

Iniciación práctica al análisis de datos OMOP

Estudios de caracterización (PatientProfiles and CohortCharacteristics)



Anna Palomar

apalomar@idiapjgol.org

Marzo 2025

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...**



Iniciación práctica al análisis de datos OMOP

PatientProfiles



- El objetivo del paquete ***PatientProfiles*** es **simplificar el código** necesario para **caracterizar individuos** (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
addAgeQuery	4
addCategories	5
addCdmName	6
addCohortIntersectCount	7
addCohortIntersectDate	8
addCohortIntersectDays	9
addCohortIntersectFlag	11
addCohortName	12
addConceptIntersectCount	13
addConceptIntersectDate	14
addConceptIntersectDays	16
addConceptIntersectField	17
addConceptIntersectFlag	19
addDateOfBirth	20
addDateOfBirthQuery	21
addDeathDate	22
addDeathDays	23
addDeathFlag	24
addDemographics	25
addDemographicsQuery	27
addFutureObservation	29
addFutureObservationQuery	30
addInObservation	31
addInObservationQuery	32
addObservationPeriodId	33
addObservationPeriodIdQuery	34
addPriorObservation	35
addPriorObservationQuery	36
addSex	37
addSexQuery	37
addTableIntersectCount	38
addTableIntersectDate	39
addTableIntersectDays	40
addTableIntersectField	42
addTableIntersectFlag	43
availableEstimates	44
benchmarkPatientProfiles	45
endDateColumn	46
filterCohortId	46

filterInObservation	47
mockDisconnect	47
mockPatientProfiles	48
sourceConceptIdColumn	49
standardConceptIdColumn	49
startDateColumn	50
summariseResult	50
variableTypes	52

addAge()

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

```
cdm$curso_omop_covid <- cdm$curso_omop_covid |>  
  addAge()
```

addAge()

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

```
cdm$curso_omop_covid <- cdm$curso_omop_covid |>  
  addAge()
```

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

addAge()

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

```
cdm$curso_omop_covid <- cdm$curso_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()

Tabla con individuos en el **cdm**

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

Variable en x que contiene la **data** en que computar la edad

Nombre de la nueva columna que contiene edad

Listado de **grupos** de edad a añadir

Mes y día del año a asignar a los individuos que no tienen esta información. Por defecto:1

Si queremos considerar que faltan el mes y día del año de nacimiento para todos los individuos.

Valor para incluir si falta la edad

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$curso_omop_covid <- cdm$curso_omop_covid |>  
  addAge(ageGroup = list("18 años o menos"=c(0,18),  
                          "19 a 70 años"=c(19,70),  
                          "Más de 70 años"=c(71,150)))
```

addAge()

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

	cohort_definition_id	subject_id	cohort_start_date	cohort_end_date	age	age_group
	<int>	<dbl>	<date>	<date>	<int>	<chr>
1	2	8	2021-01-08	2021-01-18	2	18 años o menos
2	2	36	2020-10-09	2020-10-19	4	18 años o menos
3	2	47	2020-11-26	2020-12-06	29	19 a 70 años
4	2	52	2020-12-08	2020-12-18	55	19 a 70 años
5	2	58	2020-07-12	2020-07-22	53	19 a 70 años
6	2	72	2020-12-02	2020-12-12	76	Más de 70 años
7	2	89	2020-11-27	2020-12-07	20	19 a 70 años
8	2	106	2020-08-16	2020-08-26	90	Más de 70 años
9	2	137	2020-09-08	2020-09-18	7	18 años o menos
10	2	156	2021-01-02	2021-01-12	46	19 a 70 años

addSex()

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

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

Tabla con individuos en el **cdm**

Nombre de la nueva columna que contiene el sexo

Valor para incluir si falta el sexo

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 (días o fecha)** 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 (días o fecha)** en base al periodo de observación correspondiente.

```
addPriorObservation(  
  x,  
  indexDate = "cohort_start_date",  
  priorObservationName = "prior_observation",  
  priorObservationType = "days",  
  name = NULL  
)
```

