Package 'covidmx'

February 18, 2023

Title Descarga y analiza datos de COVID-19 en México

Version 0.7.7

Description Herramientas para el análisis de datos de COVID-19 en México. Descarga y analiza los datos para COVID-19 de la Direccion General de Epidemiología de México (DGE) <https://www.gob.mx/salud/documentos/datos-abiertos-152127>, la Red de Infecciones Respiratorias Agudas Graves (Red IRAG) <https://www.gits.igg.unam.mx/red-irag-dashboard/reviewHome> y la Iniciativa Global para compartir todos los datos de influenza (GISAID) <https://gisaid.org/>.
English: Downloads and analyzes data of COVID-19 from the Mexican General Directorate of Epidemiology (DGE), the Network of Severe Acute Respiratory Infections (IRAG network),and the Global Initiative on Sharing All Influenza Data GISAID.

License MIT + file LICENSE

Encoding UTF-8

Language es

RoxygenNote 7.2.3

Imports cli, DBI, dplyr, duckdb (>= 0.4.0), fs, pins (>= 1.0.1), RCurl, readr, readxl, rlang, stats, stringr, tibble, tidyr, tools

Suggests covr, cowplot, crayon, dbplyr, EpiEstim, ggformula, ggplot2, ggstream, ggtext (>= 0.1.0), glue, lubridate, MetBrewer, remotes, rmarkdown, rstudioapi, scales, sessioninfo, testthat (>= 3.0.0), usethis

URL https://github.com/RodrigoZepeda/covidmx, https://rodrigozepeda.github.io/covidmx/

BugReports https://github.com/RodrigoZepeda/covidmx/issues

Depends R (>= 4.2.0)

LazyData true

Config/testthat/edition 3

Config/testthat/parallel true

NeedsCompilation no

Author Rodrigo Zepeda-Tello [aut, cre]
(<https://orcid.org/0000-0003-4471-5270>),
Mauricio Hernandez-Avila [aut],
Alberto Almuiña [ctb] (Author of included zzz fragment),
Jonah Gabry [ctb] (Author of included cmdstanr fragment),
Rok Češnovar [ctb] (Author of included cmdstanr fragment),
Ben Bales [ctb] (Author of included cmdstanr fragment),
Mitzi Morris [ctb] (Author of included cmdstanr fragment),
Mikhail Popov [ctb] (Author of included cmdstanr fragment),
Mike Lawrence [ctb] (Author of included cmdstanr fragment),
William Michael Landau [ctb] (Author of included cmdstanr fragment),
Jacob Socolar [ctb] (Author of included cmdstanr fragment),
Andrew Johnson [ctb] (Author of included cmdstanr fragment),
Instituto Mexicano del Seguro Social [cph, fnd]

Maintainer Rodrigo Zepeda-Tello <rzepeda17@gmail.com>

Repository CRAN

Date/Publication 2023-02-17 23:40:02 UTC

R topics documented:

`casos` *Casos de COVID-19 en Mexico*

Description

`casos` Calcula el numero de casos registrados por fecha agrupando (o sin hacerlo) por diferentes covariables. Por default calcula el total de casos (con y sin prueba positiva)

Usage

```
casos(
  datos_covid,
 entidades = c("AGUASCALIENTES", "BAJA CALIFORNIA", "BAJA CALIFORNIA SUR", "CAMPECHE",
   "CHIAPAS", "CHIHUAHUA", "CIUDAD DE MÉXICO", "COAHUILA DE ZARAGOZA", "COLIMA",
   "DURANGO", "GUANAJUATO", "GUERRERO", "HIDALGO", "JALISCO", "MÉXICO",
  "MICHOACÁN DE OCAMPO", "MORELOS", "NAYARIT", "NUEVO LEÓN", "OAXACA", "PUEBLA",
  "QUERÉTARO", "QUINTANA ROO", "SAN LUIS POTOSÍ", "SINALOA", "SONORA", "TABASCO",
  "TAMAULIPAS", "TLAXCALA", "VERACRUZ DE IGNACIO DE LA LLAVE", "YUCATÁN", "ZACATECAS"),
  group_by_entidad = TRUE,
  entidad_tipo = c("Unidad Medica", "Residencia", "Nacimiento"),
  fecha_tipo = c("Sintomas", "Ingreso", "Defuncion"),
  tipo_clasificacion = c("Sospechosos", "Confirmados COVID", "Negativo a COVID",
    "Inválido", "No realizado"),
  group_by_tipo_clasificacion = FALSE,
  tipo_paciente = c("AMBULATORIO", "HOSPITALIZADO", "NO ESPECIFICADO"),
  group_by_tipo_paciente = FALSE,
  tipo_uci = c("SI", "NO", "NO APLICA", "SE IGNORA", "NO ESPECIFICADO"),
  group_by_tipo_uci = FALSE,
 tipo_sector = c("CRUZ ROJA", "DIF", "ESTATAL", "IMSS", "IMSS-BIENESTAR", "ISSSTE",
   "MUNICIPAL", "PEMEX", "PRIVADA", "SEDENA", "SEMAR", "SSA", "UNIVERSITARIO",
   "NO ESPECIFICADO"),
  group_by_tipo_sector = FALSE,
  defunciones = FALSE,
  edad_cut = NULL,
  as_tibble = TRUE,
  fill_zeros = as_tibble,
  list_name = "casos",
  .grouping_vars = c()
)
```

Arguments

datos_covid	(**obligatorio**) Lista de tibbles o duckdbs resultante de descarga_datos_abiertos() o read_datos_abiertos()
entidades	(**opcional**) Vector con las entidades de las unidades medicas a analizar. Opciones: AGUASCALIENTES, BAJA CALIFORNIA, BAJA CALIFORNIA SUR, CAMPECHE, CHIAPAS, CHIHUAHUA, CIUDAD DE MEXICO, COAHUILA DE ZARAGOZA , COLIMA, DURANGO, GUANAJUATO, GUERRERO, HIDALGO, JALISCO, MEXICO, MICHOACAN DE OCAMPO, MORELOS,NAYARIT NUEVO LEON, OAXACA ,PUEBLA, QUERETARO,QUINTANA ROO, SAN LUIS POTOSI, SINALOA, SONORA, TABASCO, TAMAULIPAS,TLAXCALA, VERACRUZ DE IGNACIO DE LA LL YUCATAN, ZACATECAS.
group_by_entidad	(**opcional**) TRUE obtiene los casos para cada entidad reportando en cada fecha la entidad y los casos en dicha entidad. FALSE junta las entidades sumando sus casos en una sola observacion por cada fecha.
entidad_tipo	(**opcional**) Indica a que se refiere las entidades seleccionadas. Elige una de las

opciones: Unidad Medica (entidad de la unidad medica), Nacimiento (entidad de origen del individuo) o Residencia (entidad donde reside el individuo).

fecha_tipo (**opcional**) Selecciona si la fecha que se utiliza es la fecha de Ingreso (si aplica), la fecha de Sintomas o la de Defuncion (si aplica). El default es fecha de Sintomas.

tipo_clasificacion
(**opcional**) Vector con el tipo de clasificaciones (por la prueba) a incluir:Sospechosos,Confirmados COVI Negativo a COVID, Inv\u00e1lido, No realizado

group_by_tipo_clasificacion
(**opcional**) Booleana determinando si regresa la base con cada entrada agrupada por tipo_clasificacion (es decir cada fecha se generan tantos observaciones como grupos de tipo de clasificación) en caso TRUE. Si FALSE suma todos los casos del tipo de clasificacion por fecha dando un solo numero por fecha. El defalt es FALSE.

tipo_paciente (**opcional**) Vector con el tipo de pacientes a incluir. Opciones: AMBULATORIO, HOSPITALIZADO, NO ESPECIFICADO. Por default se incluyen todos.

group_by_tipo_paciente
(**opcional**) Booleana determinando (caso TRUE) si regresa la base con cada entrada agrupada por tipo_paciente (es decir cada fecha se genera un renglon para AMBULATORIO, un renglon para HOSPITALIZADO, etc) o bien si se suman todos los grupos y cada fecha reporta solo la suma de estos (estilo AMBULATORIO + HOSPITALIZADO segun las categorias de tipo_paciente) El default es FALSE.

tipo_uci (**opcional**) Vector con el tipo de valores para Unidad de Cuidado Intensivo (UCI) a incluir: SI,NO,NO APLICA,SE IGNORA,NO ESPECIFICADO. Por default se incluyen todos.

group_by_tipo_uci
(**opcional**) Booleana. El caso TRUE determina si regresa la base con cada fecha teniendo diferentes renglones uno para cada tipo_uci (es decir cada fecha se generan tantos observaciones como grupos de tipo de UCI) o bien en una sola fecha se suman todos los tipos de UCI (FALSE). El default es FALSE.

tipo_sector (**opcional**) Vector con los sectores del sistema de salud a incluir: CRUZ ROJA,DIF,ESTATAL,IMSS,IMSS-BIE MUNICIPAL,PEMEX, PRIVADA,SEDENA,SEMAR,SSA, UNIVERSITARIO,NO ESPECIFICADO.
Por default se incluyen todos.

group_by_tipo_sector
(**opcional**) Booleana determina en el caso de TRUE si regresa la base con cada entrada agrupada por tipo_sector (es decir cada fecha tiene una entrada con los del IMSS, una entrada distinta con los de ISSSTE, etc) o bien en caso de FALSE se devuelve una sola entrada por fecha con la suma IMSS + ISSSTE + etc segun los sectores seleccionados. El default es FALSE.

defunciones (**opcional**) Booleana si incluir sólo defunciones TRUE o a todos FALSE. El default es FALSE.

edad_cut (**opcional**) Vector con secuencia de edades para hacer grupos. Por ejemplo edad_cut = c(0, 10, Inf) arma dos grupos de edad de 0 a 10 y de 10 a infinito o bien edad_cut = c(15, 20) deja sólo los registros entre 15 y 20 años. Por default es NULL y no arma grupos etarios.

as_tibble	(**opcional**) Regresar como tibble el resultado. En caso de que as_tibble sea FALSE se devuelve como conexion en duckdb. Se recomienda el default (tibble).
fill_zeros	(**opcional**) En caso de que el resultado sea un tibble regresa observaciones para todas las combinaciones de variables incluyendo como 0 aquellas fechas cuando no se observaron casos. En caso contrario no se incluyen las filas donde no se observaron casos.
list_name	(**opcional**) Asigna un nombre en la lista de datos a la base generada
.grouping_vars	(**opcional**) Vector de variables adicionales de agrupacion de los conteos. Por ejemplo si se agrega .grouping_vars = 'DIABETES' entonces para cada fecha habra dos conteos de casos uno de los que tienen diabetes y uno de los que no.

Details

La función es un grupo de funciones de dplyr optimizadas para velocidad. Por ejemplo calcular los casos por entidad se hace lo siguiente

```
datos_covid |> casos()
```

es lo mismo que:

```
library(dplyr)
datos_covid$casos <- datos_covid$dats |>
    group_by(ENTIDAD_UM, FECHA_SINTOMAS) |>
    tally() |>
   left_join(datos_covid$dict$ENTIDAD_UM, by = c("ENTIDAD_UM" = "CLAVE_ENTIDAD"))
```

Elaboraciones mas complicadas en casos tienen su equivalente en dplyr por ejemplo:

```
datos_covid <- datos_covid |>
  casos(
    entidad_tipo = "Residencia",
    entidades = c("BAJA CALIFORNIA", "BAJA CALIFORNIA SUR"),
    group_by_tipo_clasificacion = FALSE,
    tipo_paciente = c("AMBULATORIO", "HOSPITALIZADO"),
    group_by_tipo_paciente = TRUE,
    list_name = "bajas"
  )
```

es equivalente a

```
datos_covid$bajas <- datos_covid$dats |>
   filter(ENTIDAD_RES == "02" | ENTIDAD_RES == "03") |> #BC/BCS
   filter(TIPO_PACIENTE == 1 | TIPO_PACIENTE == 2) |> #Ambulatorio/Hospitalizado
   group_by(FECHA_SINTOMAS, ENTIDAD_RES, TIPO_PACIENTE) |>
   tally() |>
  left_join(datos_covid$dict$ENTIDAD_RES, by = c("ENTIDAD_RES" = "CLAVE_ENTIDAD")) |>
   left_join(datos_covid$dict$PACIENTE, by = c("TIPO_PACIENTE" = "CLAVE"))
```

Value

Une a la lista de `datos_covid` una nueva entrada de nombre `list_name` (default: `casos`) con una base de datos (`tibble` o `dbConnection`) con los resultados agregados.

- casos - Base de datos generara con los datos agregados (el nombre cambia si se usa `list_name`).
- dict - Diccionario de datos
- dats - Datos originales (conexion a duckdb o `tibble`)
- disconnect - Función para desconectarte de duckdb
- ... - Cualquier otro elemento que ya existiera en `datos_covid`

See Also

`descarga_datos_abiertos()` `numero_pruebas()` `cfr()` `chr()` `estima_rt()` `positividad()`

Examples

```
# Para el ejemplo usaremos los datos precargados (datosabiertos) pero tu puedes
# correr el ejemplo descargando informacion mas reciente:
datos_covid <- datosabiertos

# Casos por entidad
datos_covid <- datos_covid |> casos()
head(datos_covid$casos)

# Defunciones por entidad
datos_covid <- datos_covid |> casos(defunciones = TRUE, list_name = "defunciones")
head(datos_covid$defunciones)

# Hospitalizados por entidad
datos_covid <- datos_covid |>
  casos(tipo_paciente = "HOSPITALIZADO", list_name = "hospitalizados")
head(datos_covid$hospitalizados)

# UCI por entidad

datos_covid <- datos_covid |> casos(tipo_uci = "SI", list_name = "uci")
head(datos_covid$uci)

# Solo pacientes IMSS
datos_covid <- datos_covid |> casos(tipo_sector = "IMSS", list_name = "imss")
head(datos_covid$imss)

# Pacientes IMSS y PEMEX separados
datos_covid <- datos_covid |> casos(tipo_sector = c("IMSS", "PEMEX"), list_name = "imss_y_pemex")
head(datos_covid$imss_y_pemex)

# Pacientes IMSS y PEMEX sumados
datos_covid <- datos_covid |>
  casos(
```

```
    tipo_sector = c("IMSS", "PEMEX"), list_name = "imss_+_pemex",
    group_by_tipo_sector = TRUE
  )
head(datos_covid$`imss_+_pemex`)

# Solo los de BAJA CALIFORNIA
datos_covid <- datos_covid |>
  casos(entidades = c("BAJA CALIFORNIA"), list_name = "BC")
head(datos_covid$BC)

# Solo los de BAJA CALIFORNIA por residencia
datos_covid <- datos_covid |>
 casos(entidades = c("BAJA CALIFORNIA"), entidad_tipo = "Residencia", list_name = "residencia")
head(datos_covid$residencia)

# Agrupando casos por tipo de clasificacion
datos_covid <- datos_covid |>
  casos(
    entidades = c("BAJA CALIFORNIA", "BAJA CALIFORNIA SUR"),
    group_by_tipo_clasificacion = TRUE,
    list_name = "BC_BCS"
  )
head(datos_covid$BC_BCS)

# Regresa la suma de los de BC + BCS por tipo de paciente
datos_covid <- datos_covid |>
  casos(
    entidades = c("BAJA CALIFORNIA", "BAJA CALIFORNIA SUR"),
    group_by_tipo_clasificacion = FALSE,
    tipo_paciente = c("AMBULATORIO", "HOSPITALIZADO"),
    group_by_tipo_paciente = TRUE,
    list_name = "BC_+_BCS"
  )
head(datos_covid$`BC_+_BCS`)

# Si deseas agrupar por una variable que no este en las opciones
datos_covid <- datos_covid |>
  casos(
    group_by_entidad = FALSE,
    tipo_paciente = c("AMBULATORIO", "HOSPITALIZADO"),
    group_by_tipo_paciente = TRUE,
    list_name = "sexo",
    .grouping_vars = c("SEXO")
  )
head(datos_covid$sexo)

# Si no recuerdas la codificacion de los sexos puedes usar el diccionario:
datos_covid$sexo <- datos_covid$sexo |>
  dplyr::left_join(datos_covid$dict$SEXO, by = c("SEXO" = "CLAVE"))
head(datos_covid$sexo)

# Si no recuerdas todas las variables de la base puedes usar glimpse para ver por
# que otras variables puedes clasificar
```

```
datos_covid$dats |> dplyr::glimpse()


# Una vez hayas concluido tu trabajo no olvides desconectar
datos_covid$disconnect()
```

