# Package: territorial (via r-universe)

November 4, 2026

**Title** Funciones para trabajar con datos comunales y regionales de Chile

**Version** 0.9.91

**Description** Este paquete busca facilitar tareas de limpieza que suelen ser necesarias al trabajar con datos de Chile a nivel comunal, por ejemplo: limpiar los nombres de las comunas y regiones, agregar códigos comunales y regionales, ordenar las regiones del país, entre otras.

**License** GPL (>= 3)

**URL** https://bastianolea.github.io/territorial, https://github.com/bastianolea/territorial

**BugReports** https://github.com/bastianolea/territorial/issues

**Depends** R (>= 3.5)

**Imports** cli, dplyr, forcats, glue, purrr, rlang, stringi, stringr

**Suggests** knitr, quarto, readxl, rmarkdown, testthat (>= 3.0.0), tidyr, here, janitor, ggplot2, colorspace, patchwork, DT, messy

**VignetteBuilder** knitr, quarto

**Config/Needs/website** quarto, rmarkdown

**Config/testthat/edition** 3

**Encoding** UTF-8

**LazyData** true

**Roxygen** list(markdown = TRUE)

**Config/roxygen2/version** 8.1.1

**Config/pak/sysreqs** libicu-dev

**Repository** https://bastianolea.r-universe.dev

**Date/Publication** 2026-11-04 16:09:06 UTC

**RemoteUrl** https://github.com/bastianolea/territorial

**RemoteRef** HEAD

**RemoteSha** 206f34e28f9c0e5c4d0a85d6b8122a743a2687d6

# Contents

abreviar_comunas . . . 3
acortar_regiones . . . 4
actualizar_codigo_comuna . . . 5
agregar_clasificacion . . . 6
agregar_macrozona . . . 7
agregar_orden_region . . . 8
as_capital_regional . . . 9
as_codigo_comuna . . . 10
as_codigo_region . . . 11
as_nombre_comuna . . . 11
as_nombre_region . . . 12
buscar_comuna . . . 13
capitales_regionales . . . 14
clasificacion . . . 15
codigos_comunas . . . 16
comunas . . . 16
contar_comunas . . . 17
contextualizar . . . 18
cut_historicos . . . 19
eliminar_texto . . . 20
insertar_texto . . . 20
is_capital_regional . . . 21
is_codigo_comuna . . . 22
is_codigo_comuna_antiguo . . . 23
is_codigo_region . . . 23
is_nombre_comuna . . . 24
is_nombre_region . . . 25
limpiar_comunas . . . 25
limpiar_regiones . . . 27
limpiar_texto . . . 29
localidades . . . 30
obtener_comunas . . . 31
ordenar_regiones . . . 31
preposicion_region . . . 33
redactar_comunas . . . 33
redactar_region . . . 34
reemplazar_texto . . . 35
regiones . . . 35
territorios . . . 36
trivia . . . 37
ubicar_comunas . . . 37
ubicar_localidades . . . 38
validar_comunas . . . 39
validar_regiones . . . 40

**Index** **41**

`abreviar_comunas` *Crear abreviaciones de 3 letras desde nombres comunas*

## Description

A partir de un vector de nombres de comunas (como `comunas()`), retorna nombres de comunas abreviados, como siglas de 3 caracteres (a excepción de Santiago, que queda STGO). Sirve para poner nombres de comunas donde hay poco espacio, como en mapas. Las abreviaciones no necesariamente son únicas.

## Usage

```
abreviar_comunas(nombre_comuna)
```

## Arguments

| | |
|---|---|
| `nombre_comuna` | Vector de nombres de comunas, como `comunas()` |

## Details

Las siglas se construyen con las tres primeras letras de las comunas con nombres de una palabra, y en las comuna de dos o más palabras con la primera letra de la primera palabra y las dos primeras letras de la segunda palabra. Se ajustan algunos casos, como eliminar símbolos (como en O'Higgins), y sacar la "u" de las comunas que empiezan con "Que" para reducir coincidencias de abreviaciones.

## Value

Vector con siglas de comunas de tres o cuatro caracteres

## See Also

`acortar_regiones()` para acortar nombres de regiones, y `redactar_comunas()` para redactar listas de comunas en una oración.

## Examples

```
abreviar_comunas(c("Cerrillos", "La Florida", "Quellón"))
```

`acortar_regiones` *Acortar nombres de las regiones de Chile*

## Description

Hay regiones de Chile con nombres extensos, y que en ciertos contextos requieren de una versión más breve. Esta función procesa el texto de los nombres de regiones para, por ejemplo, pasar desde "Aysén del General Carlos Ibáñez del Campo" a "Aysén".

## Usage

```
acortar_regiones(datos, variable = NULL)
```

## Arguments

| | |
|---|---|
| `datos` | Dataframe con una columna de nombres de regiones, o vector de nombres de regiones |
| `variable` | Columna del dataframe con los nombres de regiones (se pasa sin comillas, p.ej. `region`). Si no se especifica, se asume `nombre_region`. Si se aplica a un vector, omitir este argumento. |

## Value

Si la entrada es un dataframe, retorna el dataframe con la columna de regiones reemplazada por sus versiones breves. Si es un vector, retorna un vector de texto con nombres de regiones breves.

## See Also

`redactar_region()` para redactar nombres como "Región del Maule", y `abreviar_comunas()` para abreviar nombres de comunas.

## Examples

```
acortar_regiones("Libertador Gral. Bernardo O'Higgins")

territorios |>
  ordenar_regiones() |>
  dplyr::mutate(nombre_region_corto = acortar_regiones(nombre_region)) |>
  dplyr::select(nombre_region, nombre_region_corto)

datos <- dplyr::tibble(
  nombre_region = c(
    "Libertador Gral. Bernardo O'Higgins",
    "Aysén del General Carlos Ibáñez del Campo"
  ),
  valores = c(4, 6)
)

# si existe `nombre_region`, la función no requiere argumentos:
```

```
datos |>
  acortar_regiones()
```

---

`actualizar_codigo_comuna` *Actualizar códigos únicos territoriales comunales antiguos o desactualizados*

---

## Description

Función que recibe códigos únicos territoriales desactualizados (confirmar con `is_codigo_comuna_antiguo()`), y retorna las versiones actualizadas al 2017 (posterior a la creación de la región de Ñuble).

## Usage

```
actualizar_codigo_comuna(codigo_comuna)
```

## Arguments

| | |
|---|---|
| `codigo_comuna` | Códigos comunales en formato numérico |

## Value

Códigos comunales actualizados correspondientes a los desactualizados entregados. Si los códigos entregados no son antiguos, se mantienen sin cambios.

## See Also

`is_codigo_comuna_antiguo()` para detectar códigos desactualizados, y cut_historicos para la tabla de códigos históricos. Luego puedes usar `contextualizar()` para agregar las demás variables territoriales.

## Examples

```
actualizar_codigo_comuna(8408)
```

`agregar_clasificacion` *Agregar clasificación comunal PNDR a comunas*

## Description

Para un vector de comunas, entrega la clasificación comunal PNDR correspondiente (Rural, Mixta o Urbana), tal como aparecen en clasificacion.

## Usage

```
agregar_clasificacion(codigo_comuna, ordenar = TRUE)
```

## Arguments

| | |
|---|---|
| `codigo_comuna` | Vector de códigos de comuna, idealmente `codigo_comuna`. También puede ser un data frame con la columna `codigo_comuna`, en cuyo caso se retorna el data frame con la columna nueva `clasificacion`. |
| `ordenar` | Entregar resultados como un factor ordenado (Rural, Mixta, Urbana), o como textos sin orden. Por defecto entrega factor. |

## Value

Factor con clasificación comunal (Rural, Mixta o Urbana). Si la entrada es un data frame, retorna el data frame con la columna `clasificacion` agregada.

