

# Package ‘ciecl’

January 27, 2026

**Title** International Classification of Diseases 'ICD-10'/'ICD-11' for Chile

**Version** 0.9.2

**Description** Tools for working with the International Classification of Diseases ('ICD-10' Chile official 'MINSAL'/'DEIS' v2018). Includes optimized 'SQL' search with 'SQLite', fuzzy matching of medical terms (Jaro-Winkler), Charlson and Elixhauser comorbidity calculation, 'WHO' 'ICD-11' 'API' integration, and hierarchical code validation. Data from Centro FIC Chile 'DEIS' <<https://deis.minsal.cl/centrofic/>>.

**License** MIT + file LICENSE

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.2

**Depends** R (>= 4.1.0)

**Imports** DBI, RSQLite, stringdist, stringr, dplyr, tibble, tools, utils, comorbidity, gt

**Suggests** httr2, readxl, usethis, withr, writexl, testthat (>= 3.0.0), knitr, litedown (>= 0.9), rmarkdown

**VignetteBuilder** knitr

**URL** <https://github.com/Rodotasso/ciecl>

**BugReports** <https://github.com/Rodotasso/ciecl/issues>

**Config/testthat/edition** 3

**Language** es

**NeedsCompilation** no

**Author** Rodolfo Tasso Suazo [aut, cre, cph] (ORCID: <<https://orcid.org/0000-0002-1264-4933>>)

**Maintainer** Rodolfo Tasso Suazo <[rtasso@uchile.cl](mailto:rtasso@uchile.cl)>

**Repository** CRAN

**Date/Publication** 2026-01-27 21:20:08 UTC

## Contents

cie10_cl . . . . .	2
cie10_clear_cache . . . . .	3
cie10_sql . . . . .	3
cie11_search . . . . .	4
cie_comorbid . . . . .	5
cie_expand . . . . .	5
cie_guia_busqueda . . . . .	6
cie_lookup . . . . .	6
cie_map_comorbid . . . . .	7
cie_normalizar . . . . .	8
cie_search . . . . .	9
cie_siglas . . . . .	10
cie_table . . . . .	11
cie_validate_vector . . . . .	12

<b>Index</b>	<b>13</b>
--------------	-----------

---

cie10_cl	<i>Dataset CIE-10 Chile oficial MINSAL/DEIS v2018</i>
----------	---

---

## Description

Dataset CIE-10 Chile oficial MINSAL/DEIS v2018

## Usage

cie10\_cl

## Format

tibble con 39,873 filas (categorias y subcategorias):

**codigo** Codigo CIE-10 (ej. "E11.0")

**descripcion** Diagnostico en espanol chileno

**categoria** Categoria jerarquica

**seccion** Seccion dentro del capitulo

**capitulo\_nombre** Nombre descriptivo del capitulo

**inclusion** Terminos incluidos

**exclusion** Terminos excluidos

**capitulo** Capitulo CIE-10 (A-Z)

**es\_daga** Logical, codigo daga (+)

**es\_cruz** Logical, codigo asterisco (\*)

**Source**

<https://deis.minsal.cl/centrofic/>

**Examples**

```
data(cie10_cl)
head(cie10_cl)
```

---

cie10_clear_cache	<i>Limpiar cache SQLite (forzar rebuild)</i>
-------------------	--

---

**Description**

Limpiar cache SQLite (forzar rebuild)

**Usage**

```
cie10_clear_cache()
```

**Value**

No return value, called for side effects (deletes SQLite cache).

**Examples**

```
## Not run:
cie10_clear_cache() # Elimina cie10.db local

## End(Not run)
```

---

cie10_sql	<i>Ejecutar consultas SQL sobre CIE-10 Chile</i>
-----------	--

---

**Description**

Ejecutar consultas SQL sobre CIE-10 Chile

**Usage**

```
cie10_sql(query, close = TRUE)
```

**Arguments**

query	String SQL valido SQLite (SELECT/WHERE/JOIN)
close	Logical, cerrar conexion post-query (default TRUE)