`cfr` *Case Fatality Rate (CFR)*

Description

Calcula la proporcion de enfermos que fallecen sobre todos los enfermos confirmados en distintas categorias (residencia / edad / etc)

Usage

```
cfr(
  datos_covid,
  entidades = c("AGUASCALIENTES", "BAJA CALIFORNIA", "BAJA CALIFORNIA SUR", "CAMPECHE",
    "CHIAPAS", "CHIHUAHUA", "CIUDAD DE MÉXICO", "COAHUILA DE ZARAGOZA", "COLIMA",
    "DURANGO", "GUANAJUATO", "GUERRERO", "HIDALGO", "JALISCO", "MÉXICO",
    "MICHOACÁN DE OCAMPO", "MORELOS", "NAYARIT", "NUEVO LEÓN", "OAXACA", "PUEBLA",
    "QUERÉTARO", "QUINTANA ROO", "SAN LUIS POTOSÍ", "SINALOA", "SONORA", "TABASCO",
    "TAMAULIPAS", "TLAXCALA", "VERACRUZ DE IGNACIO DE LA LLAVE", "YUCATÁN", "ZACATECAS"),
  group_by_entidad = TRUE,
  entidad_tipo = c("Unidad Medica", "Residencia", "Nacimiento"),
  fecha_tipo = c("Sintomas", "Ingreso", "Defuncion"),
  tipo_uci = c("SI", "NO", "NO APLICA", "SE IGNORA", "NO ESPECIFICADO"),
  group_by_tipo_uci = FALSE,
  tipo_clasificacion = c("Confirmados COVID"),
  group_by_tipo_clasificacion = FALSE,
  tipo_paciente = c("AMBULATORIO", "HOSPITALIZADO", "NO ESPECIFICADO"),
  group_by_tipo_paciente = FALSE,
  tipo_sector = c("CRUZ ROJA", "DIF", "ESTATAL", "IMSS", "IMSS-BIENESTAR", "ISSSTE",
    "MUNICIPAL", "PEMEX", "PRIVADA", "SEDENA", "SEMAR", "SSA", "UNIVERSITARIO",
    "NO ESPECIFICADO"),
  group_by_tipo_sector = FALSE,
  edad_cut = NULL,
  fill_NA = TRUE,
  list_name = "case fatality rate",
  .grouping_vars = c()
)
```

Arguments

`datos_covid`	(**obligatorio**) Lista de `tibbles` o duckdbs resultante de `descarga_datos_abiertos()` o `read_datos_abiertos()`

entidades (**opcional**) Vector con las entidades de las unidades medicas a analizar. Opciones: AGUASCALIENTES, BAJA CALIFORNIA, BAJA CALIFORNIA SUR, CAMPECHE, CHIAPAS, CHIHUAHUA, CIUDAD DE MEXICO, COAHUILA DE ZARAGOZA , COLIMA, DURANGO, GUANAJUATO, GUERRERO, HIDALGO, JALISCO, MEXICO, MICHOACAN DE OCAMPO, MORELOS,NAYARIT NUEVO LEON, OAXACA ,PUEBLA, QUERETARO,QUINTANA ROO, SAN LUIS POTOSI, SINALOA, SONORA, TABASCO, TAMAULIPAS,TLAXCALA, VERACRUZ DE IGNACIO DE LA LL YUCATAN, ZACATECAS.

group_by_entidad
(**opcional**) TRUE obtiene los casos para cada entidad reportando en cada fecha la entidad y los casos en dicha entidad. FALSE junta las entidades sumando sus casos en una sola observacion por cada fecha.

entidad_tipo (**opcional**) Indica a que se refiere las entidades seleccionadas. Elige una de las opciones: Unidad Medica (entidad de la unidad medica), Nacimiento (entidad de origen del individuo) o Residencia (entidad donde reside el individuo).

fecha_tipo (**opcional**) Selecciona si la fecha que se utiliza es la fecha de Ingreso (si aplica), la fecha de Sintomas o la de Defuncion (si aplica). El default es fecha de Sintomas.

tipo_uci (**opcional**) Vector con el tipo de valores para Unidad de Cuidado Intensivo (UCI) a incluir: SI,NO,NO APLICA,SE IGNORA,NO ESPECIFICADO. Por default se incluyen todos.

group_by_tipo_uci
(**opcional**) Booleana. El caso TRUE determina si regresa la base con cada fecha teniendo diferentes renglones uno para cada tipo_uci (es decir cada fecha se generan tantos observaciones como grupos de tipo de UCI) o bien en una sola fecha se suman todos los tipos de UCI (FALSE). El default es FALSE.

tipo_clasificacion
(**opcional**) Vector con el tipo de clasificaciones (por la prueba) a incluir:Sospechosos,Confirmados COVI Negativo a COVID, Inv\u00e1lido, No realizado

group_by_tipo_clasificacion
(**opcional**) Booleana determinando si regresa la base con cada entrada agrupada por tipo_clasificacion (es decir cada fecha se generan tantos observaciones como grupos de tipo de clasificación) en caso TRUE. Si FALSE suma todos los casos del tipo de clasificacion por fecha dando un solo numero por fecha. El defalt es FALSE.

tipo_paciente (**opcional**) Vector con el tipo de pacientes a incluir. Opciones: AMBULATORIO, HOSPITALIZADO, NO ESPECIFICADO. Por default se incluyen todos.

group_by_tipo_paciente
(**opcional**) Booleana determinando (caso TRUE) si regresa la base con cada entrada agrupada por tipo_paciente (es decir cada fecha se genera un renglon para AMBULATORIO, un renglon para HOSPITALIZADO, etc) o bien si se suman todos los grupos y cada fecha reporta solo la suma de estos (estilo AMBULATORIO + HOSPITALIZADO segun las categorias de tipo_paciente) El default es FALSE.

tipo_sector (**opcional**) Vector con los sectores del sistema de salud a incluir: CRUZ ROJA,DIF,ESTATAL,IMSS,IMSS-BIE MUNICIPAL,PEMEX, PRIVADA,SEDENA,SEMAR,SSA, UNIVERSITARIO,NO ESPECIFICADO.
Por default se incluyen todos.

group_by_tipo_sector
: (**opcional**) Booleana determina en el caso de TRUE si regresa la base con cada entrada agrupada por tipo_sector (es decir cada fecha tiene una entrada con los del IMSS, una entrada distinta con los de ISSSTE, etc) o bien en caso de FALSE se devuelve una sola entrada por fecha con la suma IMSS + ISSSTE + etc segun los sectores seleccionados. El default es FALSE.

edad_cut
: (**opcional**) Vector con secuencia de edades para hacer grupos. Por ejemplo edad_cut = c(0, 10, Inf) arma dos grupos de edad de 0 a 10 y de 10 a infinito o bien edad_cut = c(15, 20) deja sólo los registros entre 15 y 20 años. Por default es NULL y no arma grupos etarios.

fill_NA
: (**opcional**) Regresa observaciones para todas las combinaciones de variables incluyendo como NA donde no se observaron casos en el denominador. En caso contrario no se incluyen las filas donde no se observaron casos.

list_name
: (**opcional**) Asigna un nombre en la lista de datos a la base generada

.grouping_vars
: (**opcional**) Vector de variables adicionales de agrupacion de los conteos. Por ejemplo si se agrega .grouping_vars = 'DIABETES' entonces para cada fecha habra dos conteos de casos uno de los que tienen diabetes y uno de los que no.

Details

El case fatality rate se define como

$$\frac{\#Defunciones}{Totaldeenfermos}$$

Si se utiliza la opción tipo_clasificacion se puede cambiar la definicion de enfermo (por default se incluyen solamente "Confirmados COVID").

Value

Une a la lista de datos_covid una nueva entrada de nombre list_name (default: case fatality rate) con una base de datos (tibble o duckdb) con los resultados agregados.

- case fatality rate - Base de datos generara con los datos agregados (el nombre cambia si se usa list_name).
- dict - Diccionario de datos
- dats - Datos originales (conexion a duckdb o tibble)
- disconnect - Función para desconectarte de duckdb
- ... - Cualquier otro elemento que ya existiera en datos_covid

See Also

descarga_datos_abiertos() numero_pruebas() chr() estima_rt() positividad() casos()

Examples

```
# Para el ejemplo usaremos los datos precargados (datosabiertos) pero tu puedes
# correr el ejemplo descargando informacion mas reciente.
datos_covid <- datosabiertos

# Casos a nivel nacional por entidad
datos_covid <- datos_covid |> cfr()
head(datos_covid$`case fatality rate`)

# Agregando todos los estados
datos_covid <- datos_covid |>
  cfr(list_name = "cfr_nacional", group_by_entidad = FALSE)
head(datos_covid$`cfr_nacional`)

# CFR en Baja California
datos_covid <- datos_covid |>
  cfr(entidades = c("BAJA CALIFORNIA"), list_name = "cfr_bc")
head(datos_covid$`cfr_bc`)

# Calcula el CFR suponiendo toda la base son confirmados
datos_covid <- datos_covid |>
  cfr(
    entidades = c("BAJA CALIFORNIA", "BAJA CALIFORNIA SUR"),
    tipo_clasificacion = c(
      "Sospechosos", "Confirmados COVID",
      "Negativo a COVID", "Inv\u00e1lido", "No realizado"
    ),
    group_by_tipo_clasificacion = TRUE, list_name = "bc_bcs_cfr"
  )
head(datos_covid$`bc_bcs_cfr`) # Los NA es porque no habia observaciones en el denominador

# Distinguiendo entre ambulatorio y hospitalizado
datos_covid <- datos_covid |>
  cfr(
    tipo_paciente = c("AMBULATORIO", "HOSPITALIZADO"),
    group_by_tipo_paciente = TRUE,
    list_name = "cfr_paciente"
  )
head(datos_covid$cfr_paciente)

# CFR en distintos grupos de edad (0 a 20, 20 a 60 y 60+)
datos_covid <- datos_covid |>
  cfr(edad_cut = c(0, 20, 60, Inf), list_name = "cfr_edad")
head(datos_covid$cfr_edad)

# Si deseas agrupar por una variable que no este en las opciones
datos_covid <- datos_covid |>
  cfr(.grouping_vars = c("DIABETES"), list_name = "cfr_diab")
head(datos_covid$cfr_diab)

# Finalmente desconectamos
```

```
datos_covid$disconnect()
```

---

check_sites *Funcion para verificar la existencia de los sitios web de datos de covid*

Description

La funcion recorre cada uno de los sitios y verifica su existencia.

Usage

```
check_sites(covid_data = TRUE, dictionnary = TRUE)
```

Arguments

covid_data	(**opcional**) Variable booleana TRUE si verifica los sitios de datos y FALSE si no los verifica
dictionnary	(**opcional**) Variable booleana TRUE si verifica los sitios del diccionario y FALSE si no lo verifica

Value

Devuelve TRUE si todos los sitios web existen, FALSEen caso de que no.

Examples

```
# Verificamos que existan los sitios cambiando a TRUE cualquiera:
check_sites(covid_data = FALSE, dictionnary = FALSE)
```

---

chr *Case Hospitalization Rate (CHR)*

Description

chr Calcula la proporción de enfermos que resultan hospitalizados sobre todos los enfermos confirmados en distintas categorías (residencia / edad / etc)

Usage

```
chr(
  datos_covid,
 entidades = c("AGUASCALIENTES", "BAJA CALIFORNIA", "BAJA CALIFORNIA SUR", "CAMPECHE",
   "CHIAPAS", "CHIHUAHUA", "CIUDAD DE MÉXICO", "COAHUILA DE ZARAGOZA", "COLIMA",
    "DURANGO", "GUANAJUATO", "GUERRERO", "HIDALGO", "JALISCO", "MÉXICO",
  "MICHOACÁN DE OCAMPO", "MORELOS", "NAYARIT", "NUEVO LEÓN", "OAXACA", "PUEBLA",
  "QUERÉTARO", "QUINTANA ROO", "SAN LUIS POTOSÍ", "SINALOA", "SONORA", "TABASCO",
  "TAMAULIPAS", "TLAXCALA", "VERACRUZ DE IGNACIO DE LA LLAVE", "YUCATÁN", "ZACATECAS"),
  group_by_entidad = TRUE,
  entidad_tipo = c("Unidad Medica", "Residencia", "Nacimiento"),
  fecha_tipo = c("Sintomas", "Ingreso", "Defuncion"),
  tipo_clasificacion = c("Confirmados COVID"),
  group_by_tipo_clasificacion = FALSE,
  incluir_paciente_no_especificado = FALSE,
 tipo_sector = c("CRUZ ROJA", "DIF", "ESTATAL", "IMSS", "IMSS-BIENESTAR", "ISSSTE",
   "MUNICIPAL", "PEMEX", "PRIVADA", "SEDENA", "SEMAR", "SSA", "UNIVERSITARIO",
   "NO ESPECIFICADO"),
  group_by_tipo_sector = FALSE,
  defunciones = FALSE,
  edad_cut = NULL,
  fill_NA = TRUE,
  list_name = "case hospitalization rate",
  .grouping_vars = c()
)
```

Arguments

datos_covid	(**obligatorio**) Lista de tibbles o duckdbs resultante de descarga_datos_abiertos() o read_datos_abiertos()
entidades	(**opcional**) Vector con las entidades de las unidades medicas a analizar. Opciones: AGUASCALIENTES, BAJA CALIFORNIA, BAJA CALIFORNIA SUR, CAMPECHE, CHIAPAS, CHIHUAHUA, CIUDAD DE MEXICO, COAHUILA DE ZARAGOZA , COLIMA, DURANGO, GUANAJUATO, GUERRERO, HIDALGO, JALISCO, MEXICO, MICHOACAN DE OCAMPO, MORELOS,NAYARIT NUEVO LEON, OAXACA ,PUEBLA, QUERETARO,QUINTANA ROO, SAN LUIS POTOSI, SINALOA, SONORA, TABASCO, TAMAULIPAS,TLAXCALA, VERACRUZ DE IGNACIO DE LA LL YUCATAN, ZACATECAS.
group_by_entidad	(**opcional**) TRUE obtiene los casos para cada entidad reportando en cada fecha la entidad y los casos en dicha entidad. FALSE junta las entidades sumando sus casos en una sola observacion por cada fecha.
entidad_tipo	(**opcional**) Indica a que se refiere las entidades seleccionadas. Elige una de las opciones: Unidad Medica (entidad de la unidad medica), Nacimiento (entidad de origen del individuo) o Residencia (entidad donde reside el individuo).
fecha_tipo	(**opcional**) Selecciona si la fecha que se utiliza es la fecha de Ingreso (si aplica), la fecha de Sintomas o la de Defuncion (si aplica). El default es fecha de Sintomas.