## See Also

clasificacion para la tabla completa de clasificación comunal, y `agregar_macrozona()` para agrupar regiones en macrozonas. Si no tienes códigos comunales, usa `as_codigo_comuna()` o `contextualizar()`.

## Examples

```
agregar_clasificacion(8107)

# con un data frame, usa la columna `codigo_comuna`
territorios |>
  agregar_clasificacion()
```

`agregar_macrozona` *Agregar macrozona del país a regiones*

## Description

Las macrozonas son agrupaciones de regiones de Chile que permiten entender el territorio en base a grupos geográficos. Para un vector de códigos de regiones (del 1 al 16), entrega las macrozonas correspondientes a cada región.

## Usage

```
agregar_macrozona(codigo_region, tipo = 1, ordenar = TRUE)
```

## Arguments

| | |
|---|---|
| `codigo_region` | Vector de códigos de región (del 1 al 16) o de nombres de región (ver `as_codigo_region()`), que se convierten automáticamente a códigos. También puede ser un data frame con la columna `codigo_region`, en cuyo caso se retorna el data frame con la columna nueva `macrozona`. |
| `tipo` | Tipo de macrozonas a aplicar. Por defecto se usa el tipo 1. Ver la documentación más arriba. |
| `ordenar` | Entregar resultados como un factor ordenado (de norte a sur), o como textos sin orden. Por defecto entrega factor. |

## Details

Como no existe una clasificación fija de macrozonas, existen varias alternativas para elegir:

- Macrozonas tipo 1: desde Arica a Coquimbo son *Norte*, desde Valparaíso a Maule *Centro*, desde Ñuble a Los Lagos *Sur*, y desde Aysén a Magallanes *Austral*.
- Macrozonas tipo 2: distribución balanceada por cantidad de regiones: 4 grupos de 4 regiones: *Norte*, *Centro*, *Centro/sur* y *Sur*.
- Macrozonas tipo 3: según las macrozonas del Ministerio de Ciencia, Tecnología, Conocimiento e Innovación, que definr 5 macrozonas (*Norte, Centro, Centro sur, Sur,* y *Austral*), y excluye a la Región Metropolitana. Para más información, #' revisar el decreto que establece a las Seremis del ministerio de Ciencia.
- Macrozonas tipo 4: basadas en el programa curricular de educación básica del Ministerio de Educación de Chile, existirían *Norte grande, Norte chico, Zona central, Zona sur* y *Zona austral*.
- Macrozonas tipo 5: dividen al país en *norte, centro* y *sur* según las agrupaciones del estudio Identificación de Localidades en Condición de Aislamiento 2012, de Subdere.

## Value

Factor con macrozonas regionales, de acuerdo al tipo de clasificación de regiones elegido. Si la entrada es un data frame, retorna el data frame con la columna `macrozona` agregada.

## See Also

ordenar_regiones() para ordenar regiones de norte a sur, y agregar_clasificacion() para clasificar comunas en urbanas, mixtas o rurales. Si no tienes códigos de región, usa as_codigo_region() o contextualizar().

## Examples

```
agregar_macrozona(c(15, 13, 12), tipo = 1)

# con un data frame, usa `codigo_region` o `codigo_comuna`
territorios |>
  agregar_macrozona(tipo = 2)

territorios |>
  dplyr::distinct(codigo_region, nombre_region) |>
  ordenar_regiones() |>
  dplyr::mutate(
    macrozona_1 = agregar_macrozona(codigo_region, tipo = 1),
    macrozona_2 = agregar_macrozona(codigo_region, tipo = 2),
    macrozona_3 = agregar_macrozona(codigo_region, tipo = 3),
    macrozona_4 = agregar_macrozona(codigo_region, tipo = 4)
 )
```

---

# agregar_orden_region *Agregar orden geográfico (norte a sur) a códigos de regiones*

---

## Description

Función usada internamente para ordenar_regiones(). Recibe un vector de números de región (entre 1 y 16) y retorna la posición de norte a sur de dicha región, de modo que Arica sea 1 y Magallanes sea 16.

## Usage

```
agregar_orden_region(codigo_region)
```

## Arguments

| | |
|---|---|
| codigo_region | Vector de códigos únicos de región (entre 1 y 16) |

## Value

Vector con la posición de norte a sur de la región respectiva

## See Also

ordenar_regiones() para ordenar directamente una tabla de datos de norte a sur, y agregar_macrozona() para agrupar regiones en macrozonas.

## Examples

```
agregar_orden_region(15) # Arica
```

---

`as_capital_regional` *Convertir regiones a capitales regionales*

---

## Description

Entregando códigos regionales (como los que aparecen en territorios) o nombres de regiones, retorna los nombres de las comunas que son capitales regionales correspondientes. Retorna NA si no corresponde con ninguna.

## Usage

```
as_capital_regional(codigos_regiones)
```

## Arguments

`codigos_regiones`
: Códigos regionales en formato numérico, o nombres de regiones (como en `regiones()`)

## Details

Esta función es útil para imputar datos cuando se cuenta con la región, pero no se tiene información de las comunas, en cuyo caso se podría imputar la capital regional de la región como comuna de reemplazo.

## Value

Vector con nombres de comunas capitales regionales

## See Also

`capitales_regionales()` para obtener la tabla de capitales regionales, `is_capital_regional()` para confirmar si las comunas son capitales, y `as_nombre_region()` para convertir códigos regionales a nombres de regiones.

## Examples

```
as_capital_regional(c(15, 4, 10, 99))

as_capital_regional(c("Maule", "Ñuble"))
```

as_codigo_comuna *Convertir nombres de comunas a códigos comunales*

## Description

Entregando nombres de comuna correctos (como los que aparecen en comunas() o en territorios), retorna los códigos de comuna correspondientes en formato numérico. Retorna NA si no corresponde con ninguna.

## Usage

```
as_codigo_comuna(nombres_comunas)
```

## Arguments

| | |
|---|---|
| nombres_comunas | Nombres de comuna (como los que aparecen en comunas()) en formato caracter. |

## Details

Para más información sobre los códigos únicos territoriales, revisa la viñeta vignette("codigos_unicos_territoriales")

## Value

Vector numérico con códigos de comuna.

## See Also

as_nombre_comuna() para la conversión inversa, y limpiar_comunas() para corregir los nombres antes de convertirlos. Para regiones, usa as_codigo_region().

## Examples

```
as_codigo_comuna("La Florida")
```

## as_codigo_region *Convertir nombres de regiones a códigos regionales*

### Description

Entregando nombres de región correctos (como los que aparecen en territorios), retorna los códigos de región correspondientes en formato numérico (números del 1 al 16). Retorna NA si no corresponde con ninguna.

### Usage

```
as_codigo_region(nombres_regiones)
```

### Arguments

| | |
|---|---|
| nombres_regiones | Nombres de región (como los que aparecen en territorios) en formato caracter. |

### Details

Para más información sobre los códigos únicos territoriales, revisa la viñeta `vignette("codigos_unicos_territoriales")`

### Value

Vector numérico con códigos de región.

### See Also

`as_nombre_region()` para la conversión inversa, y `limpiar_regiones()` para corregir los nombres antes de convertirlos. Para comunas, usa `as_codigo_comuna()`.

### Examples

```
as_codigo_region("Metropolitana de Santiago")
```

## as_nombre_comuna *Convertir códigos comunales a nombres de comunas*

### Description

Entregando códigos comunales (como los que aparecen en territorios), retorna los nombres de comuna correspondientes. Retorna NA si no corresponde con ninguna.

### Usage

```
as_nombre_comuna(codigos_comunas)
```

### Arguments

codigos_comunas
: Códigos comunales en formato numérico

### Value

Vector con nombres de comuna

### See Also

`as_codigo_comuna()` para la conversión inversa, e `is_codigo_comuna()` para verificar que los códigos sean válidos. Si tus códigos son anteriores a 2018, revisa `actualizar_codigo_comuna()`.