Tabla con individuos en el **cdm**

Variable en x que contiene la **data** en que computar la observación previa

Nombre de la nueva columna a añadir

Indicar si queremos **días o en fecha**

Nombre de la nueva tabla,
si NULL se devuelve una tabla temporal

addPriorObservation()

Podemos añadir la **historia previa (días)** en base al periodo de observación correspondiente.

```
> cdm$curso_omop_covid |> addPriorObservation(priorObservationType = "days")
# Source:   table<og_004_1741339394> [?? x 5]
# Database: DuckDB v1.1.3 [apa1omar@Windows 10 x64:R 4.4.1/C:\Users\apa1omar\AppData\Local\Temp\RtmpSsdhuU\file1e5c18635962.duckdb]
```

	cohort_definition_id	subject_id	cohort_start_date	cohort_end_date	prior_observation
	<int>	<dbl>	<date>	<date>	<int>
1	1	531	2021-09-07	2021-09-17	2895
2	1	6285	2020-11-04	2020-11-14	2509
3	1	8096	2021-01-07	2021-01-17	2274
4	1	8182	2020-11-22	2020-12-02	2334
5	1	1732	2020-07-07	2020-07-17	2393
6	1	1753	2020-09-02	2020-09-12	2163
7	1	2247	2020-12-16	2020-12-26	2530
8	1	7534	2020-07-05	2020-07-15	2455
9	1	9378	2021-07-24	2021-08-03	2608
10	1	209	2021-04-25	2021-05-05	2133

```
# i more rows
```

addPriorObservation()

Podemos añadir la **historia previa (fecha)** en base al periodo de observación correspondiente.

```
> cdm$curso_omop_covid |> addPriorObservation(priorObservationType = "date")
# Source:   table<og_005_1741339529> [?? x 5]
# Database: DuckDB v1.1.3 [apalomar@windows 10 x64:R 4.4.1/C:\Users\apalomar\AppData\Local\Temp\RtmpSsdhuU\file1e5c18635962.duckdb]
```

	cohort_definition_id	subject_id	cohort_start_date	cohort_end_date	prior_observation
	<int>	<dbl>	<date>	<date>	<date>
1	1	531	2021-09-07	2021-09-17	2013-10-04
2	1	6285	2020-11-04	2020-11-14	2013-12-22
3	1	8096	2021-01-07	2021-01-17	2014-10-17
4	1	8182	2020-11-22	2020-12-02	2014-07-03
5	1	1732	2020-07-07	2020-07-17	2013-12-18
6	1	1753	2020-09-02	2020-09-12	2014-10-01
7	1	2247	2020-12-16	2020-12-26	2014-01-12
8	1	7534	2020-07-05	2020-07-15	2013-10-15
9	1	9378	2021-07-24	2021-08-03	2014-06-03
10	1	209	2021-04-25	2021-05-05	2015-06-23

```
# i more rows
```