`tipo_clasificacion`
: (**opcional**) Vector con el tipo de clasificaciones (por la prueba) a incluir:`Sospechosos`,`Confirmados COVI` `Negativo a COVID`, `Inv\u00e1lido`, `No realizado`

`group_by_tipo_clasificacion`
: (**opcional**) Booleana determinando si regresa la base con cada entrada agrupada por `tipo_clasificacion` (es decir cada fecha se generan tantos observaciones como grupos de tipo de clasificación) en caso `TRUE`. Si `FALSE` suma todos los casos del tipo de clasificacion por fecha dando un solo numero por fecha. El defalt es `FALSE`.

`incluir_paciente_no_especificado`
: (**opcional**) Si en el denominador se incluyen los pacientescuyo tipo es `NO ESPECIFICADO`. Por default es `FALSE` por lo que sólo se incluyen `AMBULATORIO`, `HOSPITALIZADO`.

`tipo_sector`
: (**opcional**) Vector con los sectores del sistema de salud a incluir: `CRUZ ROJA`,`DIF`,`ESTATAL`,`IMSS`,`IMSS-BIE` `MUNICIPAL`,`PEMEX`, `PRIVADA`,`SEDENA`,`SEMAR`,`SSA`, `UNIVERSITARIO`,`NO ESPECIFICADO`. Por default se incluyen todos.

`group_by_tipo_sector`
: (**opcional**) Booleana determina en el caso de `TRUE` si regresa la base con cada entrada agrupada por `tipo_sector` (es decir cada fecha tiene una entrada con los del `IMSS`, una entrada distinta con los de `ISSSTE`, etc) o bien en caso de `FALSE` se devuelve una sola entrada por fecha con la suma `IMSS + ISSSTE + etc` segun los sectores seleccionados. El default es `FALSE`.

`defunciones`
: (**opcional**) Booleana si incluir sólo defunciones `TRUE` o a todos `FALSE`. El default es `FALSE`.

`edad_cut`
: (**opcional**) Vector con secuencia de edades para hacer grupos. Por ejemplo `edad_cut = c(0, 10, Inf)` arma dos grupos de edad de 0 a 10 y de 10 a infinito o bien `edad_cut = c(15, 20)` deja sólo los registros entre 15 y 20 años. Por default es `NULL` y no arma grupos etarios.

`fill_NA`
: (**opcional**) Regresa observaciones para todas las combinaciones de variables incluyendo como `NA` donde no se observaron casos en el denominador. En caso contrario no se incluyen las filas donde no se observaron casos.

`list_name`
: (**opcional**) Asigna un nombre en la lista de datos a la base generada

`.grouping_vars`
: (**opcional**) Vector de variables adicionales de agrupacion de los conteos. Por ejemplo si se agrega `.grouping_vars = 'DIABETES'` entonces para cada fecha habra dos conteos de casos uno de los que tienen diabetes y uno de los que no.

Details

El case hospitalization rate se define como

$$\frac{\#Hospitalizados}{Totaldeenfermos}$$

Si se utiliza la opción `incluir_paciente_no_especificado` se puede cambiar la definicion de **Total de enfermos** para incluir a los pacientes que dicen `NO ESPECIFICADO`. Estos por default se excluyen justo por su naturaleza desconocida.

Value

Une a la lista de `datos_covid` una nueva entrada de nombre `list_name` (default: `case hospitalization rate`) con una base de datos (`tibble` o duckdb) con los resultados agregados.

- `case hospitalization rate` - Base de datos generara con los datos agregados (el nombre cambia si se usa `list_name`).
- dict - Diccionario de datos
- dats - Datos originales (conexion a duckdb o `tibble`)
- disconnect - Función para desconectarte de duckdb
- ... - Cualquier otro elemento que ya existiera en `datos_covid`

See Also

`descarga_datos_abiertos()` `numero_pruebas()` `cfr()` `estima_rt()` `positividad()` `casos()`

Examples

```
# Para el ejemplo usaremos los datos precargados (datosabiertos) pero tu puedes
# correr el ejemplo descargando informacion mas reciente.
datos_covid <- datosabiertos

# Casos a nivel nacional
datos_covid <- datos_covid |> chr()
head(datos_covid$`case hospitalization rate`)

# Nacional

datos_covid <- datos_covid |> chr(list_name = "chr_nacional", group_by_entidad = FALSE)
head(datos_covid$`chr_nacional`)

# CHR en IMSS e ISSSTE
datos_covid <- datos_covid |>
 chr(tipo_sector = c("IMSS", "ISSSTE"), list_name = "chimss", group_by_tipo_sector = TRUE)
head(datos_covid$`chimss`)

# Calcula el CHR sobre toda la base
datos_covid <- datos_covid |>
  chr(
    tipo_clasificacion = c(
      "Sospechosos", "Confirmados COVID",
      "Negativo a COVID", "Inv\u00e1lido", "No realizado"
    ),
    group_by_tipo_clasificacion = TRUE, list_name = "chr_todos"
  )
head(datos_covid$`chr_todos`)

# Distinguiendo sólo entre defunciones
datos_covid <- datos_covid |>
  chr(defunciones = TRUE, list_name = "chr_defun")
```

```
head(datos_covid$`chr_defun`)

# Si deseas agrupar por una variable que no este en las opciones
datos_covid <- datos_covid |>
  chr(.grouping_vars = c("DIABETES"), list_name = "chr_diab")
head(datos_covid$chr_diab)

# Finalmente desconectamos
datos_covid$disconnect()
```

`datosabiertos` *Datos abiertos de COVID-19*

Description

Base de datos que contiene una extraccion pequenia de la base de datos abiertos que se obtiene mediante `descarga_datos_abiertos`.

Usage

```
datosabiertos
```

Format

Una lista con tres objetos

dats Base de datos de la DGE actualizada el 8 septiembre 2022 filtrada a BC, BCS en julio y agosto del 2022

dict Diccionario de datos

disconnect Funcion que simula desconexion de duckdb

Details

El proposito de esta base es poder probar las funciones que se aplican sobre `datos_covid`. La base contiene solo las entidades de `BAJA CALIFORNIA` y `BAJA CALIFORNIA SUR` durante julio y agosto del 2021.

Source

https://www.gob.mx/salud/documentos/datos-abiertos-152127

descarga_datos_abiertos *Descarga de datos abiertos*

Description

Funcion para la descarga de datos abiertos de la Direccion General de Epidemiologia (DGE)

Usage

```
descarga_datos_abiertos(
  dbdir = tempfile(fileext = ".duckdb"),
  sites.covid = get_sites_covid(),
  site.covid.dic = get_site_dic(),
  read_format = c("duckdb", "tibble"),
  drv = duckdb::duckdb(),
  pragma_memory_limit = Sys.getenv("pragma_memory_limit"),
  tblname = "covidmx",
  colClasses = get_col_class(),
  download_process = c("pins", "download.file"),
  unzip_command = Sys.getenv("unzip_command"),
  unzip_args = Sys.getenv("unzip_args"),
  unzip_args_dict = list(exdir = ".", overwrite = TRUE),
  check_unzip_install = TRUE,
  clear_zip = (download_process[1] != "pins"),
  clear_csv = TRUE,
  use_dict = TRUE,
  datos_abiertos_zip_paths = NULL,
  datos_abiertos_unzipped_path = NULL,
  datos_abiertos_tbl = NULL,
  diccionario_zip_path = NULL,
  diccionario_unzipped_path = NULL,
  diccionario = NULL,
  quiet = FALSE,
  cache_datos = NULL,
  use_cache_on_failure = TRUE,
  cache_diccionario = NULL,
  force_download = FALSE,
  show_warnings = TRUE,
  board_url_name = "datos_abiertos",
  board_url_name_dict = "diccionario_covid",
 download_file_args = list(method = "curl", destfile = tempfile(), quiet = quiet),
  download_file_args_dict = download_file_args,
  descarga_db_datos_abiertos_tbl_args = list(),
  descarga_db_diccionario_ssa_args = list(),
  ...
)
```

Arguments

dbdir	(**opcional**) Direccion donde guardar la base de datos con terminacion .duckdb. Corresponde al argumento de duckdb::dbConnect__duckdb_driver()
sites.covid	(**opcional**) Sitios web con el vinculo a los archivos .zip de l os datos abiertos. Puedes cambiarlo por uno de los historicos, por ejemplo. La estructura es c("nombre" = "url", "nombre2" = "url2"). La ultima verificacion del sitio web default fue el 6 de septiembre del 2022.
site.covid.dic	(**opcional**) Sitio desde el cual descarga del diccionario de datos. La ultima verificacion del sitio fue el 6 de septiembre 2022.
read_format	(**opcional**) "duckdb" o "tibble" establece el formato de lectura de la base de datos. En la mayoria de los casos "tibble" va a resultar en un error de memoria. La opcion de "duckdb" siempre es mas rapida por lo cual es el default.
drv	(**opcional**) Un driver para dbConnect (default duckdb::duckdb())
pragma_memory_limit	(**opcional**) Limite de memoria para el programa (ver PRAGMAS). Cambialo a que sea mas o menos la mitad de tu RAM. La forma mas sencilla es como una variable ambiental con Sys.setenv('pragma_memory_limit' = '1GB') por ejemplo para un limite de 1 gigabyte.
tblname	(**opcional**) Nombre de la tabla de duckdb donde guardar los datos por default se llama covidmx. Solo es relevante si estas usando el mismo dbdir para otro proyecto distinto.
colClasses	(**opcional**) Clases de la columna para leer en duckdb::read_csv_duckdb().
download_process	(**opcional**) Metodo para descargar ya sea pins o download.file. Se recomienda pins pues guarda en memoria la fecha de la ultima descarga y analiza si ha pasado mas de un dia desde la descarga. En caso afirmativo verifica si el archivo ha cambiado y si hubo cambios entonces lo descarga.
unzip_command	(**opcional**) Forma de extraer la base de datos de datos abiertos si unzip falla. La forma de llamarla es con system2(unzip_command, args = c(unzip_args, file_download_data)).
unzip_args	(**opcional**) Argumentos de extraccion de la base de datos de datos abiertos si unzip falla. La forma de llamarla es con system2(unzip_command, args = c(unzip_args, file_download_data)).
unzip_args_dict	(**opcional**) Lista de argumentos para usar utils::unzip en el diccionario de datos.
check_unzip_install	(**opcional**) Bandera de verificacion para checar si tienes lo necesario para unzippear los datos en el caso de que unzip no sirva.
clear_zip	(**opcional**) Si borrar los archivos .zip descargados para el diccionario y los datos abiertos. No se recomienda si estas usando pins. Ve la nota para mas informacion.
clear_csv	(**opcional**) Si borrar los archivos .csv que se generan despues de abrir el zip. El default es que si pues en general solo requieres el duckdb.

use_dict (**opcional**) Si descargar el diccionario de site.covid.dic.

datos_abiertos_zip_paths
(**opcional**) Camino a los datos abiertos si ya los descargaste en zip

datos_abiertos_unzipped_path
(**opcional**) Camino a los datos abiertos csv si ya los descargaste y descomprimiste el archivo zip en un csv

datos_abiertos_tbl
(**opcional**) Camino a un archivo .duckdb con los datos formateados

diccionario_zip_path
(**opcional**) Camino al diccionario si ya losdescargaste en zip

diccionario_unzipped_path
(**opcional**) Camino al diccionario csv si ya lo descargaste y descomprimiste el archivo zip en un csv

diccionario (**opcional**) Lo que resulta de realizar una descarga del diccionario usando descarga_diccionario

quiet (**opcional**) Variable para no mostrar mensajes

cache_datos (**opcional**) Direccion donde guardar los datos en memoria usando pins para no tener que volver a descargarlos si nada ha cambiado

use_cache_on_failure
(**opcional**) Booleana. Establece que si no se pueden descargar datos nuevos utilice los que tenga en memoria. Por default es TRUE.

cache_diccionario
(**opcional**) Direccion donde guardar el diccionario en memoria usando pins para no tener que volver a descargarlo si nada ha cambiado

force_download (**opcional**) Analiza si cambio el pin y descarga datos nuevos en caso afirmativo aunque haya pasado menos de un dia.

show_warnings (**opcional**) si arrojar warnings

board_url_name (**opcional**) Establece el nombre del pins::board_url para los datos abiertos (si ya usas pins para que no se empalme). Por default se llama datos_abiertos

board_url_name_dict
(**opcional**) Establece el nombre del pins::board_url para los datos abiertos. Por default se llama diccionario_covid

download_file_args
(**opcional**) Lista de argumentos adicionales para download.file de los datos si se elige este metodo para descargar.

download_file_args_dict
(**opcional**) Lista de argumentos adicionales para download.file del diccionario si se elige este metodo de descarga.

descarga_db_datos_abiertos_tbl_args
(**opcional**) Lista con argumentos adicionales para el pins::pin_download de datos abiertos

descarga_db_diccionario_ssa_args
(**opcional**) Lista con argumentos adicionales para el pins::pin_download de datos abiertos

... (**opcional**) Parametros adicionales para DBI::dbConnect.

Details

La funcion de descarga principal es `descarga_datos_abiertos()` llama las siguientes funciones en orden:

- `descarga_diccionario()` Se encarga de descargar y formatear el diccionario de datos
- `descarga_db()` Se encarga de descargar y formatear la base de datos
- `pega_db_datos_abiertos_tbl_y_diccionario()` Pega ambos en el formato lista de `covidmx`

A su vez `descarga_diccionario()` ejecuta las siguientes para obtener el diccionario de datos:

- `descarga_db_diccionario_ssa()` Descarga el diccionario de la DGE
- `unzip_db_diccionario_ssa()` Libera el archivo `zip` descargado
- `parse_db_diccionario_ssa()` Genera una lista de tiblles con el diccionario por variable

Por otro lado,`descarga_db()` ejecuta las siguientes para obtener los datos abiertos:

- `descarga_db_datos_abiertos_tbl()` Descarga las bases de datos de covid de la DGE
- `unzip_db_datos_abiertos_tbl()` Libera el archivo `zip` descargado
- `parse_db_datos_abiertos_tbl()` Genera una base de datos en `duckdb` (o `tibble`) c on la informacion

Si en algun momento se interrumpio la descarga o hubo problemas de conexion o detuviste el proceso de generacion de la base de datos abiertos puedes llamar a las funciones de `read_datos_abiertos()`.

Value

Lista de valores:

- dats - Tabla conectada mediante `duckdb::dbConnect__duckdb_driver()` (si `duckdb`) o tibble (si `tibble`)
- disconnect - Funcion para cerrar la conexion a la base de datos.
- dict - Lista de `tibbles` con el diccionario de datos para cada variable

Memoria `RAM`

Si tienes RAM que te sobre puedes no crear una base de datos en `duckdb` sino leer directo el archivo `csv`. Esto se logra con `read_format = tibble`. No lo recomiendo pues puedes terminar con tu sesion de `R` si se te acaba la memoria.