### Examples

```
as_nombre_comuna(1101)

as_nombre_comuna(c(1401, 1403, 9999, 1404))
```

---

## as_nombre_region *Convertir códigos regionales a nombres de regiones*

---

### Description

Entregando códigos regionales (como los que aparecen en territorios), retorna los nombres de región correspondientes. Retorna NA si no corresponde con ninguna.

### Usage

```
as_nombre_region(codigos_regiones)
```

### Arguments

codigos_regiones
: Códigos regionales en formato numérico

### Value

Vector con nombres de región

### See Also

`as_codigo_region()` para la conversión inversa, y `redactar_region()` o `acortar_regiones()` para cambiar cómo se escriben los nombres resultantes.

## Examples

```
as_nombre_region(16)

as_nombre_region(c(15, 4, 10, 99))
```

---

`buscar_comuna` *Buscar comunas por similitud*

---

## Description

Usar un término de búsqueda para obtener una tabla con comunas de nombres similares. Funciona con cualquier tabla de datos que contenga una columna con nombres de comunas, por lo que esta función puede ayudarte a encontrar comunas en fuentes externas y así confirmar o solucionar problemas. Para buscar sobre las comunas oficiales de Chile, usa territorios.

## Usage

```
buscar_comuna(
  datos,
  texto,
  columna = nombre_comuna,
  similitud = 0.9,
  cantidad = 6
)
```

## Arguments

| | |
|---|---|
| `datos` | Tabla de datos sobre la cual buscar, por ejemplo territorios. Debe contener una columna con nombres de comunas (especificada en el argumento `columna`). |
| `texto` | Texto que quieres buscar entre las comunas |
| `columna` | Columna de `datos` que contiene los nombres de comunas a comparar. Por defecto `nombre_comuna`. Se puede especificar sin comillas (tidy evaluation) o como texto. |
| `similitud` | El nivel de similitud mínimo a retornar, donde 1 es total similitud y 0 es nula similitud. Por defecto es 0,9. |
| `cantidad` | Cantidad máxima de resultados, por defecto 6. Poner `Inf` para mostrar todos. |

## Value

Mensajes respecto de la búsqueda, y el dataframe entregado en el argumento `datos` filtrado según comunas que cumplan con el criterio de `similitud` de la búsqueda.

## See Also

`limpiar_comunas()` para corregir automáticamente nombres de comunas mal escritos, e `is_nombre_comuna()` para verificar si un nombre es válido. Para ubicar localidades más pequeñas que una comuna, usa `ubicar_localidades()`.

## Examples

```
territorios |>
 buscar_comuna("peña") |>
 dplyr::select(nombre_comuna, puntaje)

territorios |>
 buscar_comuna("alto") |>
 dplyr::select(nombre_comuna, puntaje)

territorios |>
 buscar_comuna("antofagasta") |>
 dplyr::select(nombre_comuna, puntaje)

territorios |>
 buscar_comuna("perro") |>
 dplyr::select(nombre_comuna, puntaje)
```

---

# capitales_regionales *Capitales regionales de Chile*

---

## Description

Tabla de datos basada en territorios que contiene las comunas donde están ubicadas las capitales regionales de cada región.

## Usage

```
capitales_regionales()
```

## Value

*Dataframe* de 16 filas

## See Also

Para ver todas las comunas, regiones y provincias de Chile, usa territorios. Usa `regiones()` para un vector con todas las regiones, o `comunas()` para un vector con todas las comunas de Chile.

## Examples

```
capitales_regionales()
```

`clasificacion` *Tabla de clasificación comunal PNDR, Censo 2024*

## Description

La Oficina de Estudios y Políticas Agrarias (Odepa) informó que, en el marco de la Política Nacional de Desarrollo Rural (PNDR), se realizó la actualización de la clasificación comunal utilizando los resultados del Censo de Población y Vivienda 2024. La PNDR, promulgada en 2020, establece una tipología comunal que distingue entre comunas predominantemente urbanas, mixtas y predominantemente rurales, con el objetivo de fortalecer el enfoque territorial en el diseño, implementación y evaluación de políticas públicas, programas e instrumentos de planificación. Como resultado de esta actualización, ocho comunas del país modifican su categoría respecto de la clasificación elaborada con base en el Censo 2017. En particular, dos comunas pasan de la categoría mixta a urbana, tres transitan de mixta a rural y otras tres cambian de rural a mixta. Las comunas que registran estos cambios son Vallenar y Maule, que pasan de mixtas a urbanas; Putaendo, Gorbea y Yungay, que pasan de mixtas a rurales; y Olmué, San Clemente y Cabrero, que cambian de rurales a mixtas. De acuerdo con esta actualización, 84 comunas del país se clasifican como predominantemente urbanas, 76 como mixtas y 185 como predominantemente rurales. En términos de población, el 25,1% de las personas reside en comunas mixtas y rurales, proporción que se mantiene en línea con la medición realizada a partir del Censo 2017. Los criterios técnicos y metodológicos utilizados para esta actualización, junto con el detalle completo de los resultados, se encuentran disponibles en la Minuta Técnica y en la planilla de clasificación comunal elaboradas por Odepa, disponibles en su Biblioteca Digital.

## Usage

```
clasificacion
```

## Format

Data frame con 345 filas y 7 columnas:

**codigo_region** Código único territorial de las regiones de Chile (número entre 1 y 16)

**nombre_region** Nombre de las regiones de Chile

**codigo_provincia** Código único territorial de las provincias de Chile

**nombre_provincia** Nombre de las provincias de Chile

**codigo_comuna** Código único territorial de las comunas de Chile

**nombre_comuna** Nombre de las comunas de Chile

**clasificacion** Clasificación comunal PNDR: Rural, Mixta o Urbana

## Source

https://bibliotecadigital.odepa.gob.cl/items/ce846880-f010-4868-a5d9-18c80f7ceabd

## See Also

`agregar_clasificacion()` para agregar esta clasificación a un vector de códigos comunales.

## codigos_comunas *Códigos de comunas de Chile*

### Description

Vector numérico que contiene los códigos únicos territoriales de las 346 comunas de Chile, tal como aparecen en la tabla territorios.

### Usage

```
codigos_comunas()
```

### Value

Vector de códigos de comunas de Chile

### See Also

comunas(), regiones()

### Examples

```
codigos_comunas()
```

## comunas *Nombres de comunas de Chile*

### Description

Vector de texto que contiene el nombre de las 346 comunas de Chile, tal como aparecen en la tabla territorios.

### Usage

```
comunas()
```

### Value

Vector de nombres de comunas de Chile

### See Also

codigos_comunas() para un vector de códigos únicos territoriales de comunas, regiones() para un vector de regiones de Chile. Puedes usar as_codigo_comuna() para obtener códigos únicos territoriales (CUT) a partir de los nombres de comunas.

## Examples

```
comunas()
```

---

`contar_comunas` *Conteo de comunas disponibles, indicando las faltantes*

---

## Description

Revisa un vector o una tabla de datos (asumiendo la columna `nombre_comuna`, o indicando una columna que contenga nombres de comunas) e indica el conteo de comunas únicas, y emite mensajes dependiendo de esta cantidad: si es correcta (345 o 346), si es menor (indicando las que faltan) o si es mayor (recomendando validar con `validar_comunas()`).

## Usage

```
contar_comunas(datos, variable = NULL, largo = 30)
```

## Arguments

| | |
|---|---|
| `datos` | Dataframe con una columna de nombre de comunas, o vector de nombres de comunas |
| `variable` | Columna del dataframe con los nombres de comunas (se pasa sin comillas, p.ej. `comuna`) |
| `largo` | Si faltan comunas en los datos, enumera esta cantidad de comunas. Por defecto son 30, y el argumento se pasa a `redactar_comunas()`. |

## Details

Sirve para revisar rápidamente que los datos abarquen todos los territorios del país, o indicar si faltan algunos.

