

Iniciación práctica al análisis de datos OMOP

# **PatientProfiles**

## **Caracterizar cohortes en OMOP**



Real World Epidemiology

Abril 2024

A la hora de hacer un estudio, una vez tenemos la población definida, ¿cuál es el **primer paso** que debemos hacer?

A la hora de hacer un estudio, una vez tenemos la población definida, ¿cuál es el **primer paso** que debemos hacer?

Hacer una **descriptiva** de nuestra **población de estudio**:  
Características demográficas, condiciones previas, historia previa

❖ Tabla 1

Calcular edad, calcular historia previa, juntar tablas.

Pueden ser **muchas líneas de código...**

- El objetivo del paquete ***PatientProfiles*** es **simplificar el código** necesario para **caracterizar las cohortes** (características demográficas, historia previa, etc.)



- Se ha desarrollado para el **Proyecto de Darwin EU®** por el equipo OxInfer.
- Está disponible en CRAN y se puede instalar fácilmente en R.
- ¿Cómo podemos **descargarlo**?

```
>install.packages("PatientProfiles")  
>library(PatientProfiles)
```

<https://cran.r-project.org/web/packages/PatientProfiles/index.html>

Contiene un **conjunto de funciones** que permiten **añadir las características individuales** más comunes en **cualquier tabla del OMOP CDM** que contiene datos a **nivel de paciente** (por ejemplo, aparición de enfermedades, exposición a fármacos, etc.)

Contiene un conjunto de funciones que permiten añadir las características individuales más comunes en cualquier tabla del OMOP CDM que contiene datos a nivel de paciente (por ejemplo, aparición de enfermedades, exposición a fármacos, etc.)

¿Cuáles son estas **tablas**?



Contiene un conjunto de funciones que permiten añadir las características individuales más comunes en cualquier tabla del OMOP CDM que contiene datos a nivel de paciente (por ejemplo, aparición de enfermedades, exposición a fármacos, etc.)

¿Cuáles son estas **tablas**?

Person, conditions, drug\_exposures...

Contiene un conjunto de funciones que permiten añadir las características individuales más comunes en cualquier tabla del OMOP CDM que contiene datos a nivel de paciente (por ejemplo, aparición de enfermedades, exposición a fármacos, etc.)

¿Cuáles son estas tablas?

Person, conditions, drug\_exposures...

¿Cuáles son las **características** que queremos añadir?

Contiene un conjunto de funciones que permiten añadir las características individuales más comunes en cualquier tabla del OMOP CDM que contiene datos a nivel de paciente (por ejemplo, aparición de enfermedades, exposición a fármacos, etc.)

¿Cuáles son estas tablas?

Person, conditions, drug\_exposures...

¿Cuáles son las **características** que queremos añadir?

Edad, sexo, historia previa, intersecciones entre cohortes...

# Funciones del paquete *PatientProfiles*

addAge	3
addAttributes	4
addCategories	5
addCdmName	6
addCohortIntersect	7
addCohortIntersectCount	9
addCohortIntersectDate	12
addCohortIntersectDays	14
addCohortIntersectFlag	16
addCohortName	18
addConceptIntersect	19
addConceptIntersectCount	21
addConceptIntersectDate	22
addConceptIntersectDays	23
addConceptIntersectFlag	24
addDateOfBirth	25
addDemographics	26
addFutureObservation	28
addInObservation	29
addIntersect	30
addLargeScaleCharacteristics	32
addPriorObservation	32
addSex	34
availableFunctions	35
detectVariables	35
getConceptName	36
getEndName	37
getSourceConceptName	37
getStartName	38
gtCharacteristics	38
gtResult	40
mockPatientProfiles	41
summariseCharacteristics	44
summariseLargeScaleCharacteristics	46
summariseResult	47
suppressCounts	48
variableTypes	48

# Funciones del paquete *PatientProfiles*

addAge	3
addAttributes	4
addCategories	5
addCdmName	6
addCohortIntersect	7
addCohortIntersectCount	9
addCohortIntersectDate	12
addCohortIntersectDays	14
addCohortIntersectFlag	16
addCohortName	18
addConceptIntersect	19
addConceptIntersectCount	21
addConceptIntersectDate	22
addConceptIntersectDays	23
addConceptIntersectFlag	24
addDateOfBirth	25
addDemographics	26
addFutureObservation	28
addInObservation	29
addIntersect	30
addLargeScaleCharacteristics	32
addPriorObservation	32
addSex	34
availableFunctions	35
detectVariables	35
getConceptName	36
getEndName	37
getSourceConceptName	37
getStartName	38
gtCharacteristics	38
gtResult	40
mockPatientProfiles	41
summariseCharacteristics	44
summariseLargeScaleCharacteristics	46
summariseResult	47
suppressCounts	48
variableTypes	48