Windows Para abrir el archivo `.zip` quiza requieras tambien descargar e instalar `7Zip` por default el sistema lo busca en `C:\\Program Files\\7-Zip\\7z.exe` pero si no esta ese directorio es necesario que en `unzip_command` especifiques el camino donde se instalo `7z.exe`.

Uso de pins

Para almacenar los datos se utiliza un pequenio cambio sobre la libreria pins. Los datos se descargan y se almacenan en cache junto con informacion sobre cuando fue la descarga. Si no ha pasado un dia desde la ultima descarga no se descarga nada nuevo. Si los datos que se tienen no han cambiado respecto a lo que esta en linea tampoco se vuelven a descargar aunque haya pasado mas de un dia.

Si se te fue el Internet No te preocupes, pins lee tu descarga mas reciente.

Para ver donde estan descargados tus datos usa pins::board_cache_path(). Para borrarlos usa pins::cache_prune().

Metodos de unzip

Por default el programa intenta abrir la base de datos con utils::unzip(). Sin embargo historicamente la base de datos ha estado codificada de tal forma que utils::unzip() no pueda abrirla. Para ello se utilizaban diferentes comandos en particular el default que hemos visto funcionaba son los comandos de terminal unzip (en Linux/OSX) y 7zip (en Windows). En caso de ser requeridos el sistema te lo hara saber junto con las instrucciones de instalacion

Note

No te recomiendo borrar el cache con clear_zip o editarlo por cualquier otro medio si estas usando pins pues puede romperse la dependencia. Si accidentalmente lo borraste usa pins::board_cache_path() para ir al path y borrar manualmente toda la carpeta.

References

Secretaría de Salud (2022). Datos Abiertos de COVID-19 URL: https://www.gob.mx/salud/documentos/datos-abiertos-152127

See Also

read_datos_abiertos() descarga_datos_red_irag() descarga_datos_variantes_GISAID() casos()

Examples

```
# Descarga de la base de datos junto con diccionario en duckdb y la guarda en
# un archivo temporal.
# Puede cambiarse el dlink por el adecuado o dejarse en blanco
# quita la opción de sites.covid t site.covid.dic para descargar los de la DGE.
# Esto es solo un ejemplo.
file_duck <- tempfile(fileext = ".duckdb")

#Estos links deben omitirse en una corrida normal. Se incluyen por ahora como ejemplo
#pero las opciones site.covid.dic y sites.covid deben eliminarse de abajo.
dlink   <- "https://github.com/RodrigoZepeda/covidmx/raw/main/datos_abiertos_covid19.zip"
diclink  <- "https://github.com/RodrigoZepeda/covidmx/raw/main/diccionario_datos_covid19.zip"

#En el ejemplo de R por normas de CRAN tenemos que hacerlo así pero en tu
```

```
#computadora puedes solo usar descargar datos sin el if else
if (RCurl::url.exists(dlink) & RCurl::url.exists(diclink)){
  datos_covid <- descarga_datos_abiertos(
    dbdir = file_duck,
    sites.covid = dlink,
    site.covid.dic = diclink,
    show_warnings = FALSE
  )
  # Luego haces algo con esos datos...

  # Cuando terminas cierras la sesion:
  datos_covid$disconnect()

  # Despues podras leerlos con read_datos_abiertos cuando quieras:
  datos_covid <- read_datos_abiertos(dbdir = file_duck, site.covid.dic = diclink)
  datos_covid$disconnect()

  # Si no pones `dbdir` nota que los datos se guardan en un archivo temporal que se elimina
  # al cerrar tu sesion
  datos_covid <- descarga_datos_abiertos(sites.covid = dlink, show_warnings = FALSE,
                      site.covid.dic = diclink)

} else {
  datos_covid <- datosabiertos
}

# Desconectamos
datos_covid$disconnect()
```

`descarga_datos_red_irag` *Descarga la base de datos de ocupacion hospitalaria de la red IRAG*

Description

`descarga_datos_red_irag` Lee los datos de ocupacion hospitalaria de la `RED IRAG` disponibles en https://www.gits.igg.unam.mx/red-irag-dashboard/reviewHome# y analizados a traves de RodrigoZepeda/CapacidadHospitalariaMX

Usage

```
descarga_datos_red_irag(
  nivel = c("Estatal", "Unidad Médica"),
  cache = NULL,
  use_cache_on_failure = TRUE,
  quiet = TRUE,
  force_download = FALSE,
  show_warnings = TRUE,
```

```
  ...
)
```

Arguments

nivel	(**opcional**) Regresa la ocupacion "Estatal"(default) o por "Unidad Medica"
cache	(**opcional**) cache para pins::board_url(). Representa el directorio donde se almacenaran los datos descargados en formato de pins.
use_cache_on_failure	(**opcional**) parametro para pins::board_url(). En caso de TRUE (default) si no puede descargar nueva informacion utiliza la que ya tiene en memoria aunque sea vieja.
quiet	(**opcional**) booleana para no imprimir mensajes en la consola.
force_download	(**opcional**) analiza si cambio el pin y descarga datos nuevos en caso afirmativo.
show_warnings	(**opcional**) si arrojar warnings o callar
...	parametros adicionales para pins::pin_download().

Details

Los datos de Red IRAG son descargados diariamente de manera automatica en Github: RodrigoZepeda/CapacidadHospitalari y esta funcion los lee de ahi. Puede que esten un poco rezagados respecto a la pagina de la RED IRAG (https://www.gits.igg.unam.mx/red-irag-dashboard/reviewHome#) pero el rezago nunca es mayor a un dia.

Value

tibble con los datos de ocupacion hospitalaria

- Unidad médica - En caso nivel = "Unidad Medica" la unidad a la que pertenecen los datos
- Institución - Institucion a la que pertenece la unidad medica
- Estado - Entidad federativa de la informacion o de la unidad
- CLUES - La Clave Unica de Establecimientos de Salud para la unidad (si nivel = "Unidad Medica")
- Fecha - La fecha a la cual corresponde dicha ocupacion
- Actualizacion - La fecha de actualizacion ultima de los datos.
- Hospitalizados (%) - Porcentaje de ocupacion en camas de hospitalizacion.
- Ventilación (%) - Porcentaje de ocupacion en ventiladores.
- UCI y Ventilación (%) - Porcentaje de ocupacion en unidades de cuidado intensivo con ventilacion.

References

Secretaría de Salud (2022). Sistema de Información de la Red IRAG URL: https://www.gits.igg.unam.mx/red-irag-dashboard/reviewHome

Zepeda-Tello, R. (2022). Descarga Automática de Datos de la Red IRAG URL: https://github.com/RodrigoZepeda/CapacidadHospitalariaMX

See Also

descarga_datos_variantes_GISAID() descarga_datos_abiertos() read_datos_abiertos()

Examples

```
# Descarga de datos estatales
url_global <- paste0(
  "https://media.githubusercontent.com/media/RodrigoZepeda/",
  "CapacidadHospitalariaMX/master/processed/"
)

if (RCurl::url.exists(paste0(url_global, "HospitalizacionesMX_estatal.csv"))) {
  ocupacion_hospitalaria <- descarga_datos_red_irag("Estatal", show_warnings = FALSE)
}

# También puedes hacer la descarga por unidad medica
# Descarga de datos por unidad medica
if (RCurl::url.exists(paste0(url_global, "HospitalizacionesMX_unidad_medica.csv"))) {
  ocupacion_unidad <- descarga_datos_red_irag("Unidad Medica", show_warnings = FALSE)
}
```

---

descarga_datos_variantes_GISAID

Lee la base de datos de variantes de COVID-19 en Mexico generada por GISAID

Description

descarga_datos_variantes_GISAID Lee los datos de variantes del reporte nacional diario en RodrigoZepeda/VariantesCovid creado a partir de la informacion de la Global Initiative on Sharing Avian Influenza Data (GISAID)

Usage

```
descarga_datos_variantes_GISAID(
  nivel = c("nacional", "cdmx"),
  cache = NULL,
  use_cache_on_failure = TRUE,
  quiet = FALSE,
  force_download = FALSE,
  show_warnings = TRUE,
  ...
)
```

Arguments

nivel	(**opcional**) si se desea descargar informacion "nacional" (default) o de la Ciudad de Mexico: "cdmx".
cache	(**opcional**) cache para pins::board_url(). Representa el directorio donde se almacenaran los datos descargados en formato de pins.
use_cache_on_failure	(**opcional**) parametro para pins::board_url(). En caso de TRUE (default) si no puede descargar nueva informacion utiliza la que ya tiene en memoria aunque sea vieja.
quiet	(**opcional**) booleana para no imprimir mensajes en la consola.
force_download	(**opcional**) analiza si cambio el pin y descarga datos nuevos en caso afirmativo.
show_warnings	(**opcional**) si arrojar warnings o callar
...	parametros adicionales para pins::pin_download().

Details

Cada vez que uses estos datos necesitas citar a **GISAID** (ver referencias) asi como el reporte en RodrigoZepeda/VariantesCovid

Los datos son descargados de manera automatica en mi Github: RodrigoZepeda/VariantesCovid el programa descarga_datos_variantes_GISAID se conecta a dicho repositorio, busca si la informacion esta actualizada y si si la descarga, si no, utiliza informacion almacenada en el cache local.

La descarga usa el paquete pins

Value

tibble con los datos de porcentuales de variantes

- variant - La variante clasificada mediante Pangolin
- semana - Semana epidemiologica lubridate::epiweek() a la que corresponde la variante
- ano - Anio al que corresponde la toma de muestra
- n - El total de muestras de dicha semana registradas para esa variante
- freq - La proporcion de las variantes de dicha semana ocupada por dicha variante. Se obtiene dividiendo n/sum(n) para cada semana.
- Actualizacion - La fecha de actualizacion ultima de los datos.
- Fuente - La fuente desde la cual se obtuvo la informacion de dicha variante.

References

Khare, S., et al (2021) GISAID's Role in Pandemic Response. China CDC Weekly, 3(49): 1049-1051. doi:10.46234/ccdcw2021.255 PMCID: 8668406

Elbe, S. and Buckland-Merrett, G. (2017) Data, disease and diplomacy: GISAID's innovative contribution to global health. Global Challenges, 1:33-46. doi:10.1002/gch2.1018 PMCID: 31565258

Shu, Y. and McCauley, J. (2017) GISAID: from vision to reality. EuroSurveillance, 22(13) doi:10.2807/1560-7917.ES.2017.22.13.30494 PMCID: PMC5388101

Zepeda-Tello, R. (2022). Reporte Nacional de Variantes de COVID-19. URL: https://github.com/RodrigoZepeda/VariantesCovid

See Also

descarga_datos_red_irag() descarga_datos_abiertos() read_datos_abiertos()

Examples

```
# Descarga de variantes a nivel nacional
url_global <- "https://raw.githubusercontent.com/RodrigoZepeda/VariantesCovid/main/tablas/"
if (RCurl::url.exists(paste0(url_global, "Proporcion_variantes_nacional.csv"))) {
  variantes_covid <- descarga_datos_variantes_GISAID("nacional")
}

# Descarga de variantes para CDMX
if (RCurl::url.exists(paste0(url_global, "Proporcion_variantes_cdmx.csv"))) {
  variantes_covid <- descarga_datos_variantes_GISAID("cdmx")
}
```

---

descarga_db *Auxiliares para la descarga de datos abiertos*

Description

Conjunto de funciones para apoyar la descarga de datos abiertos de la Direccion General de Epidemiologia (DGE)

La funcion de descarga principal es descarga_datos_abiertos() llama las siguientes funciones en orden:

- descarga_diccionario() Se encarga de descargar y formatear el diccionario de datos
- descarga_db() Se encarga de descargar y formatear la base de datos
- pega_db_datos_abiertos_tbl_y_diccionario() Pega ambos en el formato lista de covidmx

A su vez descarga_diccionario() ejecuta las siguientes para obtener el diccionario de datos:

- descarga_db_diccionario_ssa() Descarga el diccionario de la DGE
- unzip_db_diccionario_ssa() Libera el archivo zip descargado
- parse_db_diccionario_ssa() Genera una lista de tiblles con el diccionario por variable

Por otro lado,descarga_db() ejecuta las siguientes para obtener los datos abiertos:

- descarga_db_datos_abiertos_tbl() Descarga las bases de datos de covid de la DGE
- unzip_db_datos_abiertos_tbl() Libera el archivo zip descargado

- parse_db_datos_abiertos_tbl() Genera una base de datos en duckdb (o tibble) con la informacion

Si en algun momento se interrumpio la descarga o hubo problemas de conexion o detuviste el proceso de generacion de la base de datos abiertos puedes llamar a las funciones de read_datos_abiertos().

Usage

```
descarga_db(
  read_format = c("duckdb", "tibble"),
  tblname = "covidmx",
  pragma_memory_limit = Sys.getenv("pragma_memory_limit"),
  drv = duckdb::duckdb(),
  dbdir = tempfile(fileext = ".duckdb"),
  colClasses = get_col_class(),
  sites.covid = get_sites_covid(),
  download_process = c("pins", "download.file"),
  unzip_command = Sys.getenv("unzip_command"),
  unzip_args = Sys.getenv("unzip_args"),
  check_unzip_install = TRUE,
  clear_zip = (download_process[1] != "pins"),
  clear_csv = TRUE,
  force_download = FALSE,
  show_warnings = TRUE,
  datos_abiertos_zip_paths = NULL,
  datos_abiertos_unzipped_path = NULL,
  datos_abiertos_tbl = NULL,
  quiet = FALSE,
  board_url_name = "datos_abiertos",
  cache = NULL,
  use_cache_on_failure = TRUE,
 download_file_args = list(method = "curl", destfile = tempfile(), quiet = quiet),
  descarga_db_datos_abiertos_tbl_args = list(),
  ...
)

descarga_diccionario(
  download_process = c("pins", "download.file"),
  site.covid.dic = get_site_dic(),
  quiet = FALSE,
  clear_zip = download_process[1] != "pins",
  clear_csv = TRUE,
  diccionario_zip_path = NULL,
  diccionario_unzipped_path = NULL,
  diccionario = NULL,
  board_url_name_dict = "diccionario_covid",
  cache_diccionario = NULL,
  use_cache_on_failure = TRUE,
  force_download = FALSE,
```

```
  show_warnings = TRUE,
 download_file_args_dict = list(method = "curl", destfile = tempfile(), quiet = quiet),
  unzip_args_dict = list(exdir = ".", overwrite = TRUE),
  descarga_db_diccionario_ssa_args = list()
)

descarga_db_datos_abiertos_tbl(
  download_process = c("pins", "download.file"),
  sites.covid = get_sites_covid(),
  quiet = FALSE,
  board_url_name = "datos_abiertos",
  cache = NULL,
  use_cache_on_failure = TRUE,
  force_download = FALSE,
  show_warnings = TRUE,
 download_file_args = list(method = "curl", destfile = tempfile(), quiet = quiet),
  ...
)

descarga_db_diccionario_ssa(
  download_process = c("pins", "download.file"),
  site.covid.dic = get_site_dic(),
  quiet = FALSE,
  board_url_name_dict = "diccionario_covid",
  cache_diccionario = NULL,
  use_cache_on_failure = TRUE,
  force_download = FALSE,
  show_warnings = TRUE,
 download_file_args_dict = list(method = "curl", destfile = tempfile(), quiet = quiet),
  ...
)

unzip_db_datos_abiertos_tbl(
  datos_abiertos_zip_paths,
  unzip_command = Sys.getenv("unzip_command"),
  unzip_args = Sys.getenv("unzip_args"),
  check_unzip_install = TRUE,
  quiet = FALSE,
  clear_zip = FALSE
)

unzip_db_diccionario_ssa(
  diccionario_zip_path,
  unzip_args_dict = list(exdir = ".", overwrite = TRUE),
  clear_zip = FALSE
)

parse_db_diccionario_ssa(diccionario_unzipped_path, clear_csv = FALSE)
```

```
parse_db_datos_abiertos_tbl(
  datos_abiertos_unzipped_path,
  read_format = c("duckdb", "tibble"),
  pragma_memory_limit = Sys.getenv("pragma_memory_limit"),
  dbdir = tempfile(fileext = ".duckdb"),
  drv = duckdb::duckdb(),
  colClasses = get_col_class(),
  tblname = "covidmx",
  quiet = TRUE,
  clear_csv = FALSE,
  ...
)

pega_db_datos_abiertos_tbl_y_diccionario(datos_abiertos_tbl, diccionario)
```