## Value

Dataframe o vector intacto pero en modo invisible, con mensajes de conteo de comunas

## See Also

`validar_comunas()` y `limpiar_comunas()` para revisar y corregir los nombres de comunas, y `obtener_comunas()` para obtener las comunas de una región.

## Examples

```
territorios |>
  dplyr::slice_sample(n = 300) |>
  contar_comunas()
```

`contextualizar` *Contextualizar datos de nivel comunal con variables territoriales*

## Description

Para cualquier tabla de datos de nivel comunal que tenga al menos una variable territorial (`codigo_comuna` o `nombre_comuna`), indicar esta variable para agregar todo el resto de variables territoriales. De esta manera, se contextualizan territorialmente los datos al agregar todas las variables territoriales faltantes.

## Usage

```
contextualizar(datos, variable = NULL)
```

## Arguments

| | |
|---|---|
| `datos` | Dataframe de datos comunales al que se quieren agregar columnas con variables territoriales |
| `variable` | Variable territorial ya existente en el dataframe (`codigo_comuna` o `nombre_comuna`). Si se omite, se asume `nombre_comuna`. |

## Details

Se usa, por ejemplo, cuando tienes una tabla de datos que tienen solamente los nombres de las comunas (primero valída con `validar_comunas()` y/o limpia con `limpiar_comunas()`), y quieres agregarle más variables territoriales como los códigos únicos territoriales, las provincias, regiones, etc.

## Value

El mismo dataframe con las columnas `codigo_region`, `nombre_region`, `codigo_provincia`, `nombre_provincia`, `codigo_comuna` y `nombre_comuna` agregadas al inicio.

## See Also

`validar_comunas()` para revisar el estado de calidad de los nombres de comunas de Chile, `limpiar_comunas()` para corregir los nombres de comunas existentes y así poder usar `contextualizar()` o `as_codigo_comuna()`.

## Examples

```
datos <- dplyr::tribble(
  ~nombre_comuna, ~valor,
  "Cerrillos",    1,
  "Arica",        2,
  "Putre",        3)

datos |>
  contextualizar(nombre_comuna)
```

```
# si ya existe una columna `nombre_comuna`, puede omitirse el argumento
datos |>
  contextualizar()
```

---

`cut_historicos` *Códigos únicos territoriales históricos*

---

## Description

Esta tabla contiene los códigos únicos territoriales de las comunas de Chile junto a los códigos que han tenido a través de los años. Sirve para identificar cambios de CUT y convertir CUT históricos a CUT vigentes.

## Usage

```
cut_historicos
```

## Format

Un data frame con 345 filas y 5 columnas:

**nombre_comuna** Nombre de las comunas de Chile

**codigo_comuna_2017** Código único territorial actual y vigente de las comunas de Chile

**codigo_comuna_2010** Código único territorial histórico (desactualizado y no vigente) de las comunas de Chile en el año 2010

**codigo_comuna_2007** Código único territorial histórico (desactualizado y no vigente) de las comunas de Chile en el año 2007

**codigo_comuna_2004** Código único territorial histórico (desactualizado y no vigente) de las comunas de Chile en el año 2004

## Details

Cada vez que se ha creado una nueva provincia o región de Chile, las comunas que las componen cambian sus códigos únicos territoriales para coincidir con la cifra de la unidad mayor que las contienen. Esta tabla permite identificar qué código único territorial vigente tienen hoy las comunas a partir de sus códigos antiguos o históricos (no vigentes).

## Source

https://www.subdere.gov.cl

## See Also

`is_codigo_comuna_antiguo()` y `actualizar_codigo_comuna()` para detectar y actualizar códigos antiguos.

## eliminar_texto *Eliminar caracteres al azar para ensuciar texto*

### Description

Eliminar caracteres al azar para ensuciar texto

### Usage

```
eliminar_texto(texto, porcentaje = 0.1)
```

### Arguments

| | |
|---|---|
| texto | Vector de texto |
| porcentaje | Porcentaje de caracteres a eliminar |

### Value

Vector de texto con caracteres eliminados al azar

### See Also

insertar_texto() y reemplazar_texto() para otras formas de ensuciar texto, útiles para probar limpiar_comunas().

### Examples

```
eliminar_texto(c("mapache", "lindo"), porcentaje = 0.3)
```

## insertar_texto *Insertar caracteres al azar para ensuciar texto*

### Description

Insertar caracteres al azar para ensuciar texto

### Usage

```
insertar_texto(texto, porcentaje = 0.1, caracteres = c(letters, LETTERS))
```

### Arguments

| | |
|---|---|
| texto | Vector de texto |
| porcentaje | Porcentaje de caracteres a insertar |
| caracteres | Vector de caracteres para insertar |

### Value

Vector de texto con caracteres insertados al azar

### See Also

`eliminar_texto()` y `reemplazar_texto()` para otras formas de ensuciar texto, útiles para probar `limpiar_comunas()`.

### Examples

```
insertar_texto(c("mapache", "lindo"), porcentaje = 0.3)
```

---

## is_capital_regional *Evaluar si una comuna corresponde a una capital regional*

---

### Description

Dado un vector de cualquier largo, retorna TRUE o FALSE para cada elemento de acuerdo si la comuna corresponde a una de las 16 capitales regionales de Chile, disponibles en `capitales_regionales()`.

### Usage

```
is_capital_regional(codigo_comuna)
```

### Arguments

| | |
|---|---|
| `codigo_comuna` | Códigos de comuna en formato numérico, o nombres de comuna (como los que aparecen en `comunas()`) en formato caracter. |

### Details

Acepta tanto códigos de comuna en formato numérico como nombres de comuna en formato caracter.

### Value

Vector lógico del mismo largo que `codigo_comuna`: TRUE o FALSE si es o no es una capital regional.

### See Also

`capitales_regionales()` para ver la tabla de capitales regionales, `as_capital_regional()` para convertir regiones a comunas capitales, `as_codigo_comuna()` para convertir nombres de comunas a códigos, y `is_codigo_comuna()` para validar códigos de comunas.

## Examples

```
is_capital_regional(13101)

is_capital_regional(c("Santiago", "Puente Alto", "Valparaíso"))
```

---

`is_codigo_comuna` *Evaluar si un dato corresponde a un código territorial válido de una comuna de Chile*

---

## Description

Dado un vector de cualquier largo, retorna TRUE o FALSE para cada elemento de acuerdo si se corresponde con los códigos únicos territoriales de comunas de Chile, disponibles en territorios.

## Usage

```
is_codigo_comuna(codigo_comuna)
```

## Arguments

| | |
|---|---|
| `codigo_comuna` | Códigos territoriales a evaluar, en formato numérico. Si vienen en formato caracter, se convierten. |

## Details

Para más información sobre los códigos únicos territoriales, revisa la viñeta `vignette("codigos_unicos_territoriales")`

## Value

Retorna TRUE o FALSE si es o no es un código único territorial válido (ver territorios)

## See Also

`as_nombre_comuna()` para convertir códigos de comunas a nombres de comunas, `is_codigo_comuna_antiguo()` y `actualizar_codigo_comuna()` para detectar y corregir códigos desactualizados.

## Examples

```
is_codigo_comuna(1101)
```

## is_codigo_comuna_antiguo *Evaluar si un código único territorial comunal es antiguo o desactualizado*

### Description

Evaluar si un código único territorial comunal es antiguo o desactualizado

### Usage

```
is_codigo_comuna_antiguo(codigo_comuna)
```

### Arguments

| | |
|---|---|
| codigo_comuna | Códigos comunales en formato numérico |

### Value

Retorna TRUE o FALSE si es o no es un código único territorial antiguo (ver cut_historicos)

### See Also

`actualizar_codigo_comuna()` para convertir códigos antiguos a los vigentes, e `is_codigo_comuna()` para verificar si un código es válido.