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",  
  ageMissingMonth = 1,  
  ageMissingDay = 1,  
  ageImposeMonth = FALSE,  
  ageImposeDay = FALSE,  
  ageGroup = NULL,  
  missingAgeGroupValue = "None",  
  sex = TRUE,  
  sexName = "sex",  
  missingSexValue = "None",  
  priorObservation = TRUE,  
  priorObservationName = "prior_observation",  
  priorObservationType = "days",  
  futureObservation = TRUE, ----->  
  futureObservationName = "future_observation",  
  futureObservationType = "days",  
  dateOfBirth = FALSE, ----->  
  dateOfBirthName = "date_of_birth",  
  name = NULL  
)
```

Argumentos para observación **futura**

Si queremos ver la **fecha de nacimiento**

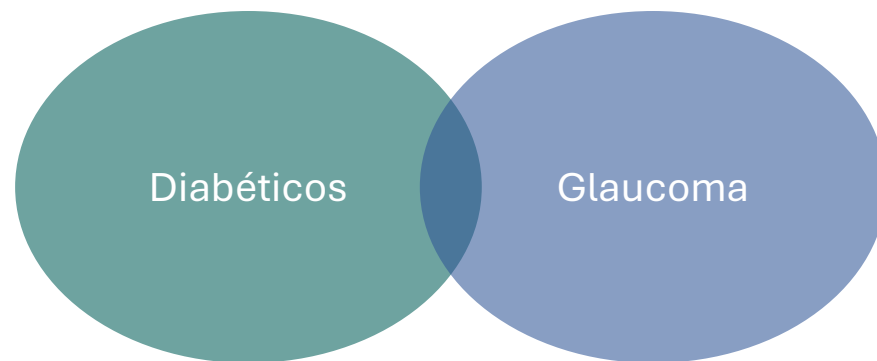
addDemographics()

```
> cdm$curso_omop_covid |> addDemographics(dateOfBirth = TRUE)
# Source:   table<og_008_1741339821> [?? x 9]
# Database: DuckDB v1.1.3 [apalomar@Windows 10 x64:R 4.4.1/C:\Users\apalomar\AppData\Local\Temp\RtmpSsdhuU\file1e5c18635962.duckdb]
  cohort_definition_id subject_id cohort_start_date cohort_end_date age sex prior_observation future_observation date_of_birth
      <int>          <dbl> <date>          <date>          <int> <chr>          <int>          <int> <date>
1             2           8 2021-01-08      2021-01-18         2 Male          872          563 2018-08-20
2             2          36 2020-10-09      2020-10-19         4 Male         1472          705 2016-09-28
3             2          47 2020-11-26      2020-12-06        29 Female        1678          520 1991-02-16
4             2          52 2020-12-08      2020-12-18        55 Female        2560          744 1965-11-11
5             2          58 2020-07-12      2020-07-22        53 Female        1980          953 1967-01-16
6             2          72 2020-12-02      2020-12-12        76 Male         2652          687 1944-05-11
7             2          89 2020-11-27      2020-12-07        20 Male         2491          463 2000-01-01
8             2         106 2020-08-16      2020-08-26        90 Male         3847          857 1930-04-23
9             2         137 2020-09-08      2020-09-18         7 Male         2584          916 2013-03-04
10            2         156 2021-01-02      2021-01-12        46 Male         2228          712 1974-11-21
# i more rows
```

Add intersections

Otra característica del paquete es la de poder ver de forma sencilla la **intersección entre dos cohortes de individuos en el 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.



Add intersections

Hay **3 funciones principales**

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

addCohortIntersectFlag()

Esta función crea una nueva columna indicando si el individuo en la cohorte 1 (ej. diabetes) tiene (1) o no tiene (0) también una condición en la cohorte 2 (ej. glaucoma).

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}", -----  
  name = NULL -----  
)
```

Tabla con individuos en el **cdm**

Nombre tabla con la que queremos ver **intersección**

Id de la cohorte a incluir

Variable en x que contiene la **data** para calcular la intersección

Si queremos censurar eventos en una fecha

Ventana temporal en que considerar intersecciones

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}", -----  
  name = NULL -----  
)
```

Tabla con individuos en el **cdm**

Nombre tabla con la que queremos ver **intersección**

Id de la cohorte a incluir

Variable en x que contiene la **data** para calcular la intersección

Si queremos censurar eventos en una fecha

Ventana temporal en que considerar intersecciones

addCohortIntersectDays()

Calcula el número de días hasta un evento de una cohorte de referencia (target cohort).