**Value**

tibble resultado query

**Examples**

```
# Buscar diabetes
cie10_sql("SELECT codigo, descripcion FROM cie10 WHERE codigo LIKE 'E11%'")

# Contar por capitulo
cie10_sql("SELECT capitulo, COUNT(*) n FROM cie10 GROUP BY capitulo")

# Join con datos pacientes (externo)
## Not run:
cie10_sql("SELECT p.id, c.descripcion
          FROM pacientes p JOIN cie10 c ON p.codigo = c.codigo")

## End(Not run)
```

---

cie11\_search

*Buscar codigos CIE-11 via API OMS*

---

**Description**

Buscar codigos CIE-11 via API OMS

**Usage**

```
cie11_search(texto, api_key = NULL, lang = "es", max_results = 10)
```

**Arguments**

texto	String termino busqueda espanol/ingles
api_key	String opcional, Client ID + Secret OMS separados ":" Obtener en: <a href="https://icd.who.int/icdapi">https://icd.who.int/icdapi</a>
lang	Character, idioma respuesta ("es" o "en")
max_results	Integer, maximo resultados (default 10)

**Value**

tibble con codigos CIE-11 + titulos o vacio si error

**Examples**

```
## Not run:
# Requiere credenciales OMS gratuitas
Sys.setenv(ICD_API_KEY = "client_id:client_secret")
cie11_search("depresion mayor")

## End(Not run)
```

---

cie_comorbid	<i>Calcular comorbilidades Charlson/Elixhauser para Chile</i>
--------------	---

---

**Description**

Calcular comorbilidades Charlson/Elixhauser para Chile

**Usage**

```
cie_comorbid(data, id, code, map = c("charlson", "elixhauser"), assign0 = TRUE)
```

**Arguments**

data	data.frame con columnas id paciente + codigos CIE-10
id	String nombre columna identificador paciente
code	String nombre columna con codigos CIE-10 (uno por fila)
map	Character, esquema comorbilidad ("charlson" o "elixhauser")
assign0	Logical, asignar 0 si sin comorbilidad (default TRUE)

**Value**

data.frame ancho con scores comorbilidad por paciente

**Examples**

```
## Not run:
df <- data.frame(
  id_pac = c(1, 1, 2, 2),
  diag = c("E11.0", "I21.0", "C50.9", "E10.9")
)
cie_comorbid(df, id = "id_pac", code = "diag", map = "charlson")

## End(Not run)
```

---

cie_expand	<i>Expandir codigo jerarquico (ej. E11 -&gt; E11.0-E11.9)</i>
------------	---

---

**Description**

Expandir codigo jerarquico (ej. E11 -> E11.0-E11.9)

**Usage**

```
cie_expand(codigo)
```

**Arguments**

codigo                      String codigo padre (ej. "E11")

**Value**

Character vector codigos hijos

**Examples**

```
cie_expand("E11")
```

---

cie_guia_busqueda	<i>Guia de funciones de busqueda CIE-10</i>
-------------------	---

---

**Description**

Muestra tabla comparativa de cuando usar cada funcion de busqueda.

**Usage**

```
cie_guia_busqueda()
```

**Value**

data.frame con guia de uso

**Examples**

```
cie_guia_busqueda()
```

---

cie_lookup	<i>Busqueda exacta por codigo CIE-10</i>
------------	--

---

**Description**

Busqueda exacta por codigo CIE-10

**Usage**

```
cie_lookup(  
  codigo,  
  expandir = FALSE,  
  normalizar = TRUE,  
  descripcion_completa = FALSE,  
  extract = FALSE,  
  check_siglas = FALSE  
)
```

**Arguments**

codigo	Character vector de codigos (ej. "E11", "E11.0", c("E11.0", "Z00")) o rango (ej. "E10-E14"). Acepta vectores de multiples codigos. Soporta formatos: con punto (E11.0), sin punto (E110), o solo categoria (E11).
expandir	Logical, expandir jerarquia completa (default FALSE)
normalizar	Logical, normalizar formato de codigos automaticamente (default TRUE)
descripcion_completa	Logical, agregar columna descripcion_completa con formato "CODIGO - DESCRIPCION" (default FALSE)
extract	Logical, extraer codigo CIE-10 de texto con prefijos/sufijos (default FALSE). <b>IMPORTANTE:</b> Solo usar con codigo ESCALAR (longitud 1). Ejemplo: "CIE:E11.0" -> "E11.0", "E11.0-confirmado" -> "E11.0". Para vectores multiples usar extract=FALSE (default).
check_siglas	Logical, buscar siglas medicas comunes (default FALSE). Ejemplo: "IAM" -> I21.0 (Infarto agudo miocardio)