### Examples

```
is_codigo_comuna_antiguo(8408)
```

## is_codigo_region *Evaluar si un dato corresponde a un código de región de Chile válido*

### Description

Dado un vector de cualquier largo, retorna TRUE o FALSE para cada elemento de acuerdo si se corresponde con los códigos de región de Chile (números del 1 al 16), disponibles en territorios.

### Usage

```
is_codigo_region(codigo_region)
```

### Arguments

| | |
|---|---|
| codigo_region | Códigos de región a evaluar, en formato numérico. |

## Details

Para más información sobre los códigos únicos territoriales, revisa la viñeta `vignette("codigos_unicos_territoriales")`

## Value

Vector lógico del mismo largo que `codigo_region`: TRUE o FALSE si es o no es un código de región válido.

## See Also

`as_codigo_region()` para convertir nombres de regiones a códigos, `is_nombre_region()` para validar nombres de regiones, y `is_codigo_comuna()` para códigos de comunas.

## Examples

```
is_codigo_region(7)

is_codigo_region(c(1, 13, 16, 17, NA))
```

---

# is_nombre_comuna *Evaluar si un texto corresponde al nombre válido de una comuna de Chile*

---

## Description

Dado un vector de cualquier largo, retorna TRUE o FALSE para cada elemento de acuerdo si se corresponde con los nombres de comunas oficiales, disponibles en la función `comunas()`.

## Usage

```
is_nombre_comuna(nombre_comuna, intentar = FALSE)
```

## Arguments

| | |
|---|---|
| `nombre_comuna` | Elemento/s a evaluar |
| `intentar` | Si la comuna no es válida, tratar de interpretarla con `limpiar_comunas()` |

## Value

TRUE o FALSE si es o no es una comuna válida

## See Also

`validar_comunas()` para un diagnóstico detallado de los problemas, y `limpiar_comunas()` para corregirlos. Para regiones, usa `is_nombre_region()`.

## Examples

```
is_nombre_comuna("Panguipulli")
```

## is_nombre_region *Evaluar si un texto corresponde a un nombre de región de Chile válido*

### Description

Dado un vector de cualquier largo, retorna TRUE o FALSE para cada elemento de acuerdo si se corresponde con los nombres de regiones de Chile, disponibles en territorios.

### Usage

```
is_nombre_region(nombre_region)
```

### Arguments

| | |
|---|---|
| nombre_region | Elemento/s a evaluar |

### Value

TRUE o FALSE si es o no es un código único territorial válido

### See Also

validar_regiones() para un diagnóstico detallado de los problemas, y limpiar_regiones() para corregirlos. Para comunas, usa is_nombre_comuna().

### Examples

```
is_nombre_region("Maule")
```

## limpiar_comunas *Limpieza de nombres de comunas de Chile a sus nombres oficiales*

### Description

A partir de un dataframe con una variable de nombres de comuna (idealmente nombre_comuna), o un vector de nombres de comunas, se realizan cuatro pasos de limpieza (confirmación de nombres correctos, limpieza de texto, limpieza de casos especiales, y detección por coincidencia) para retornar los nombres de comunas oficiales apropiados. Los nombres de comunas considerados *limpios* son los que aparecen en comunas().

### Usage

```
limpiar_comunas(
  datos,
  variable = NULL,
  aproximar = TRUE,
  procedimiento = FALSE
)
```

## Arguments

| | |
|---|---|
| `datos` | Dataframe con una columna de nombres de comunas, o vector de nombres de comunas |
| `variable` | Columna del dataframe con los nombres de comunas (se pasa sin comillas, p.ej. `comuna`). Si no se especifica, se asume `nombre_comuna`. Si se aplica a un vector, omitir este argumento. |
| `aproximar` | El paso de limpieza por aproximación y coincidencia de nombres puede entregar resultados inexactos. Cambiar a FALSE para omitir. |
| `procedimiento` | Mostrar una tabla con los resultados intermedios del proceso de limpieza. Elegir entre TRUE o FALSE, por defecto FALSE. |

## Details

Los nombres se limpian en cuatro pasos:

1. Contrastando los nombres entregados con los nombres correctos de las comunas (`comunas()`), para ver si hay comunas bien escritas antes de proseguir con la limpieza de las demás.
2. Se *limpian* los nombres de comunas entregados, transformándolos a minúsculas y eliminando todo tipo de símbolos posibles, para dejar las palabras en sus formas más básicas (por ejemplo, `Ñuñoa` se vuelve `nunoa`). Luego, se aplica el mismo proceso a los nombres de comunas correctos (`comunas()`), y se hace un cruce entre ambos conjuntos de nombres: si los nombres coinciden, significa que se entregaron nombres de comunas escritos en mayúsculas o minúsculas, comunas sin tildes o con tildes extra, comunas sin símbolos especiales o si eñe, entre otras, y son reemplazadas con sus versiones correctas.
3. Se buscan algunos casos especiales de comunas que son típicamente mal escritos, pero que son difíciles de identificar de manera automática, por ejemplo, cuando a la comuna de *Cabo de Hornos* le ponen "Ex-Navarino".
4. Si en los pasos anteriores quedaron comunas que no coincidieron (es decir, que sus problemas van más allá de tildes, mayúsculas o símbolos), se realiza una coincidencia parcial de textos o *fuzzy matching* usando la función `base::agrepl()`, que utiliza el algoritmo de distancia de Levenshtein para encontrar las comunas correctamente escritas que más se parecen a las comunas entregadas. Se esta forma, se pueden encontrar las comunas correctamente escritas para casos de comunas con faltas de ortografía (`Pobidencia` en vez de `Providencia`), comunas sin espacios entre sus palabras (`laflorida` en vez de `La Florida`), y formas alternativas de escribir las comunas (`llay-llay` en vez de `Llaillay`). En todos estos casos se emite una alerta que indica la coincidencia encontrada, ya que al ser una aproxmación, no se garantiza que la coincidencia sea correcta. Puedes desactivar este paso poniendo `aproximar = FALSE`. Finalmente, se muestra una tabla que describe el proceso de limpieza para su revisión (que puede ocultarse con `procedimiento = FALSE`, y se retornan las comunas correctas.

## Value

Si la entrada es un dataframe, retorna el dataframe con la columna de comunas reemplazada. Si es un vector, retorna un vector de nombres de comunas con correcciones aplicadas.

## See Also

validar_comunas() para revisar el estado de los nombres de comunas antes de limpiarlos, comunas() para un vector con las comunas de Chile, codigos_comunas() para un vector de códigos únicos territoriales de comunas. También revisa contextualizar() para complementar tablas de datos con otras variables territoriales a partir de los nombres de comunas. También puedes usar as_codigo_comuna() para obtener códigos únicos territoriales (CUT) a partir de los nombres de comunas.

## Examples

```
limpiar_comunas(
  c("COLCHANE", "Alto Ospicio", "probidencia",
  "huara", "laflorida", "cerritos", "llay-llay")
)

datos <- dplyr::tibble(
  nombre_comuna = c("PIRQUE", "El Monte", "Maipu",
                    "santiago", "prohibidencia", "CERRILLOS",
                    "San José De Maipo", "OHiggins"),
  valores = c(4, 6, 2, 8, 6, 3, 5, 8)
  )

# si existe `nombre_comuna`, la función no requiere argumentos:
datos |>
  limpiar_comunas()

# también puede usarse sobre un vector:
datos |>
  dplyr::mutate(nombre_corregido = limpiar_comunas(nombre_comuna))
```

---

# limpiar_regiones *Limpieza de nombres de regiones de Chile a sus nombres oficiales*

---

## Description

A partir de un dataframe con una variable de nombres de región (idealmente nombre_region), o un vector de nombres de regiones, se realizan cuatro pasos de limpieza (confirmación de nombres correctos, limpieza de texto, limpieza de casos especiales, y detección por coincidencia) para retornar los nombres de regiones oficiales apropiados. Los nombres de regiones considerados *limpios* son los que aparecen en regiones(), incluyendo también sus versiones cortas (ver acortar_regiones()); por ejemplo, tanto "Aysén" como "Aysén del General Carlos Ibáñez del Campo" se consideran nombres limpios.