```
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}",  
  name = NULL  
)
```

Tabla con individuos en el **cdm**

Nombre tabla con la que queremos ver **intersección**

Id de la cohorte a incluir

Variable en x que contiene la **data** para calcular la intersección

Si queremos censurar eventos en una fecha

Fecha que usar si hay múltiples registros para un mismo individuo durante la ventana temporal . "first" or "last"

Ventana temporal en que considerar intersecciones

Iniciación práctica al análisis de datos OMOP

CohortCharacteristics



- El objetivo del paquete ***CohortCharacteristics*** es resumir y describir características generales de una cohorte de pacientes.
- Calcula **estadísticas descriptivas a nivel de cohorte**, como edad media, distribución de género, comorbilidades, 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("CohortCharacteristics")  
library(CohortCharacteristics)
```

- <https://CRAN.R-project.org/package=CohortCharacteristics>

Funciones CohortCharacteristics

summariseCharacteristics	14
summariseCohortAttrition	17
summariseCohortCount	18
summariseCohortOverlap	18
summariseCohortTiming	19
summariseLargeScaleCharacteristics	20

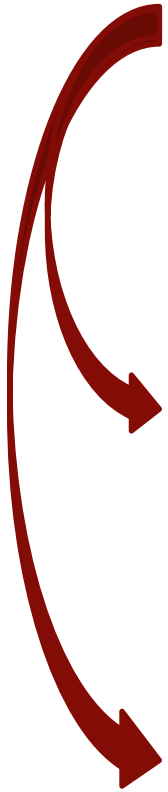
Resumir información

tableCharacteristics	22
tableCohortAttrition	23
tableCohortCount	24
tableCohortOverlap	25
tableCohortTiming	26
tableLargeScaleCharacteristics	27

Crear tablas en base a los datos resumidos

plotCharacteristics	6
plotCohortAttrition	7
plotCohortCount	8
plotCohortOverlap	9
plotCohortTiming	10
plotComparedLargeScaleCharacteristics	12
plotLargeScaleCharacteristics	13

Representar gráficamente estos datos



- Resumir características en una cohorte.

```
summariseCharacteristics(  
  cohort,   
  cohortId = NULL,   
  strata = list(),   
  counts = TRUE,   
  demographics = TRUE,   
  ageGroup = NULL,   
  tableIntersectFlag = list(),   
  tableIntersectCount = list(),   
  tableIntersectDate = list(),   
  tableIntersectDays = list(),   
  cohortIntersectFlag = list(),   
  cohortIntersectCount = list(),   
  cohortIntersectDate = list(),   
  cohortIntersectDays = list(),   
  conceptIntersectFlag = list(),   
  conceptIntersectCount = list(),   
  conceptIntersectDate = list(),   
  conceptIntersectDays = list(),   
  otherVariables = character(),   
  otherVariablesEstimates = c("min", "q25", "median", "q75", "max",  
                               "percentage")  
)
```

Tabla con individuos en el **cdm**

Id de la cohorte

Lista de variables para estratificar los resultados. Tienen que añadirse previamente

Lista de grupos de edad para estratificar los resultados.

Si queremos mostrar intersecciones con alguna tabla, cohorte o concept set

Si quieres resumir otras variables contenidas en la cohorte.

Nombre de las columnas de otras variables

- Formatear un objeto summarised_characteristics en una tabla visual.

```
tableCharacteristics(  
  result, -----> Objeto summarised_characteristics  
  type = "gt", -----> Formatos tabla "gt", "flextable", "tibble"  
  formatEstimateName = c(`N (%)` = "<count> (<percentage>)",  
                        N = "<count>",  
                        `Median [Q25 - Q75]` = "<median> [<q25> - <q75>]",  
                        `Mean (SD)` = "<mean> (<sd>)",  
                        Range = "<min> to <max>"),  
  header = c("group"), -----> Vector con los elementos en orden para encabezado  
  split = c("group", "strata"), -----> Un vector indicando los grupos a dividir  
  groupColumn = NULL,  
  excludeColumns = c("result_id", "estimate_type", "additional_name", "additional_level"),  
  .options = list()  
)
```

