
#> Inserting data took 0.0312 secs
```

```
CohortAlgebra::minusCohorts(  
  connection = connection,  
  sourceCohortDatabaseSchema = cohortDatabaseSchema,  
  sourceCohortTable = tableName,  
  targetCohortDatabaseSchema = cohortDatabaseSchema,  
  targetCohortTable = tableName,  
  firstCohortId = 1,  
  secondCohortId = 2,  
  newCohortId = 3  
)
```

```
#> |
```

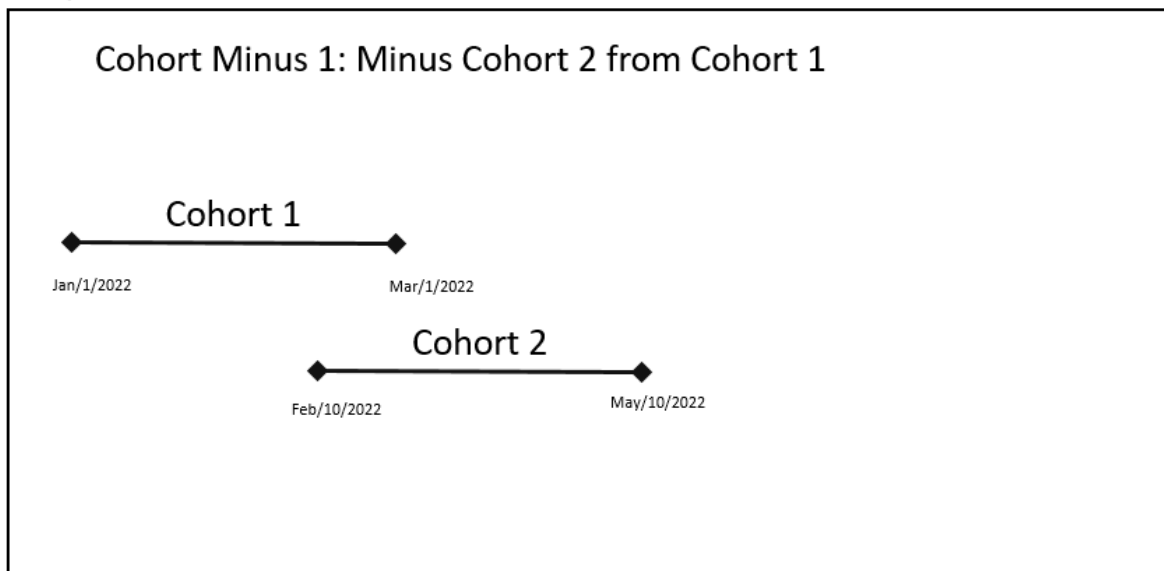
```
#> Executing SQL took 0.0138 secs
```

```
#> |
```

```
#> Executing SQL took 0.015 secs
```

```
#> Performing minus operation.  
#> |
```

## Input



## Output

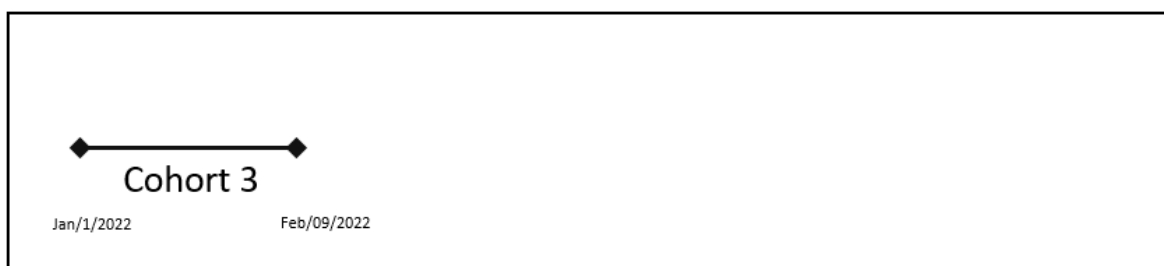


Figure 6: Cohort Minus

```
#> Executing SQL took 0.0144 secs
```

```
#> Intersecting cohorts.  
#> Generating eras and saving.
```

Output for example 1