# addAge()

Si queremos **añadir la edad** a cualquier tabla que contenga la información a nivel del paciente.

```
> cdm$curs_omop_covid <- cdm$curs_omop_covid %>%  
  addAge()
```

Si queremos añadir la edad a cualquier tabla que contenga la información a nivel del paciente.

```
>cdm$curs_omop_covid <- cdm$curs_omop_covid %>%  
  addAge()
```

¿Qué **problemas** pueden surgir a la hora de calcular la edad?

Si queremos añadir la edad a cualquier tabla que contenga la información a nivel del paciente.

```
>cdm$curs_omop_covid <- cdm$curs_omop_covid %>%  
  addAge()
```

¿Qué **problemas** pueden surgir a la hora de calcular la edad?

Puede ser que nos falte información sobre el mes o el día de nacimiento.



# addAge()

```
addAge(  
    x,  
    indexDate = "cohort_start_date",  
    ageName = "age",  
    ageGroup = NULL,  
    ageDefaultMonth = 1,  
    ageDefaultDay = 1,  
    ageImposeMonth = FALSE,  
    ageImposeDay = FALSE,  
    missingAgeGroupValue = "None"  
)
```

# addAge()

```
> cdm$curso_omop_covid
# Source:   table<curso_omop_covid> [?? x 4]
# Database: DuckDB v0.10.0 [apalomar@windows 10 x64:R 4.2.3/C:\Users\
  cohort_definition_id subject_id cohort_start_date cohort_end_date
                <int>         <dbl> <date>          <date>
1                   1           493 2020-10-06      2020-10-16
2                   1          3447 2021-02-21      2021-03-03
3                   1          3596 2021-06-05      2021-06-15
4                   1          4251 2021-04-19      2021-04-29
5                   1          7310 2021-01-20      2021-01-30
6                   1          3598 2020-05-20      2020-05-30
7                   1          8462 2020-07-25      2020-08-04
8                   1          7813 2020-12-09      2020-12-19
9                   1          9848 2021-01-09      2021-01-19
10                  1          2764 2020-08-26      2020-09-05
```

# addAge()

```
> cdm$curso_omop_covid %>% addAge()
# Source:   table<og_008_1712431175> [?? x 5]
# Database: DuckDB v0.10.0 [apalomar@Windows 10 x64:R 4.2.3/C:\Users\apaloma]

  cohort_definition_id subject_id cohort_start_date cohort_end_date age
      <int>          <dbl> <date>          <date>          <dbl>
1             1           493 2020-10-06      2020-10-16         15
2             1          3447 2021-02-21      2021-03-03         13
3             1          3596 2021-06-05      2021-06-15         34
4             1          4251 2021-04-19      2021-04-29         36
5             1          7310 2021-01-20      2021-01-30         89
6             1          3598 2020-05-20      2020-05-30         70
7             1          7813 2020-12-09      2020-12-19          1
8             1          9848 2021-01-09      2021-01-19          7
9             1          2764 2020-08-26      2020-09-05         35
10            1          7269 2021-01-12      2021-01-22         12
```

# addAge()

También se pueden crear **grupos de edades** y darles un nombre

```
> cdm$curso_omop_covid %>% addAge(  
+   ageGroup = list("menores"=c(0,18),  
+                   "adultos"=c(19,70),  
+                   "mayores"=c(71, 150)))
```