Arguments

read_format	(**opcional**) "duckdb" o "tibble" establece el formato de lectura de la base de datos. En la mayoria de los casos "tibble" va a resultar en un error de memoria. La opcion de "duckdb" siempre es mas rapida por lo cual es el default.
tblname	(**opcional**) Nombre de la tabla de duckdb donde guardar los datos por default se llama covidmx. Solo es relevante si estas usando el mismo dbdir para otro proyecto distinto.
pragma_memory_limit	(**opcional**) Limite de memoria para el programa (ver PRAGMAS). Cambialo a que sea mas o menos la mitad de tu RAM. La forma mas sencilla es como una variable ambiental con Sys.setenv('pragma_memory_limit' = '1GB') por ejemplo para un limite de 1 gigabyte.
drv	(**opcional**) Un driver para dbConnect (default duckdb::duckdb())
dbdir	(**opcional**) Direccion donde guardar la base de datos con terminacion .duckdb. Corresponde al argumento de duckdb::dbConnect__duckdb_driver()
colClasses	(**opcional**) Clases de la columna para leer en duckdb::read_csv_duckdb().
sites.covid	(**opcional**) Sitios web con el vinculo a los archivos .zip de l os datos abiertos. Puedes cambiarlo por uno de los historicos, por ejemplo. La estructura es c("nombre" = "url", "nombre2" = "url2"). La ultima verificacion del sitio web default fue el 6 de septiembre del 2022.
download_process	(**opcional**) Metodo para descargar ya sea pins o download.file. Se recomienda pins pues guarda en memoria la fecha de la ultima descarga y analiza si ha pasado mas de un dia desde la descarga. En caso afirmativo verifica si el archivo ha cambiado y si hubo cambios entonces lo descarga.
unzip_command	(**opcional**) Forma de extraer la base de datos de datos abiertos si unzip falla. La forma de llamarla es con system2(unzip_command, args = c(unzip_args, file_download_data)).

`unzip_args` (**opcional**) Argumentos de extraccion de la base de datos de datos abiertos si `unzip` falla. La forma de llamarla es con `system2(unzip_command, args = c(unzip_args, file_download_data))`.

`check_unzip_install`
(**opcional**) Bandera de verificacion para checar si tienes lo necesario para unzippear los datos en el caso de que `unzip` no sirva.

`clear_zip` (**opcional**) Si borrar los archivos `.zip` descargados para el diccionario y los datos abiertos. No se recomienda si estas usando `pins`. Ve la nota para mas informacion.

`clear_csv` (**opcional**) Si borrar los archivos `.csv` que se generan despues de abrir el zip. El default es que si pues en general solo requieres el `duckdb`.

`force_download` (**opcional**) Analiza si cambio el pin y descarga datos nuevos en caso afirmativo aunque haya pasado menos de un dia.

`show_warnings` (**opcional**) si arrojar `warnings`

`datos_abiertos_zip_paths`
(**opcional**) Camino a los datos abiertos si ya los descargaste en `zip`

`datos_abiertos_unzipped_path`
(**opcional**) Camino a los datos abiertos `csv` si ya los descargaste y descomprimiste el archivo `zip` en un `csv`

`datos_abiertos_tbl`
(**opcional**) Camino a un archivo `.duckdb` con los datos formateados

`quiet` (**opcional**) Variable para no mostrar mensajes

`board_url_name` (**opcional**) Establece el nombre del `pins::board_url` para los datos abiertos (si ya usas pins para que no se empalme). Por default se llama `datos_abiertos`

`cache` parametro para el cache de `pins::board_url`

`use_cache_on_failure`
(**opcional**) Booleana. Establece que si no se pueden descargar datos nuevos utilice los que tenga en memoria. Por default es `TRUE`.

`download_file_args`
(**opcional**) Lista de argumentos adicionales para `download.file` de los datos si se elige este metodo para descargar.

`descarga_db_datos_abiertos_tbl_args`
(**opcional**) Lista con argumentos adicionales para el `pins::pin_download` de datos abiertos

`...` Parametros adicionales para `pins::pin_download`

`site.covid.dic` (**opcional**) Sitio desde el cual descarga del diccionario de datos. La ultima verificacion del sitio fue el 6 de septiembre 2022.

`diccionario_zip_path`
(**opcional**) Camino al diccionario si ya losdescargaste en `zip`

`diccionario_unzipped_path`
(**opcional**) Camino al diccionario `csv` si ya lo descargaste y descomprimiste el archivo `zip` en un `csv`

`diccionario` (**opcional**) Lo que resulta de realizar una descarga del diccionario usando `descarga_diccionario`

board_url_name_dict
: (**opcional**) Establece el nombre del pins::board_url para los datos abiertos. Por default se llama diccionario_covid

cache_diccionario
: (**opcional**) Direccion donde guardar el diccionario en memoria usando pins para no tener que volver a descargarlo si nada ha cambiado

download_file_args_dict
: (**opcional**) Lista de argumentos adicionales para download.file del diccionario si se elige este metodo de descarga.

unzip_args_dict
: (**opcional**) Lista de argumentos para usar utils::unzip en el diccionario de datos.

descarga_db_diccionario_ssa_args
: (**opcional**) Lista con argumentos adicionales para el pins::pin_download de datos abiertos

Value

Lista de valores:

- dats - Tabla conectada mediante duckdb::dbConnect__duckdb_driver() (si duckdb) o tibble (si tibble)
- disconnect - Funcion para cerrar la conexion a la base de datos.
- dict - Lista de tibbles con el diccionario de datos para cada variable

See Also

descarga_datos_abiertos() read_datos_abiertos() descarga_datos_red_irag() descarga_datos_variantes_GI

Examples

```
#Estos links deben omitirse en una corrida normal. Se incluyen por ahora como ejemplo
#pero las opciones site.covid.dic y sites.covid deben eliminarse de abajo.
diclink  <- "https://github.com/RodrigoZepeda/covidmx/raw/main/diccionario_datos_covid19.zip"
dlink    <- "https://github.com/RodrigoZepeda/covidmx/raw/main/datos_abiertos_covid19.zip"

#' #En el ejemplo de R por normas de CRAN tenemos que hacerlo así pero en tu
#computadora puedes solo usar descargar datos sin el if else
if (RCurl::url.exists(dlink) & RCurl::url.exists(diclink)){

  # Descarga solo el diccionario no oficial (omite el site.covid.dic para el de DGE)
  diccionario <- descarga_diccionario(show_warnings = FALSE, site.covid.dic = diclink)

  # O bien descarga solo los datos abiertos de ejemplo desde Github
  # omite el dlink (o cámbialo por el vínculo correcto) para descargar los datos de la DGE
  datos_abiertos <- descarga_db(sites.covid = dlink, show_warnings = FALSE)

  # Pegalos en el formato que se necesita para el resto de funciones
  datos_covid <- pega_db_datos_abiertos_tbl_y_diccionario(datos_abiertos, diccionario)
```

```
  # Desconectamos
  datos_covid$disconnect()

  # Tambien puedes descargar paso por paso
  datos_abiertos <- descarga_db_datos_abiertos_tbl(
    sites.covid = dlink,
    show_warnings = FALSE
  ) |> # Descarga
  unzip_db_datos_abiertos_tbl() |> # Unzippea
  parse_db_datos_abiertos_tbl() # Duckdb

  # O bien el diccionario
  diccionario <- descarga_db_diccionario_ssa(site.covid.dic = diclink) |> # Descarga
  unzip_db_diccionario_ssa() |> # Unzippea
  parse_db_diccionario_ssa() # Tibble

  # Si descargaste cada uno por separado necesitas la funcion pega para
  # juntarlos en un unico objeto
  datos_covid <- pega_db_datos_abiertos_tbl_y_diccionario(datos_abiertos, diccionario)
}
```

`estima_rt` *RT: Número efectivo de reproducción*

Description

`estima_rt` Calcula el número efectivo de reproducción por fecha y entidad usando los metodos de `EpiEstim::estimate_R()`. Por default calcula el número efectivo de reproducción para cada estado.

Usage

```
estima_rt(
  datos_covid,
 entidades = c("AGUASCALIENTES", "BAJA CALIFORNIA", "BAJA CALIFORNIA SUR", "CAMPECHE",
   "CHIAPAS", "CHIHUAHUA", "CIUDAD DE MÉXICO", "COAHUILA DE ZARAGOZA", "COLIMA",
    "DURANGO", "GUANAJUATO", "GUERRERO", "HIDALGO", "JALISCO", "MÉXICO",
   "MICHOACÁN DE OCAMPO", "MORELOS", "NAYARIT", "NUEVO LEÓN", "OAXACA", "PUEBLA",
   "QUERÉTARO", "QUINTANA ROO", "SAN LUIS POTOSÍ", "SINALOA", "SONORA", "TABASCO",
   "TAMAULIPAS", "TLAXCALA", "VERACRUZ DE IGNACIO DE LA LLAVE", "YUCATÁN", "ZACATECAS"),
  group_by_entidad = TRUE,
  entidad_tipo = c("Unidad Medica", "Residencia", "Nacimiento"),
  fecha_tipo = c("Sintomas", "Ingreso", "Defuncion"),
  tipo_clasificacion = c("Sospechosos", "Confirmados COVID", "Negativo a COVID",
    "Inválido", "No realizado"),
  tipo_paciente = c("AMBULATORIO", "HOSPITALIZADO", "NO ESPECIFICADO"),
  list_name = "estima_rt",
```

```
  min_date = as.POSIXct("2020-01-01", tz = Sys.timezone(), format = "%Y-%m-%d"),
  max_date = as.POSIXct(Sys.time()),
  method = "parametric_si",
  config = if (requireNamespace("EpiEstim", quietly = TRUE)) {

    EpiEstim::make_config(list(mean_si = 2.5, std_si = 1.6))
  } else {
     NULL
  },
   ...
)
```

Arguments

datos_covid	(**obligatorio**) Lista de tibbles o duckdbs resultante de descarga_datos_abiertos() o read_datos_abiertos()
entidades	(**opcional**) Vector con las entidades de las unidades medicas a analizar. Opciones: AGUASCALIENTES, BAJA CALIFORNIA, BAJA CALIFORNIA SUR, CAMPECHE, CHIAPAS, CHIHUAHUA, CIUDAD DE MEXICO, COAHUILA DE ZARAGOZA , COLIMA, DURANGO, GUANAJUATO, GUERRERO, HIDALGO, JALISCO, MEXICO, MICHOACAN DE OCAMPO, MORELOS,NAYARIT NUEVO LEON, OAXACA ,PUEBLA, QUERETARO,QUINTANA ROO, SAN LUIS POTOSI, SINALOA, SONORA, TABASCO, TAMAULIPAS,TLAXCALA, VERACRUZ DE IGNACIO DE LA LL YUCATAN, ZACATECAS.
group_by_entidad	(**opcional**) TRUE obtiene los casos para cada entidad reportando en cada fecha la entidad y los casos en dicha entidad. FALSE junta las entidades sumando sus casos en una sola observacion por cada fecha.
entidad_tipo	(**opcional**) Indica a que se refiere las entidades seleccionadas. Elige una de las opciones: Unidad Medica (entidad de la unidad medica), Nacimiento (entidad de origen del individuo) o Residencia (entidad donde reside el individuo).
fecha_tipo	(**opcional**) Selecciona si la fecha que se utiliza es la fecha de Ingreso (si aplica), la fecha de Sintomas o la de Defuncion (si aplica). El default es fecha de Sintomas.
tipo_clasificacion	(**opcional**) Vector con el tipo de clasificaciones (por la prueba) a incluir:Sospechosos,Confirmados COVI Negativo a COVID, Inv\u00e1lido, No realizado
tipo_paciente	(**opcional**) Vector con el tipo de pacientes a incluir. Opciones: AMBULATORIO, HOSPITALIZADO, NO ESPECIFICADO. Por default se incluyen todos.
list_name	(**opcional**) Asigna un nombre en la lista de datos a la base generada
min_date	(**opcional**) Mínima fecha a partir de la cual estimar el RT.
max_date	(**opcional**) Máxima fecha a partir de la cual estimar el RT.
method	(**opcional**) Metodo para estimar el RT con EpiEstim::estimate_R(). Por default se recomienda el método paramétrico de intervalo serial parametric_si.
config	(**opcional**) Configuracion para la estimacion del RT usando EpiEstim::make_config(). Por default se utiliza una media del intervalo serial de mean_si = 2.5 y una desviación estandar de std_si = 1.6. Sin embargo, como el intervalo serial depende mucho de la variante se recomienda cambiarlo.

... (**opcional**) Parámetros adicionales para `EpiEstim::estimate_R()`.

Details

Se sugiere establecer una mínima fecha y una máxima fecha con `min_date` y `max_date` para la estimación pues los intervalos seriales de omicron son distintos a los de la variante delta.

Value

Une a la lista de `datos_covid` una nueva entrada de nombre `list_name` (default: `estima_rt`) con una base de datos (`tibble`) con los resultados agregados.