**Value**

tibble con codigo(s) matcheado(s)

**Examples**

```

cie_lookup("E11.0")      # Con punto
cie_lookup("E110")       # Sin punto
cie_lookup("E11")        # Solo categoria
cie_lookup("E11", expandir = TRUE) # Todos E11.x
# Vectorizado - multiples codigos y formatos
cie_lookup(c("E11.0", "Z00", "I10"))
# Con descripcion completa
cie_lookup("E110", descripcion_completa = TRUE)
# Extraer codigo de texto con ruido (solo codigo escalar)
cie_lookup("CIE:E11.0", extract = TRUE)
cie_lookup("E11.0-confirmado", extract = TRUE)
# Nota: Para vectores multiples usar extract=FALSE (default)
# Buscar por siglas medicas
cie_lookup("IAM", check_siglas = TRUE)
cie_lookup("DM2", check_siglas = TRUE)
cie_lookup("EPOC", check_siglas = TRUE)

```

---

cie\_map\_comorbid

---

*Mapeo manual grupos comorbilidad Chile-especifico*


---

**Description**

Agrupar codigos CIE-10 chilenos en categorias comorbilidad MINSAL. Basado en Decreto 1301/2016 MINSAL + icd::icd10\_map\_charlson.

**Usage**

```
cie_map_comorbid(codigos)
```

**Arguments**

`codigos`                      Character vector codigos CIE-10

**Value**

tibble con codigo + categoria\_comorbilidad

**Examples**

```
cie_map_comorbid(c("E11.0", "I50.9", "C50.9"))
```

---

<code>cie_normalizar</code>	<i>Normalizar codigos CIE-10 a formato con punto</i>
-----------------------------	--

---

**Description**

Convierte codigos CIE-10 de diferentes formatos al formato estandar (con punto). Maneja multiples variaciones de entrada comunes en datos clinicos.

**Usage**

```
cie_normalizar(codigos, buscar_db = TRUE)
```

**Arguments**

`codigos`                      Character vector de codigos en cualquier formato  
`buscar_db`                    Logical, buscar codigo en base de datos si no se encuentra exacto (default TRUE)

**Details**

La normalizacion incluye:

- Conversion a mayusculas
- Eliminacion de espacios (inicio, fin e internos)
- Eliminacion de simbolos daga y asterisco (codificacion dual)
- Conversion de guiones a puntos (I10-0 -> I10.0)
- Eliminacion de puntos iniciales (.I10 -> I10)
- Correccion de puntos multiples (E..11 -> E.11)
- Eliminacion de sufijo X en codigos cortos (I10X -> I10)
- Preservacion de X en codigos largos (placeholder 7o caracter)
- Agregado de punto en posicion correcta (E110 -> E11.0)

El sistema de daga/asterisco indica codificacion dual donde la daga marca la enfermedad subyacente y el asterisco la manifestacion. Ambos simbolos se eliminan para normalizacion.



**Value**

Character vector con codigos normalizados al formato con punto

**Examples**

```
cie_normalizar("E110")      # Retorna "E11.0"
cie_normalizar("E11")       # Retorna "E11" (categoria)
cie_normalizar("I10X")      # Retorna "I10" (elimina X)
cie_normalizar("E 11 0")    # Retorna "E11.0" (espacios internos)
cie_normalizar("I10-0")     # Retorna "I10.0" (guion a punto)
cie_normalizar("A17.0\u2020") # Retorna "A17.0" (elimina daga)
cie_normalizar("G01*")      # Retorna "G01" (elimina asterisco)
cie_normalizar(c("E110", "I10X", "Z00")) # Vectorizado
```

---

 cie\_search

---

*Busqueda difusa (fuzzy) de terminos medicos CIE-10*


---

**Description**

Busca en descripciones CIE-10 usando multiples estrategias:

1. Expansion de siglas medicas (IAM, TBC, DM, etc.)
2. Busqueda exacta por subcadena (mas rapida)
3. Busqueda fuzzy con Jaro-Winkler (tolera typos)