# addAge()

También se pueden crear **grupos de edades** y darles un nombre

```
> cdm$curso_omop_covid %>% addAge(  
+   ageGroup = list("menores"=c(0,18),  
+                   "adultos"=c(19,70),  
+                   "mayores"=c(71, 150)))  
# Source:   table<og_010_1712431251> [?? x 6]  
# Database: DuckDB v0.10.0 [apalomar@Windows 10 x64:R 4.2.3/C:\Users\apalomar\AppData
```

	cohort_definition_id	subject_id	cohort_start_date	cohort_end_date	age	age_group
	<int>	<dbl>	<date>	<date>	<dbl>	<chr>
1	1	493	2020-10-06	2020-10-16	15	menores
2	1	3447	2021-02-21	2021-03-03	13	menores
3	1	3596	2021-06-05	2021-06-15	34	adultos
4	1	4251	2021-04-19	2021-04-29	36	adultos
5	1	7310	2021-01-20	2021-01-30	89	mayores
6	1	3598	2020-05-20	2020-05-30	70	adultos
7	1	7813	2020-12-09	2020-12-19	1	menores
8	1	9848	2021-01-09	2021-01-19	7	menores
9	1	2764	2020-08-26	2020-09-05	35	adultos
10	1	7269	2021-01-12	2021-01-22	12	menores

# addSex()

De forma parecida, podemos añadir **información sobre el sexo** de cada individuo.

```
addSex(  
    x,  
    sexName = "sex",  
    missingSexValue = "None"  
)
```

# addSex()

De forma parecida, podemos añadir **información sobre el sexo** de cada individuo.

```
cdm$curs_omop_covid %>% addSex()
Source:   table<dbplyr_114> [?? x 6]
Database: DuckDB v0.9.2 [apistillo@windows 10 x64:R 4.3.2/C:\Users\APISTI~1\AppData
e568870545f68.duckdb]
  cohort_definition_id subject_id cohort_start_date cohort_end_date   age sex
      <int>          <dbl>   <date>          <date>     <dbl> <chr>
1             1           8 2021-01-08      2021-01-18         2 Male
2             1          36 2020-10-09      2020-10-19         4 Male
3             1          47 2020-11-26      2020-12-06        29 Female
4             1          52 2020-12-08      2020-12-18        55 Female
5             1          58 2020-07-12      2020-07-22        53 Female
6             1          72 2020-12-02      2020-12-12        76 Male
7             1          89 2020-11-27      2020-12-07        20 Male
8             1         106 2020-08-16      2020-08-26        90 Male
9             1         137 2020-09-08      2020-09-18         7 Male
10            1         156 2021-01-02      2021-01-12        46 Male
i more rows
```

# addPriorObservation()

Podemos añadir la **historia previa (número de días)** en base al periodo de observación correspondiente. Por ejemplo, podemos añadir la historia previa y después filtrar a los pacientes que tengan al menos un año de historia previa.



# addPriorObservation()

Podemos añadir la **historia previa (número de días)** en base al periodo de observación correspondiente. Por ejemplo, podemos añadir la historia previa y después filtrar a los pacientes que tengan al menos un año de historia previa.

```
addPriorObservation(  
    x,  
    indexDate = "cohort_start_date",  
    priorObservationName = "prior_observation"  
)
```

# addPriorObservation()

Podemos añadir la **historia previa (número de días)** en base al periodo de observación correspondiente. Por ejemplo, podemos añadir la historia previa y después filtrar a los pacientes que tengan al menos un año de historia previa.

```
cdm$curs_omop_covid %>% addPriorObservation()
Source:   table<dbplyr_118> [?? x 7]
Database: DuckDB v0.9.2 [apistillo@windows 10 x64:R 4.3.2/C:\Users\APISTI~1\AppData\Local\Temp\Rtmp
le568870545f68.duckdb]
  cohort_definition_id subject_id cohort_start_date cohort_end_date   age sex   prior_observation
      <int>          <dbl>   <date>          <date>          <dbl> <chr>      <dbl>
1             2           8 2021-01-08      2021-01-18         2 Male        872
2             2          36 2020-10-09      2020-10-19         4 Male       1472
3             2          47 2020-11-26      2020-12-06        29 Female     1678
4             2          52 2020-12-08      2020-12-18        55 Female     2560
5             2          58 2020-07-12      2020-07-22        53 Female     1980
6             2          72 2020-12-02      2020-12-12        76 Male      2652
7             2          89 2020-11-27      2020-12-07        20 Male      2491
8             2         106 2020-08-16      2020-08-26        90 Male      3847
9             2         137 2020-09-08      2020-09-18         7 Male      2584
10            2         156 2021-01-02      2021-01-12        46 Male      2228
```

# addDemographics()

También podemos utilizar las tres funciones que acabamos de ver a la vez.

```
cdm$curs_omop_covid %>%  
  addAge() %>%  
  addSex() %>%  
  addPriorObservation()
```