- Se puede exportar fácilmente a Word

```
table |> gt::gtsave("table.docx")
```

tableCharacteristics

```
cdm$curso_omop_covid |>
  filter(cohort_definition_id==1) |>
  addSex() |>
  summariseCharacteristics(strata = list("sex")) |> tableCharacteristics(header = "strata")
```

CDM name	Variable name	Variable level	Estimate name	Sex		
				Overall	Female	Male
Covid19 diagnosis						
Synthea	Number records	-	N	964	497	467
	Number subjects	-	N	964	497	467
	Cohort start date	-	Median [Q25 - Q75]	2020-12-13 [2020-10-22 - 2021-01-31]	2020-12-17 [2020-10-19 - 2021-02-01]	2020-12-09 [2020-10-27 - 2021-01-31]
			Range	2020-03-15 to 2021-09-14	2020-03-16 to 2021-09-14	2020-03-15 to 2021-09-08
	Cohort end date	-	Median [Q25 - Q75]	2020-12-23 [2020-11-01 - 2021-02-10]	2020-12-27 [2020-10-29 - 2021-02-11]	2020-12-19 [2020-11-06 - 2021-02-10]
			Range	2020-03-25 to 2021-09-24	2020-03-26 to 2021-09-24	2020-03-25 to 2021-09-18
	Sex	Female	N (%)	497 (51.6%)	497 (100.0%)	-
		Male	N (%)	467 (48.4%)	-	467 (100.0%)

- Crea un ggplot a partir de la salida de summariseCharacteristics.

```
plotCharacteristics(  
  data,  
  x = "variable_name",  
  plotStyle = "barplot",  
  facet = NULL,  
  colour = NULL,  
  colourName = NULL,  
  .options = list()  
)
```

Output de **summariseCharacteristics**

Qué poner en **eje X**. Tiene que ser una columna en los datos

Tipo de **gráfico**. Solo boxplot o barplot

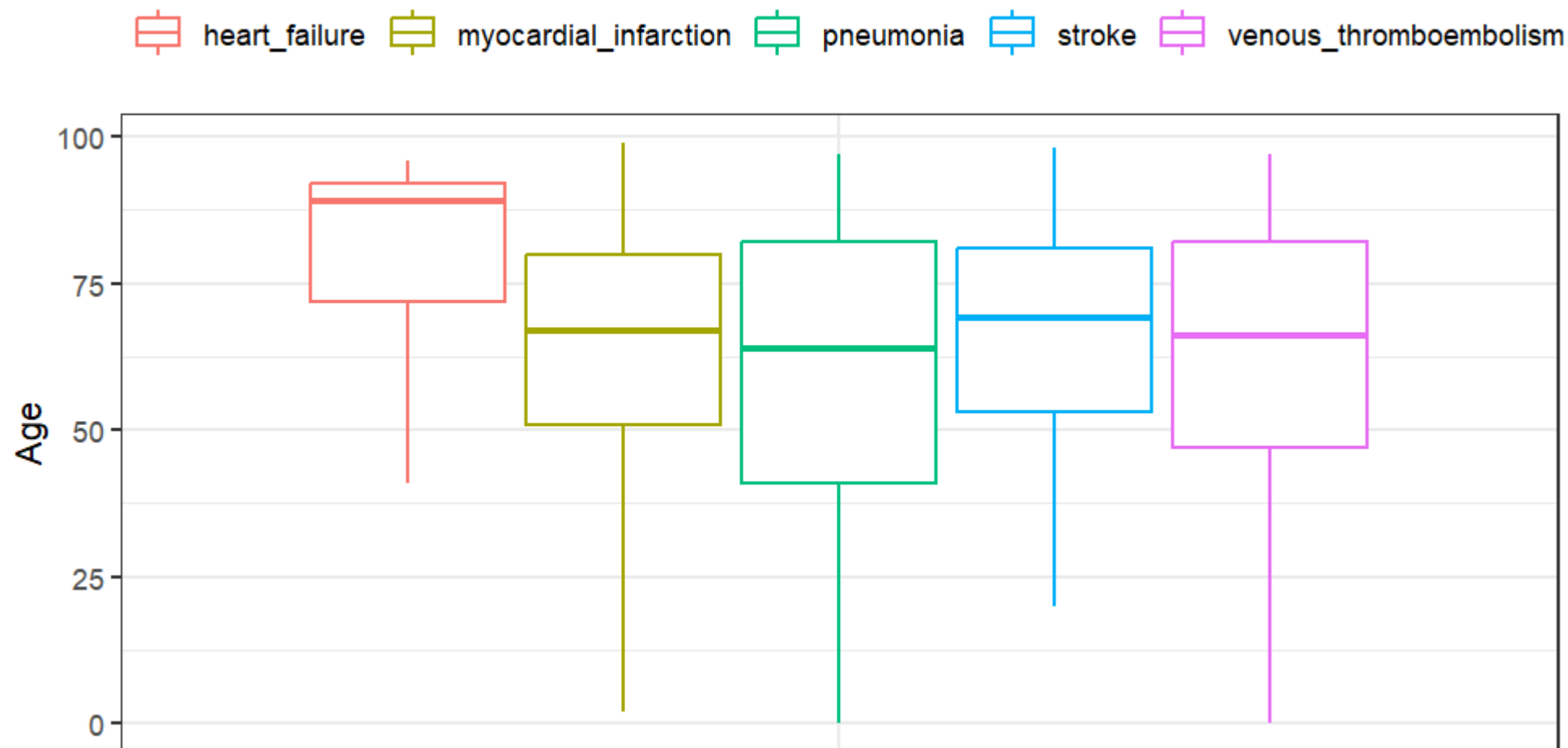
Si queremos **separar** por alguna variable