- `estima_rt` - Base de datos generara con los datos agregados (el nombre cambia si se usa `list_name`).
- dict - Diccionario de datos
- dats - Datos originales (conexion a `duckdb` o `tibble`)
- disconnect - Función para desconectarte de `duckdb`
- ... - Cualquier otro elemento que ya existiera en `datos_covid`

See Also

`descarga_datos_abiertos()` `numero_pruebas()` `cfr()` `chr()` `positividad()` `casos()`

Examples

```
# Para el ejemplo usaremos los datos precargados (datosabiertos) pero tu puedes
# correr el ejemplo descargando informacion mas reciente.
datos_covid <- datosabiertos

# Casos a nivel nacional por estado en todos

suppressWarnings(
  datos_covid <- datos_covid |> estima_rt()
)
head(datos_covid$estima_rt)

# Cambios en la fecha de estimacion siguiendo la recomendacion
# y obtenemos todo a nivel nacional
datos_covid <- datos_covid |> estima_rt(
  min_date = as.POSIXct("2021-07-01"),
  max_date = as.POSIXct("2021-09-01"),
  list_name = "rt_min_max",
  group_by_entidad = FALSE
)
head(datos_covid$rt_min_max)

# Casos a nivel nacional en los confirmados
datos_covid <- datos_covid |>
  estima_rt(
    tipo_clasificacion = "Confirmados COVID",
```

```
    group_by_entidad = FALSE,
    list_name = "rt_confirmados"
  )
head(datos_covid$rt_confirmados)

#' # Cambios en los parametros de epiestim
# estos parametros no tienen razon de ser mas alla de mostrar como se cambian
datos_covid <- datos_covid |>
  estima_rt(
    group_by_entidad = FALSE,
    list_name = "config_rt",
    method = "uncertain_si", # Metodo de estimacion
    config = EpiEstim::make_config(
      mean_si = 2.4,
      std_si = 0.3,
      std_mean_si = 0.2,
      min_mean_si = 2,
      max_mean_si = 4,
      std_std_si = 0.1,
      min_std_si = 0.1,
      max_std_si = 1.0
    )
  )
head(datos_covid$config_rt)

# Casos en BC, BCS en los confirmados
datos_covid |>
  estima_rt(
    entidades = c("BAJA CALIFORNIA", "BAJA CALIFORNIA SUR"),
    tipo_clasificacion = "Confirmados COVID",
    group_by_entidad = TRUE,
    list_name = "rt_bc_bcs"
  ) |>
  plot_covid(
    df_name = "rt_bc_bcs", df_date_index = "FECHA_SINTOMAS",
    df_variable = "Mean(R)", df_covariates = "ENTIDAD_FEDERATIVA"
  )


# Finalmente desconectamos
datos_covid$disconnect()
```

---

numero_pruebas *Numero de Pruebas*

Description

numero_pruebas Calcula el numero total de pruebas por fecha agrupando (o sin hacerlo) por covariables. Por default calcula la el numero de pruebas de antigeno y PCR por separado para cada estado.

Usage

```
numero_pruebas(
  datos_covid,
 entidades = c("AGUASCALIENTES", "BAJA CALIFORNIA", "BAJA CALIFORNIA SUR", "CAMPECHE",
   "CHIAPAS", "CHIHUAHUA", "CIUDAD DE MÉXICO", "COAHUILA DE ZARAGOZA", "COLIMA",
    "DURANGO", "GUANAJUATO", "GUERRERO", "HIDALGO", "JALISCO", "MÉXICO",
  "MICHOACÁN DE OCAMPO", "MORELOS", "NAYARIT", "NUEVO LEÓN", "OAXACA", "PUEBLA",
 "QUERÉTARO", "QUINTANA ROO", "SAN LUIS POTOSÍ", "SINALOA", "SONORA", "TABASCO",
  "TAMAULIPAS", "TLAXCALA", "VERACRUZ DE IGNACIO DE LA LLAVE", "YUCATÁN", "ZACATECAS"),
  group_by_entidad = TRUE,
  entidad_tipo = c("Unidad Medica", "Residencia", "Nacimiento"),
  fecha_tipo = c("Sintomas", "Ingreso", "Defuncion"),
  tipo_prueba = c("Antigeno", "PCR"),
  group_by_tipo_prueba = TRUE,
  tipo_paciente = c("AMBULATORIO", "HOSPITALIZADO", "NO ESPECIFICADO"),
  group_by_tipo_paciente = FALSE,
  tipo_uci = c("SI", "NO", "NO APLICA", "SE IGNORA", "NO ESPECIFICADO"),
  group_by_tipo_uci = FALSE,
 tipo_sector = c("CRUZ ROJA", "DIF", "ESTATAL", "IMSS", "IMSS-BIENESTAR", "ISSSTE",
   "MUNICIPAL", "PEMEX", "PRIVADA", "SEDENA", "SEMAR", "SSA", "UNIVERSITARIO",
   "NO ESPECIFICADO"),
  group_by_tipo_sector = FALSE,
  defunciones = FALSE,
  edad_cut = NULL,
  as_tibble = TRUE,
  fill_zeros = as_tibble,
  list_name = "numero_pruebas",
  .grouping_vars = c()
)
```

Arguments

datos_covid	(**obligatorio**) Lista de tibbles o duckdbs resultante de descarga_datos_abiertos() o read_datos_abiertos()
entidades	(**opcional**) Vector con las entidades de las unidades medicas a analizar. Opciones: AGUASCALIENTES, BAJA CALIFORNIA, BAJA CALIFORNIA SUR, CAMPECHE, CHIAPAS, CHIHUAHUA, CIUDAD DE MEXICO, COAHUILA DE ZARAGOZA , COLIMA, DURANGO, GUANAJUATO, GUERRERO, HIDALGO, JALISCO, MEXICO, MICHOACAN DE OCAMPO, MORELOS,NAYARIT NUEVO LEON, OAXACA ,PUEBLA, QUERETARO,QUINTANA ROO, SAN LUIS POTOSI, SINALOA, SONORA, TABASCO, TAMAULIPAS,TLAXCALA, VERACRUZ DE IGNACIO DE LA LL YUCATAN, ZACATECAS.
group_by_entidad	(**opcional**) TRUE obtiene los casos para cada entidad reportando en cada fecha la entidad y los casos en dicha entidad. FALSE junta las entidades sumando sus casos en una sola observacion por cada fecha.
entidad_tipo	(**opcional**) Indica a que se refiere las entidades seleccionadas. Elige una de las opciones: Unidad Medica (entidad de la unidad medica), Nacimiento (entidad de origen del individuo) o Residencia (entidad donde reside el individuo).

fecha_tipo (**opcional**) Selecciona si la fecha que se utiliza es la fecha de Ingreso (si aplica), la fecha de Sintomas o la de Defuncion (si aplica). El default es fecha de Sintomas.

tipo_prueba (**opcional**) Vector con el tipo de pruebas a incluir Antigeno, PCR. Por default se incluyen ambas.

group_by_tipo_prueba
(**opcional**) Booleana determinando si regresa la base con cada entrada agrupada por tipo_prueba. En caso TRUE (cada fecha y entidad reporta separado el los casos de PCR y Antigeno). En caso FALSE se juntan los casos de PCR y Antigeno para devolver un unico numero por fecha.

tipo_paciente (**opcional**) Vector con el tipo de pacientes a incluir. Opciones: AMBULATORIO, HOSPITALIZADO, NO ESPECIFICADO. Por default se incluyen todos.

group_by_tipo_paciente
(**opcional**) Booleana determinando (caso TRUE) si regresa la base con cada entrada agrupada por tipo_paciente (es decir cada fecha se genera un renglon para AMBULATORIO, un renglon para HOSPITALIZADO, etc) o bien si se suman todos los grupos y cada fecha reporta solo la suma de estos (estilo AMBULATORIO + HOSPITALIZADO segun las categorias de tipo_paciente) El default es FALSE.

tipo_uci (**opcional**) Vector con el tipo de valores para Unidad de Cuidado Intensivo (UCI) a incluir: SI,NO,NO APLICA,SE IGNORA,NO ESPECIFICADO. Por default se incluyen todos.

group_by_tipo_uci
(**opcional**) Booleana. El caso TRUE determina si regresa la base con cada fecha teniendo diferentes renglones uno para cada tipo_uci (es decir cada fecha se generan tantos observaciones como grupos de tipo de UCI) o bien en una sola fecha se suman todos los tipos de UCI (FALSE). El default es FALSE.

tipo_sector (**opcional**) Vector con los sectores del sistema de salud a incluir: CRUZ ROJA,DIF,ESTATAL,IMSS,IMSS-BIE MUNICIPAL,PEMEX, PRIVADA,SEDENA,SEMAR,SSA, UNIVERSITARIO,NO ESPECIFICADO.
Por default se incluyen todos.

group_by_tipo_sector
(**opcional**) Booleana determina en el caso de TRUE si regresa la base con cada entrada agrupada por tipo_sector (es decir cada fecha tiene una entrada con los del IMSS, una entrada distinta con los de ISSSTE, etc) o bien en caso de FALSE se devuelve una sola entrada por fecha con la suma IMSS + ISSSTE + etc segun los sectores seleccionados. El default es FALSE.

defunciones (**opcional**) Booleana si incluir sólo defunciones TRUE o a todos FALSE. El default es FALSE.

edad_cut (**opcional**) Vector con secuencia de edades para hacer grupos. Por ejemplo edad_cut = c(0, 10, Inf) arma dos grupos de edad de 0 a 10 y de 10 a infinito o bien edad_cut = c(15, 20) deja sólo los registros entre 15 y 20 años. Por default es NULL y no arma grupos etarios.

as_tibble (**opcional**) Regresar como tibble el resultado. En caso de que as_tibble sea FALSE se devuelve como conexion en duckdb. Se recomienda el default (tibble).

fill_zeros (**opcional**) En caso de que el resultado sea un tibble regresa observaciones para todas las combinaciones de variables incluyendo como 0 aquellas fechas

	cuando no se observaron casos. En caso contrario no se incluyen las filas donde no se observaron casos.
`list_name`	(**opcional**) Asigna un nombre en la lista de datos a la base generada
`.grouping_vars`	(**opcional**) Vector de variables adicionales de agrupacion de los conteos. Por ejemplo si se agrega `.grouping_vars = 'DIABETES'` entonces para cada fecha habra dos conteos de casos uno de los que tienen diabetes y uno de los que no.

Details

Las pruebas de PCR (polymerase chain reaction) identifican material genetico de un organismo (por ejemplo un virus como el COVID-19 o la influenza). Las pruebas de antigeno (o pruebas rapidas) detectan algunas proteinas que conforman el virus.

Para mas informacion sobre las pruebas y su interpretacion puedes consultar las guias del CDC

Value

Adiciona a la lista de `datos_covid` una nueva entrada de nombre `list_name` (default: `numero_pruebas`) con una base de datos (`tibble` o duckdb) con los resultados agregados.

- numero_pruebas - Base de datos generara con los datos agregados (el nombre cambia si se usa `list_name`).
- dict - Diccionario de datos
- dats - Datos originales (conexion a duckdb o `tibble`)
- disconnect - Función para desconectarte de duckdb
- ... - Cualquier otro elemento que ya existiera en `datos_covid`

Examples

```
# Para el ejemplo usaremos los datos precargados (datosabiertos) pero tu puedes
# correr el ejemplo descargando informacion mas reciente.
datos_covid <- datosabiertos

# Número de pruebas PCR/Antigeno a nivel nacional por estado
datos_covid <- datos_covid |> numero_pruebas()
head(datos_covid$numero_pruebas)

# Número de pruebas nacionales pero sin separar por tipo ni estado
datos_covid <- datos_covid |>
  numero_pruebas(
    group_by_entidad = FALSE, group_by_tipo_prueba = FALSE,
    list_name = "Todas_las_pruebas"
  )
head(datos_covid$Todas_las_pruebas)

# Positivos en Baja California Sur
datos_covid <- datos_covid |>
  numero_pruebas(
    entidades = c("BAJA CALIFORNIA SUR"),
```

```
    list_name = "BCS"
  )
head(datos_covid$BCS)

# Si deseas agrupar por una variable que no este en las opciones asi como tipo paciente
datos_covid <- datos_covid |>
  numero_pruebas(
    tipo_paciente = c("AMBULATORIO", "HOSPITALIZADO"),
    group_by_tipo_paciente = TRUE,
    .grouping_vars = c("DIABETES"),
    list_name = "pruebas_diabetes"
  )
head(datos_covid$pruebas_diabetes)

# Una vez hayas concluido tu trabajo no olvides desconectar
datos_covid$disconnect()
```

`plot_covid` *Grafica los casos de COVID-19*

Description

`plot_covid` Intenta graficar automaticamente la base de datos de covid generados por `casos()`

Usage

```
plot_covid(
  datos_covid,
  df_name = "casos",
  df_date_index = stringr::str_subset(colnames(datos_covid[df_name][[1]]),
    "FECHA|fecha|Fecha"),
  df_variable = NULL,
  df_covariates = c(),
  facet_scale = "free_y",
  facet_ncol = 4,
  date_break_format = "2 months",
  date_labels_format = "%B-%y",
  type = c("point", "line", "spline", "area"),
 plot_theme = ggplot2::theme(panel.background = ggplot2::element_rect(fill = "white"),
    plot.background = ggplot2::element_rect(fill = "white"), axis.text.x =
    ggplot2::element_text(angle = 90, hjust = 1), axis.line.x =
    ggplot2::element_line(color = "black"), legend.position = "none"),
  ...
)
```

Arguments

`datos_covid`	(**obligatorio**) Lista de `tibbles` resultante de `casos()`, `cfr()`, `chr()`, `positividad()` o `rt()`

df_name	(**opcional**) Nombre de la base de datos dentro de la lista datos_covid
df_date_index	(**opcional**) Nombre de la variable que contiene la fecha
df_variable	(**opcional**) Nombre de la variable que se va a graficar en el eje y
df_covariates	(**opcional**) Covariables para el facet_wrap (maximo 2)
facet_scale	(**opcional**) Escala para el ggplot2::facet_wrap()
facet_ncol	(**opcional**) Numero de columnas para el ggplot2::facet_wrap()
date_break_format	(**opcional**) Breaks para el eje x ggplot2::scale_x_date()
date_labels_format	(**opcional**) Formato de fecha para el eje x ggplot2::scale_x_date()
type	(**opcional**) Tipo de grafica (line, area, spline o point)
plot_theme	(**opcional**) Tema para el ggplot2 (ejemplo ggplot2::theme_classic()).
...	(**opcional**) Parametros adicionales para ggformula::geom_spline() en caso de elegir type="spline"

Value

Un ggplot2 con la imagen graficada.

See Also

casos()

Examples

```
# Para el ejemplo usaremos los datos precargados (datosabiertos) pero tu puedes
# correr el ejemplo descargando informacion mas reciente:
datos_covid <- datosabiertos

# Aqui muchos aparecen en cero si usas el default de datosabiertos
# porque la base de datosabiertos tiene muy pocos casos
datos_covid |>
  casos(list_name = "casos_for_plot", group_by_entidad = FALSE) |>
  plot_covid(df_name = "casos_for_plot")

# Grafica de casos nacional

datos_covid |>
  casos(group_by_entidad = FALSE, list_name = "plot_nal") |>
  plot_covid(df_name = "plot_nal")

# Ajuste mediante splines
datos_covid |>
  casos(group_by_entidad = FALSE, list_name = "spline_nacional") |>
  plot_covid(df_name = "spline_nacional", type = "spline", spar = 0.5)

# Graficacion por covariables
```

```
# el objeto devuelto es un objeto de ggplot2 al que se le puede dar formato
if (!requireNamespace("ggplot2", quietly = TRUE)) {
  datos_covid |>
    chr(
      group_by_entidad = TRUE, list_name = "plot_nal", .grouping_vars = c("SEXO"),
      entidades = c("BAJA CALIFORNIA", "BAJA CALIFORNIA SUR")
    ) |>
    plot_covid(
      df_name = "plot_nal",
      date_break_format = "1 week",
      date_labels_format = "%d/%B/%Y",
      df_covariates = c("SEXO", "ENTIDAD_FEDERATIVA"),
      type = "area"
    ) +
    ggplot2::ggtitle("Plot nacional")
}

# Puedes tambien primero editar el tibble que usaras por ejemplo poniendo
# los nombres de los sexos
datos_covid <- datos_covid |>
  chr(
    group_by_entidad = TRUE, list_name = "plot_nal", .grouping_vars = c("SEXO"),
    entidades = c("BAJA CALIFORNIA", "BAJA CALIFORNIA SUR")
  )


# Finalmente desconectamos
datos_covid$disconnect()
```

---

positividad *Positividad*

Description

positividad Calcula la positividad por fecha agrupando (o sin hacerlo) por covariables. Por default calcula la positividad de las pruebas haciendo Antigeno y PCR por separado, cada una por fecha y entidad.