# addDemographics()

O podemos usar la función *addDemographics()*

```
addDemographics(  
  x,  
  indexDate = "cohort_start_date",  
  age = TRUE,  
  ageName = "age",  
  ageDefaultMonth = 1,  
  ageDefaultDay = 1,  
  ageImposeMonth = FALSE,  
  ageImposeDay = FALSE,  
  ageGroup = NULL,  
  missingAgeGroupValue = "None",  
  sex = TRUE,  
  sexName = "sex",  
  missingSexValue = "None",  
  priorObservation = TRUE,  
  priorObservationName = "prior_observation",  
  futureObservation = TRUE,  
  futureObservationName = "future_observation"  
)
```

Otra característica del paquete es la de poder ver de forma sencilla la **intersección entre dos tablas distintas del CDM**. Por ejemplo, queremos ver cuántas personas de una cohorte específica también tienen alguna otra condición o bien tienen una prescripción de algún medicamento, en una determinada ventana temporal.

Otra característica del paquete es la de poder ver de forma sencilla la intersección con dos tablas distintas del CDM. Por ejemplo, queremos ver cuántas personas de una cohorte específica también tienen alguna otra condición o bien tienen una prescripción de algún medicamento, en una determinada ventana temporal.

**Hay 3 funciones principales**

- addCohortIntersectFlag()
- addCohortIntersectCount()
- addCohortIntersectDays()

# addCohortIntersectFlag()

Esta función crea una nueva columna indicando si el paciente en la cohorte 1 tiene (1) o no tiene (0) un tromboembolismo venoso en cualquier otro momento también.

# addCohortIntersectFlag()

La función tiene más opciones

```
addCohortIntersectFlag(  
    x,  
    targetCohortTable,  
    targetCohortId = NULL,  
    indexDate = "cohort_start_date",  
    censorDate = NULL,  
    targetStartDate = "cohort_start_date",  
    targetEndDate = "cohort_end_date",  
    window = list(c(0, Inf)),  
    nameStyle = "{cohort_name}_{window_name}" )
```

Por ejemplo, se puede definir una, o varias, ventanas temporales.



# addCohortIntersectFlag()

cohort_definition_id	subject_id	cohort_start_date	cohort_end_date	venous_thromboembolism_0_to_inf
<int>	<dbl>	<date>	<date>	<dbl>
1	3251	2021-01-27	2021-02-06	0
1	7937	2020-07-11	2020-07-21	0
1	942	2020-11-26	2020-12-06	1
1	3565	2021-01-27	2021-02-06	1
1	8924	2021-08-29	2021-09-08	1
1	6969	2020-12-23	2021-01-02	0
1	5105	2021-08-26	2021-09-05	1
1	8709	2021-01-06	2021-01-16	1
1	454	2021-03-19	2021-03-29	0
1	5055	2021-01-19	2021-01-29	0

# addCohortIntersectCount ()

También se pueden **contar las intersecciones por persona.**

```
addCohortIntersectCount(  
  x,  
  targetCohortTable,  
  targetCohortId = NULL,  
  indexDate = "cohort_start_date",  
  censorDate = NULL,  
  targetStartDate = "cohort_start_date",  
  targetEndDate = "cohort_end_date",  
  window = list(c(0, Inf)),  
  nameStyle = "{cohort_name}_{window_name}"  
)
```

# addCohortIntersectDays()

Se pueden contar también los **días** que **dura la intersección**.

```
addCohortIntersectDays(  
  x,  
  targetCohortTable,  
  targetCohortId = NULL,  
  indexDate = "cohort_start_date",  
  censorDate = NULL,  
  targetDate = "cohort_start_date",  
  order = "first",  
  window = c(0, Inf),  
  nameStyle = "{cohort_name}_{window_name}"  
)
```

# summariseResults()

Finalmente, se puede obtener un **resumen de las características de los pacientes** en una determinada tabla con la función `summariseResults()`

Se deben especificar las funciones que se aplican a cada grupo de variables.