Si queremos darle **color** por una variable

Nombre para la leyenda de color

Opciones adicionales

plotCharacteristics



```
cdm$condiciones |> summariseCharacteristics() |> filter(variable_name=="Age") |>  
plotCharacteristics(plotStyle = "boxplot", colour = "group_level")
```

summariseLargeScaleCharacteristics

- Esta función se utiliza para resumir las **características a gran escala** de una tabla de cohorte, proporcionando un **resumen descriptivo detallado** de las características **clínicas** y de **tratamiento**.

```
summariseLargeScaleCharacteristics(  
  cohort, -----> Cohorte a caracterizar  
  strata = list(), -----> Lista de estratificación  
  window = list(c(-Inf, -366), c(-365, -31), c(-30, -1), c(0, 0), c(1, 30), c(31, 365),  
                c(366, Inf)), -----> Ventana(s) temporales a caracterizar  
  eventInWindow = NULL, -----> Tabla a caracterizar los eventos (condiciones)  
  episodeInWindow = NULL, -----> Tabla a caracterizar los episodios (fármacos)  
  indexDate = "cohort_start_date",  
  censorDate = NULL,  
  includeSource = FALSE, -----> Si queremos ver los source concept  
  minimumFrequency = 0.005,  
  excludedCodes = c(0)  
)
```


tableLargeScaleCharacteristics

- Formatear un objeto summarised_characteristics en una tabla visual.

```
tableLargeScaleCharacteristics(  
  result,  
  type = "gt",  
  formatEstimateName = c(`N (%)` = "<count> (<percentage>%)"),  
  splitStrata = TRUE,  
  header = c("cdm name", "cohort name", "strata", "window name"),  
  topConcepts = NULL  
)
```