## Usage

```
limpiar_regiones(
  datos,
  variable = NULL,
```

```
  aproximar = TRUE,
  procedimiento = FALSE
)
```

## Arguments

| | |
|---|---|
| `datos` | Dataframe con una columna de nombres de regiones, o vector de nombres de regiones |
| `variable` | Columna del dataframe con los nombres de regiones (se pasa sin comillas, p.ej. `region`). Si no se especifica, se asume `nombre_region`. Si se aplica a un vector, omitir este argumento. |
| `aproximar` | El paso de limpieza por aproximación y coincidencia de nombres puede entregar resultados inexactos. Cambiar a FALSE para omitir. |
| `procedimiento` | Mostrar una tabla con los resultados intermedios del proceso de limpieza. Elegir entre TRUE o FALSE, por defecto FALSE. |

## Details

Los nombres se limpian en cuatro pasos:

1. Contrastando los nombres entregados con los nombres correctos de las regiones (`regiones()`) y sus versiones cortas (`acortar_regiones()`), para ver si hay regiones bien escritas antes de proseguir con la limpieza de las demás.
2. Se *limpian* los nombres de regiones entregados, transformándolos a minúsculas y eliminando todo tipo de símbolos posibles, para dejar las palabras en sus formas más básicas (por ejemplo, `Ñuble` se vuelve `nuble`). Luego, se aplica el mismo proceso a los nombres de regiones correctos (largos y cortos), y se hace un cruce entre ambos conjuntos de nombres: si los nombres coinciden, significa que se entregaron nombres de regiones escritos en mayúsculas o minúsculas, regiones sin tildes o con tildes extra, regiones sin símbolos especiales, entre otras, y son reemplazadas con sus versiones correctas.
3. Se buscan algunos casos especiales de regiones que son típicamente mal escritos, pero que son difíciles de identificar de manera automática, por ejemplo, cuando a la Región Metropolitana le dicen "RM" o "Santiago", o cuando a la región de Aysén le ponen "Aisén" (con i latina).
4. Si en los pasos anteriores quedaron regiones que no coincidieron (es decir, que sus problemas van más allá de tildes, mayúsculas o símbolos), se realiza una coincidencia parcial de textos o *fuzzy matching* usando la función `base::agrepl()`, que utiliza el algoritmo de distancia de Levenshtein para encontrar las regiones correctamente escritas que más se parecen a las regiones entregadas. En todos estos casos se emite una alerta que indica la coincidencia encontrada, ya que al ser una aproximación, no se garantiza que la coincidencia sea correcta. Puedes desactivar este paso poniendo `aproximar = FALSE`. Finalmente, se muestra una tabla que describe el proceso de limpieza para su revisión (que puede ocultarse con `procedimiento = FALSE`), y se retornan los nombres de regiones oficiales (en su versión larga).

## Value

Si la entrada es un dataframe, retorna el dataframe con la columna de regiones reemplazada. Si es un vector, retorna un vector de nombres de regiones oficiales (en su versión larga) con correcciones aplicadas.

## See Also

validar_regiones() para revisar los nombres de regiones antes de limpiarlos, regiones() para un vector con las regiones de Chile, y acortar_regiones() o redactar_region() para cambiar cómo se escriben sus nombres. Para comunas, usa limpiar_comunas().

## Examples

```
limpiar_regiones(c("MAULE", "biobio", "la araucania", "Los rios", "RM", "aisen"))

datos <- dplyr::tibble(
  nombre_region = c("TARAPACA", "Coquimbo", "valparaiso",
                    "santiago", "ohiggins", "NUBLE"),
  valores = c(4, 6, 2, 8, 6, 3)
  )

# si existe `nombre_region`, la función no requiere argumentos:
datos |>
  limpiar_regiones()

# también puede usarse sobre un vector:
datos |>
  dplyr::mutate(nombre_corregido = limpiar_regiones(nombre_region))
```

---

# limpiar_texto *Eliminar puntuación, símbolos y números de textos*

---

## Description

Eliminar puntuación, símbolos y números de textos

## Usage

```
limpiar_texto(texto)
```

## Arguments

| | |
|---|---|
| texto | Vector de texto |

## Value

Vector de texto sin puntuación, símbolos y números, en minúsculas y sin acentos

## See Also

limpiar_comunas() y limpiar_regiones(), que usan este tipo de limpieza para corregir nombres de comunas y regiones.

## Examples

```
limpiar_texto("Ñuñoa")
```

`localidades` *Tabla de localidades de Chile*

## Description

Tabla de datos que contiene todas las localidades de Chile, asentamientos o áreas subcomunales que no son entidades administrativas. Según la definición del INE, una localidad es un "Área geográfica con nombre propio de conocimiento generalizado por la población. Puede estar poblada o no, aunque debe contener, a lo menos, una vivienda susceptible de ser habitada. Generalmente, se circunscribe dentro de un distrito censal; no obstante, puede rebasar a éste." Esta tabla de localidades proviene del estudio de **Identificación de localidades en condición de aislamiento**, realizado por la Subsecretaría de Desarrollo Regional y Administrativo y publicado el año 2021.

## Usage

```
localidades
```

## Format

Un data frame con 29.256 filas y 6 columnas:

**codigo_comuna** Código único territorial de las comunas de Chile

**localidad** Nombre de la localidad

**tipo** Tipo de localidad

**subtipo** Subtipo de la localidad

**habitantes** Habitantes de la localidad

**viviendas** Viviendas de la localidad

## Source

https://ide.subdere.gov.cl/estudio-actualizacion-de-base-censal-identificacion-de-localidades-en-c

## See Also

`ubicar_localidades()` para encontrar la comuna de una localidad.

## obtener_comunas *Obtener las comunas de una región de Chile*

### Description

Entregando uno o más códigos de región o nombres de región, retorna un vector con los nombres de las comunas correspondientes.

### Usage

```
obtener_comunas(codigo_region)
```

### Arguments

| | |
|---|---|
| codigo_region | Uno o más códigos de región en formato numérico (como aparecen en territorios), o nombres de región (como en `regiones()`) en formato caracter o factor. |

### Value

Vector de comunas de la región o regiones

### See Also

`ubicar_comunas()` para hacer lo inverso (obtener la región de una comuna), y `comunas()` o `regiones()` para vectores con todas las comunas o regiones del país.

### Examples

```
obtener_comunas("Tarapacá")

obtener_comunas(c(15, 1))
```

## ordenar_regiones *Ordenar regiones de Chile geográficamente*

### Description

Ordena una columna de nombres de regiones de Chile (`nombre_region`) a partir de la columna `codigo_region`, resultando en una columna `nombre_region` en formato factor, ordenada de norte a sur.

### Usage

```
ordenar_regiones(datos, limpiar = TRUE, ordenar = TRUE, invertir = FALSE)
```

## Arguments

| | |
|---|---|
| datos | Tabla de datos que contenga las columnas nombre_region y codigo_region |
| limpiar | Para ordenar la variable nombre_region, se crea la variable intermedia orden_region, que por defecto se elimina después de usarla. Cambiar a FALSE para mantenerla. |
| ordenar | ¿Ordenar la tabla de datos según regiones? (por defecto TRUE) |
| invertir | Ordenar las regiones al revés, útil para visualizaciones de datos (para que al usar las regiones en el eje vertical de un gráfico, las regiones del norte aparezcan primero) |

## Value

Tabla de datos con la variable nombre_region en formato factor, ordenado geogr\u00e1ficamente (de norte a sur)

## See Also

agregar_orden_region() para obtener solo la posición de norte a sur, y agregar_macrozona() para agrupar regiones en macrozonas. Si falta codigo_region, usa contextualizar() o as_codigo_region().