# summariseResults()

```
summariseResult(  
  table,  
  group = list(),  
  includeOverallGroup = FALSE,  
  strata = list(),  
  includeOverallStrata = TRUE,  
  variables = list(numericVariables =  
    detectVariables(table, "numeric"), dateVariables =  
    detectVariables(table, "date"), binaryVariables =  
    detectVariables(table, "binary"), categoricalVariables =  
    detectVariables(table, "categorical")),  
  estimates = list(numericVariables = c("median", "min",  
    "q25", "q75", "max"), dateVariables = c("median", "min",  
    "q25", "q75", "max"), binaryVariables = c("count",  
    "percentage"), categoricalVariables = c("count",  
    "percentage"))  
)
```

# summariseResults()

```
taula <- summariseResult(taula)
view(taula)
taula
A tibble: 39 x 9
  group_name group_level strata_name strata_level variable variable_level variable_type estimate_type
  <chr>      <chr>      <chr>      <chr>      <chr>      <chr>      <chr>      <chr>
1 Overall   Overall    Overall    Overall    number sub... NA        numeric    count
2 Overall   Overall    Overall    Overall    number rec... NA        NA        count
3 Overall   Overall    Overall    Overall    cohort_sta... NA        date      median
4 Overall   Overall    Overall    Overall    cohort_sta... NA        date      min
5 Overall   Overall    Overall    Overall    cohort_sta... NA        date      q25
6 Overall   Overall    Overall    Overall    cohort_sta... NA        date      q75
7 Overall   Overall    Overall    Overall    cohort_sta... NA        date      max
8 Overall   Overall    Overall    Overall    cohort_end... NA        date      median
9 Overall   Overall    Overall    Overall    cohort_end... NA        date      min
10 Overall  Overall    Overall    Overall    cohort_end... NA        date      q25
i 29 more rows
```

# summariseResults()






```
taula <- summariseResult(cdm$curso_omop_covid,  
  variables = list(numeric=c("age", "prior_observation"),  
    categorical=c("sex", "stroke_0_to_inf", "heart_failure_0_to_inf")),  
  estimates = list(numeric=c("median", "q25", "q75"),  
    categorical= c("count", "percentage")))
```

# summariseResults()

Podemos **estratificar** esta tabla de resultados por las variables que tenemos en la tabla.

group_name	group_level	strata_name	strata_level	variable	variable_level	variable_type	estimate_type	estimate
Overall	Overall	Overall	Overall	antineoplastic_agents_0_to_inf	NA	binary	percentage	0
Overall	Overall	Overall	Overall	stroke_0_to_inf	NA	binary	count	24
Overall	Overall	Overall	Overall	stroke_0_to_inf	NA	binary	percentage	1.244813278008
Overall	Overall	Overall	Overall	heart_failure_0_to_inf	NA	binary	count	34
Overall	Overall	Overall	Overall	heart_failure_0_to_inf	NA	binary	percentage	1.763485477178
Overall	Overall	sex	Female	number subjects	NA	numeric	count	497
Overall	Overall	sex	Female	number records	NA	NA	count	994
Overall	Overall	sex	Female	cohort_start_date	NA	date	median	2020-12-17
Overall	Overall	sex	Female	cohort_start_date	NA	date	min	2020-03-16
Overall	Overall	sex	Female	cohort_start_date	NA	date	q25	2020-10-19
Overall	Overall	sex	Female	cohort_start_date	NA	date	q75	2021-02-01
Overall	Overall	sex	Female	cohort_start_date	NA	date	max	2021-09-14
Overall	Overall	sex	Female	cohort_end_date	NA	date	median	2020-12-27
Overall	Overall	sex	Female	cohort_end_date	NA	date	min	2020-03-26



-  Cohortes
-  1\_PatientProfiles
-  PatientProfiles
-  patientprofiles\_practica
-  patientprofiles\_soluciones

## Objetivo

El objetivo de esta práctica es caracterizar una población a partir de varias tablas de una base de datos en formato OMOP. Veremos cómo añadir de forma sencilla información demográfica, combinar tablas, seleccionar según criterios (flowchart) y hacer una tabla descriptiva.

## ¿Cómo funciona?

En este fichero encontrarás una serie de ejercicios que te proponemos, acompañados de teoría y pistas para su resolución. Puedes crear un script.R en el mismo directorio donde está este fichero, e ir resolviéndolos en el script. En cada ejercicio hay pistas, puesto que el curso es abierto a muchos perfiles y habrá gente que necesitará indicaciones sobre programación en R, otras sobre dónde encontrar las cosas en OMOP, y otras con todo. Además, en este mismo directorio, hay un fichero adicional con las soluciones a los ejercicios, para que puedas autocorregirte la práctica. Si tienes cualquier duda, pregunta a los docentes de apoyo en las prácticas, ¡e intenta utilizar las soluciones solo para corregir!