plotLargeScaleCharacteristics

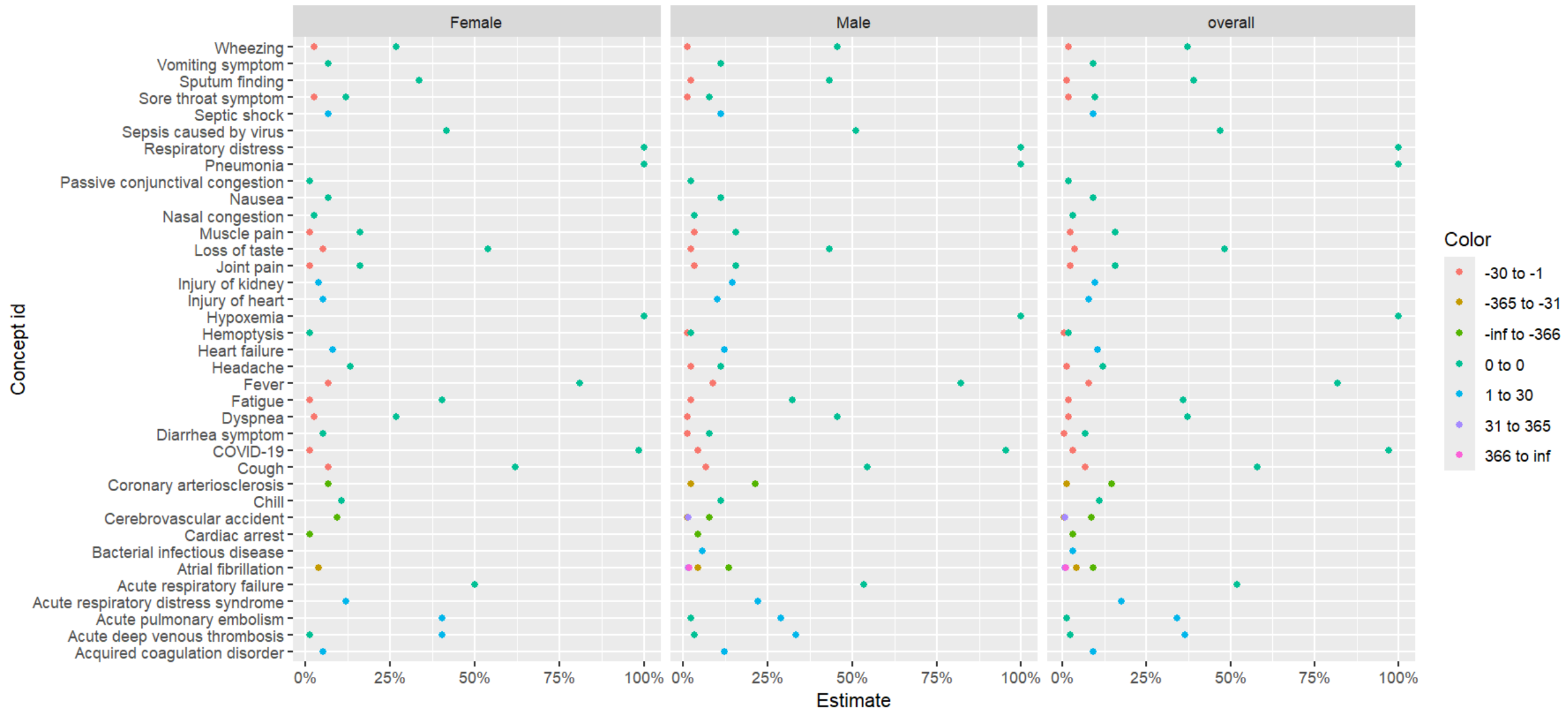
- Crea un ggplot a partir de la salida de summariseLargeScaleCharacteristics.






```
plotLargeScaleCharacteristics(  
  data,  
  position = "horizontal",  
  splitStrata = FALSE,  
  facet = NULL,  
  colorVars = "variable_level"  
)
```

Output de **summariseLargeScaleCharacteristics()**

Horizontal = eje-x es "variable_name" y eje-y es "estimate_value". Vertical al revés

plotLargeScaleCharacteristics



-  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!