## Examples

```
regiones <- dplyr::tribble(
  ~codigo_region,                  ~nombre_region,
  1,                                   "Tarapacá",
  2,                                "Antofagasta",
  3,                                    "Atacama",
  4,                                   "Coquimbo",
  5,                                 "Valparaíso",
  6,      "Libertador General Bernardo O'Higgins",
  7,                                     "Maule",
  8,                                    "Biobío",
  9,                                "La Araucanía",
  10,                                 "Los Lagos",
  11, "Aysén del General Carlos Ibáñez del Campo",
  12,      "Magallanes y de la Antártica Chilena",
  13,                 "Metropolitana de Santiago",
  14,                                 "Los Ríos",
  15,                         "Arica y Parinacota",
  16,                                     "Ñuble"
  )

regiones |>
  ordenar_regiones()
```

## preposicion_region *Preposición (de/del) de cada región de Chile*

### Description

Esta función entrega la preposición usada para cada región de Chile; por ejemplo, Valparaíso es la "Región *de* Valparaíso", pero Maule es la "Región *del* Maule".

### Usage

```
preposicion_region(nombre_region)
```

### Arguments

| | |
|---|---|
| nombre_region | Nombres de región, como aparecen en regiones() |

### Value

Vector de preposiciones para cada nombre de región. Retorna NA si no se detecta la región.

### See Also

redactar_region() para redactar el nombre completo de la región ("Región del Maule").

### Examples

```
preposicion_region("Ñuble")

preposicion_region("O'Higgins")
```

## redactar_comunas *Redactar una secuencia de comunas en una oración*

### Description

Para un vector de nombres de comuna, retorna un texto con las comunas redactadas y separadas por comas. Por defecto, sólo redacta un texto de 10 comunas y al final pone "y x comunas más".

### Usage

```
redactar_comunas(nombre_comuna, largo = 10)
```

### Arguments

| | |
|---|---|
| nombre_comuna | Nombres de comunas |
| largo | Por defecto, si son demasiadas comunas, trunca el largo a 10 comunas e indica "y x comunas más". Cambiar al número máximo de comunas, o 0 para que muestre todas las comunas. |

## Value

Texto con comunas separadas por comas

## See Also

`abreviar_comunas()` para siglas de comunas, y `redactar_region()` para redactar nombres de regiones.

## Examples

```
redactar_comunas(c("Paine", "Isla de Maipo", "Pirque"))
```

---

`redactar_region` *Redactar nombres de regiones de Chile*

---

## Description

A partir de los nombres de regiones de Chile (como aparecen en `regiones()`), retorna la redacción de los nombres incluyendo "Región de/del"; por ejemplo, "Maule" se vuelve en "Región del Maule".

## Usage

```
redactar_region(nombre_region)
```

## Arguments

| | |
|---|---|
| `nombre_region` | Nombres de región, como aparecen en `regiones()` |

## Value

Vector de nombres de región redactados, incluyendo la palabra "Región", y la preposición apropiada ('de' o 'del', según `preposicion_region()`) para cada nombre de región.

## See Also

`preposicion_region()` para obtener solo la preposición, y `acortar_regiones()` para versiones breves de los nombres de regiones.

## Examples

```
redactar_region("Maule")

redactar_region("Aysén")
```

## reemplazar_texto *Reemplazar caracteres al azar para ensuciar texto*

### Description

Reemplazar caracteres al azar para ensuciar texto

### Usage

```
reemplazar_texto(texto, porcentaje = 0.1, caracteres = c(letters, LETTERS))
```

### Arguments

| | |
|---|---|
| texto | Vector de texto |
| porcentaje | Porcentaje de caracteres a reemplazar |
| caracteres | Vector de caracteres para insertar |

### Value

Vector de texto con caracteres reemplazados al azar

### See Also

eliminar_texto() e insertar_texto() para otras formas de ensuciar texto, útiles para probar limpiar_comunas().

### Examples

```
reemplazar_texto(c("mapache", "lindo"), porcentaje = 0.3)
```

## regiones *Nombres de regiones de Chile*

### Description

Vector de texto que contiene el nombre de las 15 regiones de Chile, ordenadas de norte a sur, tal como aparecen en la tabla territorios.

### Usage

```
regiones()
```

### Value

Vector de nombres de regiones de Chile

## See Also

comunas(), codigos_comunas()

## Examples

```
regiones()
```

---

territorios *Tabla de comunas, provincias y regiones de Chile*

---

## Description

Tabla de datos que contiene todas las comunas del país con sus nombres oficiales, sus códigos únicos comunales, y lo mismo para las provincias y regiones del país. Al ser una tabla de las comunas de Chile, tiene 346 filas. Estos datos corresponden al Ministerio del Interior de Chile, publicados en el Diario Oficial el 21 de septiembre de 2018, con leves modificaciones.

## Usage

```
territorios
```

## Format

Un data frame con 346 filas y 6 columnas:

**codigo_region** Código único territorial de las regiones de Chile (número entre 1 y 16)

**nombre_region** Nombre de las regiones de Chile

**codigo_provincia** Código único territorial de las provincias de Chile

**nombre_provincia** Nombre de las provincias de Chile

**codigo_comuna** Código único territorial de las comunas de Chile

**nombre_comuna** Nombre de las comunas de Chile

## Details

Las modificaciones realizadas a los datos son: formas alternativas de escribir las comunas de Treguaco (Trehuaco) y Paiguano (Paihuano), optando por las formas que son usadas por sus municipios (Trehuaco y Paihuano, respectivamente), y comunas que deberían tener tilde (Los Ángeles, entre otras).

## Source

https://www.subdere.gov.cl/documentacion/cÃşdigos-Ãžnicos-territoriales-actualizados-al-06-de-septi

## See Also

comunas(), regiones() y codigos_comunas() para vectores derivados de esta tabla, y contextualizar() para agregar sus columnas a tus datos.

## trivia *Jugar a una trivia de comunas y regiones de Chile*

### Description

Inicia una trivia interactiva en la consola para probar tu conocimientos sobre los territorios de Chile! Aparecerán 7 preguntas en la consola, una por una, y debes responder con el número de la alternativa correspondiente. Al final aparecerá tu puntaje! Son 5 tipos de preguntas distintas, y los territorios que aparecen son aleatorios, mientras que las alternativas están diseñadas para confundirte!

### Usage

```
trivia()
```

### Details

Las preguntas son: *¿En qué región está la comuna? ¿qué comuna está en la región? ¿Cuál de estas comunas es urbana/rural? ¿En qué provincia está la comuna?*

También puedes jugar la trivia en tu navegador, sin instalar R.

### Examples

```
trivia()
```

## ubicar_comunas *Ubicar comunas en la región que les corresponde*

### Description

Entregando códigos comunales (como los que aparecen en territorios) o nombres de comunas, retorna los nombres de las regiones donde se ubican dichas comunas. Retorna NA si no corresponde con ninguna.

### Usage

```
ubicar_comunas(codigo_comuna)
```

### Arguments

| | |
|---|---|
| codigo_comuna | Códigos comunales en formato numérico, o nombres de comunas (como en comunas()) |

### Value

Vector con nombres de regiones correspondientes

## See Also

obtener_comunas() para hacer lo inverso, contextualizar() para agregar región, provincia y códigos a una tabla, y ubicar_localidades() para territorios más pequeños que una comuna.