**Usage**

```
cie_search(
  texto,
  threshold = 0.7,
  max_results = 50,
  campo = c("descripcion", "inclusion"),
  solo_fuzzy = FALSE,
  verbose = TRUE
)
```

**Arguments**

texto	String termino medico en espanol o sigla (ej. "diabetes", "IAM", "TBC")
threshold	Numeric entre 0 y 1, umbral similitud Jaro-Winkler (default 0.70)
max_results	Integer, maximo resultados a retornar (default 50)
campo	Character, campo busqueda ("descripcion" o "inclusion")
solo_fuzzy	Logical, usar solo busqueda fuzzy sin busqueda exacta (default FALSE)
verbose	Logical, mostrar mensajes informativos (default TRUE). Usar FALSE en scripts.

**Details**

La búsqueda es tolerante a tildes: "neumonia" encuentra "neumonia". Soporta siglas medicas comunes: "IAM" busca "infarto agudo miocardio".

**Value**

tibble ordenado por score descendente (1.0 = coincidencia exacta). Incluye atributo "sigla\_expandida" si se uso una sigla.

**Examples**

```
# Búsqueda basica
cie_search("diabetes")

cie_search("neumonia")

# Búsqueda por siglas medicas
cie_search("IAM")
cie_search("DM2")

# Tolerante a tildes y typos
cie_search("diabetis")

# Buscar en inclusiones
cie_search("bacteriana", campo = "inclusion")
```

---

cie\_siglas

*Listar siglas medicas soportadas*

---

**Description**

Muestra todas las siglas medicas que pueden usarse en `cie_search()`.

**Usage**

```
cie_siglas(categoria = NULL)
```

**Arguments**

**categoria** Character opcional, filtrar por categoria. Valores validos: "cardiovascular", "respiratoria", "metabolica", "gastrointestinal", "infecciosa", "oncologica", "reumatologica", "neurologica", "psiquiatrica", "traumatologica", "pediatrica", "gineco\_obstetrica". Si es NULL (default), retorna todas las siglas.

**Value**

tibble con columnas: sigla, termino\_búsqueda, categoria

**Examples**

```
# Ver todas las siglas
cie_siglas()

# Filtrar por categoria
cie_siglas("cardiovascular")
cie_siglas("oncologica")

# Buscar una sigla especifica
cie_siglas() |> dplyr::filter(sigla == "iam")
```

---

**cie\_table***Generar tabla HTML interactiva GT de codigo CIE-10*

---

**Description**

Generar tabla HTML interactiva GT de codigo CIE-10

**Usage**

```
cie_table(codigo, interactive = TRUE)
```

**Arguments**

codigo	String codigo (ej. "E11" muestra jerarquia)
interactive	Logical, tabla gt reactable (default TRUE)

**Value**

Objeto gt table HTML

**Examples**

```
## Not run:
cie_table("E11") # Diabetes tipo 2 completo

## End(Not run)
```

---

cie_validate_vector	<i>Validar vector de codigos CIE-10 formato</i>
---------------------	---

---

**Description**

Validar vector de codigos CIE-10 formato

**Usage**

```
cie_validate_vector(codigos, strict = FALSE)
```

**Arguments**

codigos	Character vector codigos (ej. c("E11.0", "Z00.0"))
strict	Logical, validar existencia en DB (default FALSE)

**Value**

Logical vector misma longitud input

**Examples**

```
cie_validate_vector(c("E11.0", "INVALIDO", "Z00"))
```

# Index

## \* **datasets**

- [cie10\\_cl, 2](#)
- [cie10\\_cl, 2](#)
- [cie10\\_clear\\_cache, 3](#)
- [cie10\\_sql, 3](#)
- [cie11\\_search, 4](#)
- [cie\\_comorbid, 5](#)
- [cie\\_expand, 5](#)
- [cie\\_guia\\_busqueda, 6](#)
- [cie\\_lookup, 6](#)
- [cie\\_map\\_comorbid, 7](#)
- [cie\\_normalizar, 8](#)
- [cie\\_search, 9](#)
- [cie\\_siglas, 10](#)
- [cie\\_table, 11](#)
- [cie\\_validate\\_vector, 12](#)