Usage

```
positividad(
  datos_covid,
 entidades = c("AGUASCALIENTES", "BAJA CALIFORNIA", "BAJA CALIFORNIA SUR", "CAMPECHE",
   "CHIAPAS", "CHIHUAHUA", "CIUDAD DE MÉXICO", "COAHUILA DE ZARAGOZA", "COLIMA",
    "DURANGO", "GUANAJUATO", "GUERRERO", "HIDALGO", "JALISCO", "MÉXICO",
  "MICHOACÁN DE OCAMPO", "MORELOS", "NAYARIT", "NUEVO LEÓN", "OAXACA", "PUEBLA",
  "QUERÉTARO", "QUINTANA ROO", "SAN LUIS POTOSÍ", "SINALOA", "SONORA", "TABASCO",
  "TAMAULIPAS", "TLAXCALA", "VERACRUZ DE IGNACIO DE LA LLAVE", "YUCATÁN", "ZACATECAS"),
```

```
  group_by_entidad = TRUE,
  entidad_tipo = c("Unidad Medica", "Residencia", "Nacimiento"),
  fecha_tipo = c("Sintomas", "Ingreso", "Defuncion"),
  tipo_prueba = c("Antigeno", "PCR"),
  group_by_tipo_prueba = TRUE,
  tipo_paciente = c("AMBULATORIO", "HOSPITALIZADO", "NO ESPECIFICADO"),
  group_by_tipo_paciente = FALSE,
  tipo_uci = c("SI", "NO", "NO APLICA", "SE IGNORA", "NO ESPECIFICADO"),
  group_by_tipo_uci = FALSE,
 tipo_sector = c("CRUZ ROJA", "DIF", "ESTATAL", "IMSS", "IMSS-BIENESTAR", "ISSSTE",
    "MUNICIPAL", "PEMEX", "PRIVADA", "SEDENA", "SEMAR", "SSA", "UNIVERSITARIO",
    "NO ESPECIFICADO"),
  group_by_tipo_sector = FALSE,
  defunciones = FALSE,
  edad_cut = NULL,
  fill_NA = TRUE,
  list_name = "positividad",
  remove_inconclusive = TRUE,
  .grouping_vars = c()
)
```

Arguments

datos_covid	(**obligatorio**) Lista de tibbles o duckdbs resultante de descarga_datos_abiertos() o read_datos_abiertos()
entidades	(**opcional**) Vector con las entidades de las unidades medicas a analizar. Opciones: AGUASCALIENTES, BAJA CALIFORNIA, BAJA CALIFORNIA SUR, CAMPECHE, CHIAPAS, CHIHUAHUA, CIUDAD DE MEXICO, COAHUILA DE ZARAGOZA , COLIMA, DURANGO, GUANAJUATO, GUERRERO, HIDALGO, JALISCO, MEXICO, MICHOACAN DE OCAMPO, MORELOS,NAYARIT NUEVO LEON, OAXACA ,PUEBLA, QUERETARO,QUINTANA ROO, SAN LUIS POTOSI, SINALOA, SONORA, TABASCO, TAMAULIPAS,TLAXCALA, VERACRUZ DE IGNACIO DE LA LL YUCATAN, ZACATECAS.
group_by_entidad	(**opcional**) TRUE obtiene los casos para cada entidad reportando en cada fecha la entidad y los casos en dicha entidad. FALSE junta las entidades sumando sus casos en una sola observacion por cada fecha.
entidad_tipo	(**opcional**) Indica a que se refiere las entidades seleccionadas. Elige una de las opciones: Unidad Medica (entidad de la unidad medica), Nacimiento (entidad de origen del individuo) o Residencia (entidad donde reside el individuo).
fecha_tipo	(**opcional**) Selecciona si la fecha que se utiliza es la fecha de Ingreso (si aplica), la fecha de Sintomas o la de Defuncion (si aplica). El default es fecha de Sintomas.
tipo_prueba	(**opcional**) Vector con el tipo de pruebas a incluir Antigeno, PCR. Por default se incluyen ambas.
group_by_tipo_prueba	(**opcional**) Booleana determinando si regresa la base con cada entrada agrupada por tipo_prueba. En caso TRUE (cada fecha y entidad reporta separado

el los casos de PCR y Antigeno). En caso FALSE se juntan los casos de PCR y Antigeno para devolver un unico numero por fecha.

tipo_paciente (**opcional**) Vector con el tipo de pacientes a incluir. Opciones: AMBULATORIO, HOSPITALIZADO, NO ESPECIFICADO. Por default se incluyen todos.

group_by_tipo_paciente
(**opcional**) Booleana determinando (caso TRUE) si regresa la base con cada entrada agrupada por tipo_paciente (es decir cada fecha se genera un renglon para AMBULATORIO, un renglon para HOSPITALIZADO, etc) o bien si se suman todos los grupos y cada fecha reporta solo la suma de estos (estilo AMBULATORIO + HOSPITALIZADO segun las categorias de tipo_paciente) El default es FALSE.

tipo_uci (**opcional**) Vector con el tipo de valores para Unidad de Cuidado Intensivo (UCI) a incluir: SI,NO,NO APLICA,SE IGNORA,NO ESPECIFICADO. Por default se incluyen todos.

group_by_tipo_uci
(**opcional**) Booleana. El caso TRUE determina si regresa la base con cada fecha teniendo diferentes renglones uno para cada tipo_uci (es decir cada fecha se generan tantos observaciones como grupos de tipo de UCI) o bien en una sola fecha se suman todos los tipos de UCI (FALSE). El default es FALSE.

tipo_sector (**opcional**) Vector con los sectores del sistema de salud a incluir: CRUZ ROJA,DIF,ESTATAL,IMSS,IMSS-BIE MUNICIPAL,PEMEX, PRIVADA,SEDENA,SEMAR,SSA, UNIVERSITARIO,NO ESPECIFICADO.
Por default se incluyen todos.

group_by_tipo_sector
(**opcional**) Booleana determina en el caso de TRUE si regresa la base con cada entrada agrupada por tipo_sector (es decir cada fecha tiene una entrada con los del IMSS, una entrada distinta con los de ISSSTE, etc) o bien en caso de FALSE se devuelve una sola entrada por fecha con la suma IMSS + ISSSTE + etc segun los sectores seleccionados. El default es FALSE.

defunciones (**opcional**) Booleana si incluir sólo defunciones TRUE o a todos FALSE. El default es FALSE.

edad_cut (**opcional**) Vector con secuencia de edades para hacer grupos. Por ejemplo edad_cut = c(0, 10, Inf) arma dos grupos de edad de 0 a 10 y de 10 a infinito o bien edad_cut = c(15, 20) deja sólo los registros entre 15 y 20 años. Por default es NULL y no arma grupos etarios.

fill_NA (**opcional**) Regresa observaciones para todas las combinaciones de variables incluyendo como NA donde no se observaron casos en el denominador. En caso contrario no se incluyen las filas donde no se observaron casos.

list_name (**opcional**) Asigna un nombre en la lista de datos a la base generada

remove_inconclusive
(**opcional**) Si TRUE no considera en el denominador de la positividad las pruebas cuyo resultado es inconcluso o aún no ha sido otorgado. Si FALSE considera a todos. Por default es TRUE.

.grouping_vars (**opcional**) Vector de variables adicionales de agrupacion de los conteos. Por ejemplo si se agrega .grouping_vars = 'DIABETES' entonces para cada fecha habra dos conteos de casos uno de los que tienen diabetes y uno de los que no.

Details

La positividad se define como

$$\frac{\#Pruebaspositivas}{Totaldepruebas}$$

Si se utiliza la opción `remove_inconclusive = TRUE` el **Total de pruebas** se calcula utilizando solo `POSITIVOS + NEGATIVOS`. Si `remove_inconclusive = FALSE` se calcula utilizando todas las personas que tuvieron prueba: `POSITIVOS + NEGATIVOS + INCONCLUSOS + SIN RESULTADO`.

Si no se realizaron pruebas un dia la positividad no esta definida pues el **Total de pruebas** es cero. En ese caso si `fill_NA = TRUE` se devuelven las entradas de esos dias pero con valor `NA`.

Value

Une a la lista de `datos_covid` una nueva entrada de nombre `list_name` (default: `positividad`) con una base de datos (`tibble`) con los resultados agregados.

- positividad - Base de datos generara con los datos agregados (el nombre cambia si se usa `list_name`).
- dict - Diccionario de datos
- dats - Datos originales (conexion a `duckdb` o `tibble`)
- disconnect - Función para desconectarte de `duckdb`
- ... - Cualquier otro elemento que ya existiera en `datos_covid`

References

Furuse, Y., Ko, Y. K., Ninomiya, K., Suzuki, M., & Oshitani, H. (2021). Relationship of test positivity rates with COVID-19 epidemic dynamics. International journal of environmental research and public health, 18(9), 4655.

Al Dallal, A., AlDallal, U., & Al Dallal, J. (2021). Positivity rate: an indicator for the spread of COVID-19. Current Medical Research and Opinion, 37(12), 2067-2076.

See Also

`descarga_datos_abiertos()` `numero_pruebas()` `cfr()` `chr()` `estima_rt()` `casos()`

Examples

```
# Para el ejemplo usaremos los datos precargados (datosabiertos) pero tu puedes
# correr el ejemplo descargando informacion mas reciente.
datos_covid <- datosabiertos

# Casos a nivel nacional por estado por tipo de prueba
datos_covid <- datos_covid |> positividad()
head(datos_covid$positividad)


# Total nacional sumando todas las pruebas del pais
datos_covid <- datos_covid |>
```

```
  positividad(group_by_entidad = FALSE, list_name = "positividad_nacional")
head(datos_covid$positividad_nacional)

# Positivos en Baja California y Baja California Sur
datos_covid <- datos_covid |>
  positividad(
    entidades = c("BAJA CALIFORNIA", "BAJA CALIFORNIA SUR"),
    list_name = "positividad_californiana"
  )
head(datos_covid$positividad_californiana)

# Agrupando ambas pruebas en una sola positividad global
datos_covid <- datos_covid |>
  positividad(
    entidades = c("BAJA CALIFORNIA", "BAJA CALIFORNIA SUR"),
    group_by_tipo_prueba = FALSE,
    list_name = "positividad_californiana_2"
  )
head(datos_covid$positividad_californiana_2)

# Regresa la suma de ambos estados pero dividiendo por tipo de paciente
datos_covid <- datos_covid |>
  positividad(
    entidades = c("BAJA CALIFORNIA", "BAJA CALIFORNIA SUR"),
    group_by_entidad = FALSE,
    tipo_paciente = c("AMBULATORIO", "HOSPITALIZADO"),
    group_by_tipo_paciente = TRUE,
    list_name = "positividad_paciente"
  )
head(datos_covid$positividad_paciente)

# Si deseas agrupar por una variable que no este en las opciones va en .grouping_vars
datos_covid <- datos_covid |>
  positividad(
    tipo_sector = "IMSS",
    .grouping_vars = c("SEXO"),
    list_name = "positividad_imss_sexo"
  )
head(datos_covid$positividad_imss_sexo)


# Una vez hayas concluido tu trabajo no olvides desconectar
datos_covid$disconnect()
```

---

read_datos_abiertos *Lee la base de datos de la direccion general de epidemiologia que ya descargaste*

Description

`read_datos_abiertos` Lee los datos abiertos almacenados en tu base de duckdb que bajaste con `descarga_datos_abiertos`. Intenta de manera automática determinar si los lee de duckdb, `csv` ó `zip`

Usage

```
read_datos_abiertos(
  datos_abiertos_path = NULL,
  dbdir = tempfile(fileext = ".duckdb"),
  tblname = "covidmx",
  pragma_memory_limit = Sys.getenv("pragma_memory_limit"),
  drv = duckdb::duckdb(),
  colClasses = get_col_class(),
  read_format = c("duckdb", "tibble"),
  ...
)
```

Arguments

`datos_abiertos_path`	(**obligatorio**) Camino a los datos abiertos si son un `zip`, un `csv` o un `.duckdb`
`dbdir`	(**opcional**) Direccion donde guardar la base de datos con terminacion `.duckdb`. Corresponde al argumento de `duckdb::dbConnect__duckdb_driver()`
`tblname`	(**opcional**) Nombre de la tabla de `duckdb` donde guardar los datos por default se llama `covidmx`. Solo es relevante si estas usando el mismo `dbdir` para otro proyecto distinto.
`pragma_memory_limit`	(**opcional**) Limite de memoria para el programa (ver PRAGMAS). Cambialo a que sea mas o menos la mitad de tu RAM. La forma mas sencilla es como una variable ambiental con `Sys.setenv('pragma_memory_limit' = '1GB')` por ejemplo para un limite de 1 gigabyte.
`drv`	(**opcional**) Un driver para `dbConnect` (default `duckdb::duckdb()`)
`colClasses`	(**opcional**) Clases de la columna para leer en `duckdb::read_csv_duckdb()`.
`read_format`	(**opcional**) `"duckdb"` o `"tibble"` establece el formato de lectura de la base de datos. En la mayoria de los casos `"tibble"` va a resultar en un error de memoria. La opcion de `"duckdb"` siempre es mas rapida por lo cual es el default.
`...`	(**opcional**) parametros adicionales para `descarga_datos_abiertos()`

Value

Lista de valores:

- dats - Tabla conectada mediante `DBI::dbConnect` (si `duckdb`) o tibble (si `tibble`)
- disconnect - Funcion para cerrar la conexion a la base de datos.
- dict - Lista de `tibbles` con el diccionario de datos para cada variable

Note

Para guardar tu base con duckdb cambia el `dbdir` a un archivo `.duckdb`. Como ejemplo `dbdir = "ejemplo.duckdb"`.