## Examples

```
ubicar_comunas(c("Cerrillos", "Navidad"))

ubicar_comunas(c(1101, 13103))
```

---

# ubicar_localidades *Ubicar localidades en la comuna que les corresponde*

---

## Description

Al igual que ubicar_comunas(), para un vector de nombres de localidades, retorna las comunas donde se ubican dichas localidades. Las localidades son zonas geográficas, grupos de viviendas o asentamientos más pequeños que una comuna y que poseen un nombre propio, pero que no son entidades administrativas (para más información, ver la tabla localidades).

## Usage

```
ubicar_localidades(
  nombre_localidad = NULL,
  nombre_region = NULL,
  procedimiento = TRUE
)
```

## Arguments

| | |
|---|---|
| nombre_localidad | Nombres de localidades, idealmente como aparecen en localidades. |
| nombre_region | Opcionalmente, una región para circunscribir la búsqueda y así retornar mejores resultados. |
| procedimiento | Por defecto, muestra una tabla con el resultado del proceso de limpieza. Cambiar a FALSE para ocultar. |

## Details

Al buscar las localidades, primero busca por coincidencias exactas de nombres y, si existen más de una con el mismo nombre, retorna la localidad con mayor población (según el estudio de Localidades Aisladas 2021 de Subdere). Si no se encuentra una localidad exacta, se buscan por coincidencia inexacta.

## Value

Vector con nombres de comunas correspondientes

## See Also

localidades para la tabla completa de localidades, `ubicar_comunas()` para obtener la región de una comuna, y `buscar_comuna()` para buscar comunas por similitud.

## Examples

```
ubicar_localidades("La Boca")

ubicar_localidades(c("Chada", "La Dormida", "Mallarauco"))

ubicar_localidades("Alfalfal")

# cuando se conoce el nombre de la región, se pueden mejorar los resultados:
ubicar_localidades("Alfalfal", nombre_region = "Metropolitana de Santiago")
```

---

# validar_comunas *Validación de calidad de nombres de comunas de Chile*

---

## Description

Esta función recibe una variable con nombres de comunas de un dataframe, o un vector con nombres de comunas, y retorna una evaluación de posibles problemas con los nombres existentes.

## Usage

```
validar_comunas(datos, variable = NULL)
```

## Arguments

| | |
|---|---|
| `datos` | Dataframe con una columna de nombre de comunas, o vector de nombres de comunas |
| `variable` | Columna del dataframe con los nombres de comunas (se pasa sin comillas, p.ej. `comuna`) |

## Details

Funciona tanto con un dataframe (si la columna se llama `nombre_comuna` no es necesario especificarla), o un vector que contenga los nombres de comunas a evaluar. La función solamente retorna avisos cuando existan problemas, y retorna los datos de manera invisible.

## Value

Dataframe o vector intacto pero en modo invisible, con mensajes de diagnóstico si se encuentran problemas de calidad

## See Also

`limpiar_comunas()` para corregir automáticamente los problemas detectados, y `contar_comunas()` para revisar si faltan comunas en los datos. Para regiones, usa `validar_regiones()`.

## Examples

```
validar_comunas(c("chiguayante", "la florida", "paine"))

territorios |>
  validar_comunas(nombre_comuna)

# si ya existe una columna `nombre_comuna`, puede omitirse el argumento
territorios |>
  validar_comunas()
```

---

`validar_regiones` *Validación de calidad de nombres de regiones de Chile*

---

## Description

Esta función recibe la columna con nombres de regiones de un dataframe (idealmente `nombre_region`), y retorna una evaluación de posibles problemas con los nombres existentes. Funciona tanto con un dataframe con una columna `nombre_region`, o un vector que contenga los nombres de regiones a evaluar. La función solamente retorna avisos cuando existan problemas, por lo que si todos los datos son correctos, solo devolverá los datos tal cual.

## Usage

```
validar_regiones(datos, variable = NULL)
```

## Arguments

| | |
|---|---|
| `datos` | Dataframe con una columna de nombre de regiones, o vector de nombres de regiones |
| `variable` | Columna del dataframe con los nombres de regiones (se pasa sin comillas, p.ej. `region`) |

## Value

Dataframe o vector intacto pero en modo invisible, con mensajes de diagnóstico si se encuentran problemas de calidad

## See Also

`limpiar_regiones()` para corregir automáticamente los problemas detectados. Para comunas, usa `validar_comunas()`.

## Examples

```
validar_regiones(c("los lagos", "nuble", "OHIGGINS"))

territorial::territorios |>
  validar_regiones(nombre_region)
```

# Index


∗ **datasets**
clasificacion, 15
cut_historicos, 19
localidades, 30
territorios, 36

abreviar_comunas, 3
abreviar_comunas(), *4*, *34*
acortar_regiones, 4
acortar_regiones(), *3*, *12*, *27–29*, *34*
actualizar_codigo_comuna, 5
actualizar_codigo_comuna(), *12*, *19*, *22*, *23*
agregar_clasificacion, 6
agregar_clasificacion(), *8*, *15*
agregar_macrozona, 7
agregar_macrozona(), *6*, *8*, *32*
agregar_orden_region, 8
agregar_orden_region(), *32*
as_capital_regional, 9
as_capital_regional(), *21*
as_codigo_comuna, 10
as_codigo_comuna(), *6*, *11*, *12*, *16*, *18*, *21*, *27*
as_codigo_region, 11
as_codigo_region(), *7*, *8*, *10*, *12*, *24*, *32*
as_nombre_comuna, 11
as_nombre_comuna(), *10*, *22*
as_nombre_region, 12
as_nombre_region(), *9*, *11*

buscar_comuna, 13
buscar_comuna(), *39*

capitales_regionales, 14
capitales_regionales(), *9*, *21*
clasificacion, *6*, 15
codigos_comunas, 16
codigos_comunas(), *16*, *27*, *36*
comunas, 16
comunas(), *3*, *10*, *14*, *16*, *21*, *24–27*, *31*, *36*, *37*
contar_comunas, 17
contar_comunas(), *39*
contextualizar, 18
contextualizar(), *5*, *6*, *8*, *18*, *27*, *32*, *36*, *38*
cut_historicos, *5*, 19, *23*

eliminar_texto, 20
eliminar_texto(), *21*, *35*

insertar_texto, 20
insertar_texto(), *20*, *35*
is_capital_regional, 21
is_capital_regional(), *9*
is_codigo_comuna, 22
is_codigo_comuna(), *12*, *21*, *23*, *24*
is_codigo_comuna_antiguo, 23
is_codigo_comuna_antiguo(), *5*, *19*, *22*
is_codigo_region, 23
is_nombre_comuna, 24
is_nombre_comuna(), *13*, *25*
is_nombre_region, 25
is_nombre_region(), *24*

limpiar_comunas, 25
limpiar_comunas(), *10*, *13*, *17*, *18*, *20*, *21*, *24*, *29*, *35*, *39*
limpiar_regiones, 27
limpiar_regiones(), *11*, *25*, *29*, *40*
limpiar_texto, 29
localidades, 30, *38*, *39*

obtener_comunas, 31
obtener_comunas(), *17*, *38*
ordenar_regiones, 31
ordenar_regiones(), *8*

preposicion_region, 33
preposicion_region(), *34*

redactar_comunas, 33
redactar_comunas(), *3*, *17*

redactar_region, 34
redactar_region(), *4*, *12*, *29*, *33, 34*
reemplazar_texto, 35
reemplazar_texto(), *20, 21*
regiones, 35
regiones(), *9*, *14*, *16*, *27–29*, *31*, *33, 34*, *36*

territorios, *9–14*, *16*, *22, 23*, *25*, *31*, *35*, 36, *37*
trivia, 37

ubicar_comunas, 37
ubicar_comunas(), *31*, *38, 39*
ubicar_localidades, 38
ubicar_localidades(), *13*, *30*, *38*

validar_comunas, 39
validar_comunas(), *17, 18*, *24*, *27*, *40*
validar_regiones, 40
validar_regiones(), *25*, *29*, *39*