```
data
```

```
#> # 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,  
  sourceCohortDatabaseSchema = cohortDatabaseSchema,  
  sourceCohortTable = tableName,  
  targetCohortDatabaseSchema = cohortDatabaseSchema,  
  targetCohortTable = tableName,  
  firstCohortId = 2,  
  secondCohortId = 1,  
  newCohortId = 4  
)
```

```
#> |
```

```
#> Executing SQL took 0.0146 secs
```

```
#> |
```

```
#> Executing SQL took 0.0138 secs
```

```
#> Performing minus operation.  
#> |
```

```
#> Executing SQL took 0.0137 secs
```

```
#> Intersecting cohorts.  
#> Generating eras and saving.
```

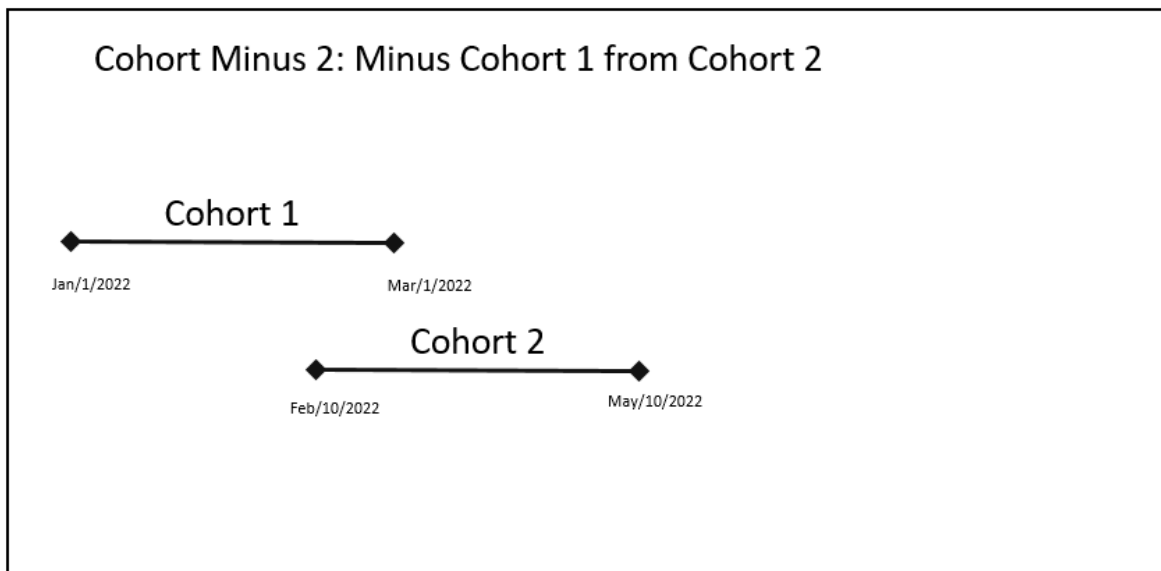
Output

```
data
```

```
#> # 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

## Input



## Output

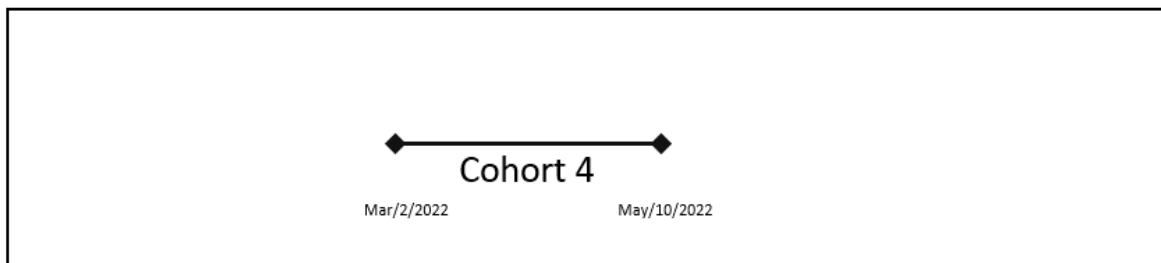


Figure 7: Cohort Minus