See Also

`descarga_datos_abiertos()` `descarga_datos_red_irag()` `descarga_datos_variantes_GISAID()` `casos()`

Examples

```
#Archivo temporal donde guardar las cosas es cualquier .duckdb
file_duck <- tempfile(fileext = ".duckdb")

#Estos links deben omitirse en una corrida normal. Se incluyen por ahora como ejemplo
#pero las opciones site.covid.dic y sites.covid deben eliminarse de abajo.
dlink  <- "https://github.com/RodrigoZepeda/covidmx/raw/main/datos_abiertos_covid19.zip"
diclink <- "https://github.com/RodrigoZepeda/covidmx/raw/main/diccionario_datos_covid19.zip"

if (RCurl::url.exists(dlink) & RCurl::url.exists(diclink)){
  # EJEMPLO 0: Descarga los datos abiertos en archivo file_duck
  descarga_datos_abiertos(dbdir = file_duck, sites.covid = dlink, show_warnings = FALSE,
      site.covid.dic = diclink)$disconnect()

  # EJEMPLO 1: Lee los datos de duckdb una vez descargados
  datos_covid <- read_datos_abiertos(file_duck, show_warnings = FALSE,
                                     site.covid.dic = diclink) # Lee duckdb
  datos_covid$disconnect()

# EJEMPLO 2: Lee los datos desde un zip descargado
# Descarga archivos de la DGE y guarda el zip
direccion_zip <- descarga_db_datos_abiertos_tbl(sites.covid = dlink, show_warnings = FALSE)
# Lee zip
datos_covid <- read_datos_abiertos(direccion_zip, dbdir = file_duck, show_warnings = FALSE,
                    site.covid.dic = diclink)
datos_covid$disconnect()

# EJEMPLO 3: Lee los datos desde un zip descargado
# Descarga archivos zip de la DGE
direccion_zip <- descarga_db_datos_abiertos_tbl(sites.covid = dlink, show_warnings = FALSE)
direccion_csv <- unzip_db_datos_abiertos_tbl(direccion_zip) # Descomprime el zip para tener csv
# Lee los csv
datos_covid <- read_datos_abiertos(direccion_csv, dbdir = file_duck, show_warnings = FALSE,
                                   site.covid.dic = diclink)
datos_covid$disconnect()

# EJEMPLO 4: Si ya tenias el diccionario lo puedes agregar
# Simula la idea de ya tener el diccionario
diccionario <- descarga_diccionario(show_warnings = FALSE, site.covid.dic = diclink)
datos_covid <- read_datos_abiertos(file_duck, diccionario = diccionario, show_warnings = FALSE)
datos_covid$disconnect()
```

```
# EJEMPLO 5: Si ya tenias el diccionario como archivo zip
# Descarga el diccionario para tenerlo como zip
diccionario_zip <- descarga_db_diccionario_ssa(show_warnings = FALSE, site.covid.dic = diclink)
datos_covid <- read_datos_abiertos(file_duck, diccionario_zip_path = diccionario_zip,
                                   show_warnings = FALSE)
datos_covid$disconnect()

# EJEMPLO 6: Si ya tenias el diccionario como archivo xlsx
# Descarga el diccionario para tenerlo como zip
diccionario_zip <- descarga_db_diccionario_ssa(show_warnings = FALSE, site.covid.dic = diclink)
# Abre el csv del diccionario
diccionario_csv <- unzip_db_diccionario_ssa(diccionario_zip)
datos_covid <- read_datos_abiertos(file_duck,
  diccionario_unzipped_path = diccionario_csv,
  show_warnings = FALSE
)
datos_covid$disconnect()
}
```

`read_datos_abiertos_zip` *Auxiliares de lectura para la base de la Direccion General de Epidemiologia*

Description

La funcion principal es `read_datos_abiertos()` la cual decide si los lee de `zip`, `duckdb` o `csv` Tambien puedes usar las auxiliares respectivas

- `read_datos_abiertos_zip()` Si sólo descargaste los datos de la DGE en `.zip`
- `read_datos_abiertos_csv()` Si descargaste los datos de la DGE en `.zip` y los descomprimiste.
- `read_datos_abiertos_duckdb()` Si ya creaste tu table en duckdb

Usage

```
read_datos_abiertos_zip(
  datos_abiertos_zip_paths,
  diccionario_zip_path = NULL,
  diccionario_unzipped_path = NULL,
  diccionario = NULL,
  read_format = c("duckdb", "tibble"),
  tblname = "covidmx",
  drv = duckdb::duckdb(),
  dbdir = tempfile(fileext = ".duckdb"),
  colClasses = get_col_class(),
```

```
  download_process = c("pins", "download.file"),
  site.covid.dic = paste0("http://datosabiertos.salud.", "gob.mx/gobmx/salud/datos_a",
    "biertos/diccionario_datos_", "covid19.zip"),
  unzip_command = Sys.getenv("unzip_command"),
  unzip_args = Sys.getenv("unzip_args"),
  unzip_args_dict = list(exdir = ".", overwrite = TRUE),
  check_unzip_install = TRUE,
  clear_zip = (download_process[1] != "pins"),
  clear_csv = TRUE,
  use_dict = TRUE,
  quiet = FALSE,
  cache_datos = NULL,
  use_cache_on_failure = TRUE,
  cache_diccionario = NULL,
  force_download = FALSE,
  show_warnings = TRUE,
  board_url_name = "datos_abiertos",
  board_url_name_dict = "diccionario_covid",
  download_file_args = list(method = "curl", destfile = tempfile(), quiet = quiet),
  descarga_db_diccionario_ssa_args = list(),
  ...
)

read_datos_abiertos_csv(
  datos_abiertos_unzipped_path,
  diccionario_zip_path = NULL,
  diccionario_unzipped_path = NULL,
  diccionario = NULL,
  read_format = c("duckdb", "tibble"),
  tblname = "covidmx",
  drv = duckdb::duckdb(),
  dbdir = tempfile(fileext = ".duckdb"),
  colClasses = get_col_class(),
  download_process = c("pins", "download.file"),
  site.covid.dic = paste0("http://datosabiertos.salud.", "gob.mx/gobmx/salud/datos_a",
    "biertos/diccionario_datos_", "covid19.zip"),
  unzip_args_dict = list(exdir = ".", overwrite = TRUE),
  clear_csv = TRUE,
  quiet = FALSE,
  use_cache_on_failure = TRUE,
  cache_diccionario = NULL,
  force_download = FALSE,
  show_warnings = TRUE,
  board_url_name_dict = "diccionario_covid",
  download_file_args = list(method = "curl", destfile = tempfile(), quiet = quiet),
  descarga_db_diccionario_ssa_args = list(),
  ...
)
```

```
read_datos_abiertos_duckdb(
  datos_abiertos_tbl,
  drv = duckdb::duckdb(),
  tblname = "covidmx",
  pragma_memory_limit = Sys.getenv("pragma_memory_limit"),
  diccionario_zip_path = NULL,
  diccionario_unzipped_path = NULL,
  diccionario = NULL,
  download_process = c("pins", "download.file"),
 site.covid.dic = paste0("http://datosabiertos.salud.", "gob.mx/gobmx/salud/datos_a",
    "biertos/diccionario_datos_", "covid19.zip"),
  unzip_args_dict = list(exdir = ".", overwrite = TRUE),
  clear_zip = download_process[1] != "pins",
  clear_csv = TRUE,
  use_dict = TRUE,
  quiet = FALSE,
  use_cache_on_failure = TRUE,
  cache_diccionario = NULL,
  force_download = FALSE,
  show_warnings = TRUE,
  board_url_name_dict = "diccionario_covid",
 download_file_args = list(method = "curl", destfile = tempfile(), quiet = quiet),
  descarga_db_diccionario_ssa_args = list(),
  ...
)
```

Arguments

datos_abiertos_zip_paths	(**opcional**) Camino a los datos abiertos si ya los descargaste en zip
diccionario_zip_path	(**opcional**) Camino al diccionario si ya losdescargaste en zip
diccionario_unzipped_path	(**opcional**) Camino al diccionario csv si ya lo descargaste y descomprimiste el archivo zip en un csv
diccionario	(**opcional**) Lo que resulta de realizar una descarga del diccionario usando descarga_diccionario
read_format	(**opcional**) "duckdb" o "tibble" establece el formato de lectura de la base de datos. En la mayoria de los casos "tibble" va a resultar en un error de memoria. La opcion de "duckdb" siempre es mas rapida por lo cual es el default.
tblname	(**opcional**) Nombre de la tabla de duckdb donde guardar los datos por default se llama covidmx. Solo es relevante si estas usando el mismo dbdir para otro proyecto distinto.
drv	(**opcional**) Un driver para dbConnect (default duckdb::duckdb())
dbdir	(**opcional**) Direccion donde guardar la base de datos con terminacion .duckdb. Corresponde al argumento de duckdb::dbConnect__duckdb_driver()
colClasses	(**opcional**) Clases de la columna para leer en duckdb::read_csv_duckdb().

download_process
: (**opcional**) Metodo para descargar ya sea pins o download.file. Se recomienda pins pues guarda en memoria la fecha de la ultima descarga y analiza si ha pasado mas de un dia desde la descarga. En caso afirmativo verifica si el archivo ha cambiado y si hubo cambios entonces lo descarga.

site.covid.dic
: (**opcional**) Sitio desde el cual descarga del diccionario de datos. La ultima verificacion del sitio fue el 6 de septiembre 2022.

unzip_command
: (**opcional**) Forma de extraer la base de datos de datos abiertos si unzip falla. La forma de llamarla es con system2(unzip_command, args = c(unzip_args, file_download_data)).

unzip_args
: (**opcional**) Argumentos de extraccion de la base de datos de datos abiertos si unzip falla. La forma de llamarla es con system2(unzip_command, args = c(unzip_args, file_download_data)).

unzip_args_dict
: (**opcional**) Lista de argumentos para usar utils::unzip en el diccionario de datos.

check_unzip_install
: (**opcional**) Bandera de verificacion para checar si tienes lo necesario para unzippear los datos en el caso de que unzip no sirva.

clear_zip
: (**opcional**) Si borrar los archivos .zip descargados para el diccionario y los datos abiertos. No se recomienda si estas usando pins. Ve la nota para mas informacion.

clear_csv
: (**opcional**) Si borrar los archivos .csv que se generan despues de abrir el zip. El default es que si pues en general solo requieres el duckdb.

use_dict
: (**opcional**) Si descargar el diccionario de site.covid.dic.

quiet
: (**opcional**) Variable para no mostrar mensajes

cache_datos
: (**opcional**) Direccion donde guardar los datos en memoria usando pins para no tener que volver a descargarlos si nada ha cambiado

use_cache_on_failure
: (**opcional**) Booleana. Establece que si no se pueden descargar datos nuevos utilice los que tenga en memoria. Por default es TRUE.

cache_diccionario
: (**opcional**) Direccion donde guardar el diccionario en memoria usando pins para no tener que volver a descargarlo si nada ha cambiado

force_download
: (**opcional**) Analiza si cambio el pin y descarga datos nuevos en caso afirmativo aunque haya pasado menos de un dia.

show_warnings
: (**opcional**) si arrojar warnings

board_url_name
: (**opcional**) Establece el nombre del pins::board_url para los datos abiertos (si ya usas pins para que no se empalme). Por default se llama datos_abiertos

board_url_name_dict
: (**opcional**) Establece el nombre del pins::board_url para los datos abiertos. Por default se llama diccionario_covid

download_file_args
: (**opcional**) Lista de argumentos adicionales para download.file de los datos si se elige este metodo para descargar.

descarga_db_diccionario_ssa_args
: (**opcional**) Lista con argumentos adicionales para el `pins::pin_download` de datos abiertos

`...`
: (**opcional**) Parametros adicionales para `DBI::dbConnect`.

datos_abiertos_unzipped_path
: (**opcional**) Camino a los datos abiertos `csv` si ya los descargaste y descomprimiste el archivo `zip` en un `csv`

datos_abiertos_tbl
: (**opcional**) Camino a un archivo `.duckdb` con los datos formateados

pragma_memory_limit
: (**opcional**) Limite de memoria para el programa (ver PRAGMAS). Cambialo a que sea mas o menos la mitad de tu RAM. La forma mas sencilla es como una variable ambiental con `Sys.setenv('pragma_memory_limit' = '1GB')` por ejemplo para un limite de 1 gigabyte.

Value

Lista de valores:

- dats - Tabla conectada mediante `DBI::dbConnect` (si `duckdb`) o tibble (si `tibble`)
- disconnect - Funcion para cerrar la conexion a la base de datos.
- dict - Lista de `tibbles` con el diccionario de datos para cada variable

Note

Para guardar tu base con `duckdb` cambia el `dbdir` a un archivo `.duckdb`. Como ejemplo `dbdir = "ejemplo.duckdb"`.

Examples

```
# Lee los datos de duckdb una vez descargados
# quita la opción de sites.covid para descargar los de la DGE.
# Esto es solo un ejemplo.
file_ejemplo <- tempfile(fileext = ".duckdb")

#Estos links deben omitirse en una corrida normal. Se incluyen por ahora como ejemplo
#pero las opciones site.covid.dic y sites.covid deben eliminarse de abajo.
dlink     <- "https://github.com/RodrigoZepeda/covidmx/raw/main/datos_abiertos_covid19.zip"
diclink    <- "https://github.com/RodrigoZepeda/covidmx/raw/main/diccionario_datos_covid19.zip"

#En el ejemplo de R por normas de CRAN tenemos que hacerlo así pero en tu
#computadora puedes solo usar descargar datos sin el if else
if (RCurl::url.exists(dlink) & RCurl::url.exists(diclink)){

  #Necesitamos la base para verificar los reads
  datos_covid <- descarga_datos_abiertos(
    dbdir = file_ejemplo,
    sites.covid = dlink,
    site.covid.dic = diclink,
```

```
  show_warnings = FALSE
)
datos_covid$disconnect()

datos_covid <- read_datos_abiertos(file_ejemplo, show_warnings = FALSE,
                  site.covid.dic = diclink)
datos_covid$disconnect()

 # Es lo mismo que:
datos_covid <- read_datos_abiertos_duckdb(file_ejemplo, show_warnings = FALSE,
                                          site.covid.dic = diclink)
datos_covid$disconnect()

# Descarga los datos y lee de un zip guardandolos a la vez en
# base de nombre datos_desde_zip.duckdb
direccion_zip <- descarga_db_datos_abiertos_tbl(sites.covid = dlink, show_warnings = FALSE,
                                                site.covid.dic = diclink)

datos_covid <- read_datos_abiertos(direccion_zip,
  dbdir = file_ejemplo,
  site.covid.dic = diclink,
  show_warnings = FALSE
)
datos_covid$disconnect()

# Es lo mismo que:
datos_covid <- read_datos_abiertos_zip(direccion_zip,
  site.covid.dic = diclink,
  dbdir = file_ejemplo,
  show_warnings = FALSE
)
datos_covid$disconnect()

# Descarga los datos y lee de un csv
direccion_zip <- descarga_db_datos_abiertos_tbl(sites.covid = dlink, show_warnings = FALSE)
direccion_csv <- unzip_db_datos_abiertos_tbl(direccion_zip)
datos_covid   <- read_datos_abiertos(direccion_csv, show_warnings = FALSE,
                                     site.covid.dic = diclink)
datos_covid$disconnect()

# Es lo mismo que:
direccion_csv <- unzip_db_datos_abiertos_tbl(direccion_zip)
datos_covid   <- read_datos_abiertos_csv(direccion_csv, show_warnings = FALSE,
                         site.covid.dic = diclink)
datos_covid$disconnect()
}
```

---

update_covidmx *Actualiza el paquete covidmx*

Description

Descarga e instala la version mas reciente de covidmx desde Github https://github.com/RodrigoZepeda/covidmx

Usage

```
update_covidmx(quiet = FALSE, force = FALSE, ...)
```

Arguments

quiet	(**opcional**) Determina si instalar en silencio
force	(**opcional**) Determina si forzar la reinstalacion
...	(**opcional**) Parametros adicionales para remotes::install_github()

Value

Ningún valor. Se llama para actualizar el paquete a la versión más reciente desde Github.

Note

Actualiza el paquete instalando todas las dependencias necesarias.

Examples

```
## Not run:
# Actualiza el paquete de coivdmx de Github | Updates the covidmx package from Github
update_covidmx()

## End(Not run)
```

Index