Package 'clc'

May 8, 2026

Type Package

Title CORINE Land Cover Data and Styles

Version 1.0.0

Description Streamline the management, analysis, and visualization of CORINE Land Cover data. Addresses challenges associated with its classification system and related styles, such as color mappings and descriptive labels.

License MIT + file LICENSE

URL https://josesamos.github.io/clc/, https://github.com/josesamos/clc

BugReports https://github.com/josesamos/clc/issues

Depends R (>= 3.5)

Imports gdalUtilities, ggplot2, rlang, RPostgres, sf, stats, terra, xml2

Suggests knitr, mockery, purrr, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

NeedsCompilation no

Author Jose Samos [aut, cre] (ORCID: <https://orcid.org/0000-0002-4457-3439>), Universidad de Granada [cph]

Maintainer Jose Samos <jsamos@ugr.es>

Repository CRAN

Date/Publication 2024-12-10 21:10:06 UTC

Contents

`as_raster` *Convert a 'clc' Object to Raster Format*

Description

Returns an object of class 'clc_raster' that contains a 'terra::SpatRaster' raster object representing the converted vector layer into raster format.

Usage

```
as_raster(clo, base_raster, resolution)

## S3 method for class 'clc'
as_raster(clo, base_raster = NULL, resolution = NULL)
```

Arguments

`clo`	A 'clc' object.
`base_raster`	(Optional) A raster object to use as the base for rasterization.
`resolution`	(Optional) Numeric resolution to define the raster grid.

Details

The function requires either 'base_raster' or 'resolution' to be provided. If both are missing, an error is raised.

Value

An object of class 'clc_raster'.

See Also

Other CLC class functions: clc(), copy_to(), cut_to_extent(), get_colors.clc(), get_levels.clc(), get_raster(), plot_clc(), prepare_plot(), save_to()

Examples

```
source_gpkg <- system.file("extdata", "clc.gpkg", package = "clc")
clo <- clc(source = source_gpkg, layer_name = "clc")

raster_path <- system.file("extdata", "mdt.tif", package = "clc")
base_raster <- terra::rast(raster_path)

# ex1
r <- clo |>
     as_raster(base_raster = base_raster)

# ex2
r <- clo |>
     as_raster(resolution = 50)
```

---

clc *'clc' S3 Class*

Description

Create an object of class 'clc'.

Usage

```
clc(source, layer_name, field = NULL)
```

Arguments

source	The source of the vector layer. This can be: - A string representing the path to a GeoPackage file. - A 'DBI' database connection object to a PostGIS database, created using [RPostgres::dbConnect()].
layer_name	The name of the layer in the source to be used.
field	(Optional) A string, the layer field that contains CLC codes. If NULL, the function will attempt to locate the column containing the CLC codes.

Details

This function creates an object of class 'clc' from a vector layer in either a GeoPackage or a PostGIS database.

The layer must have a style defined in the source.

Value

An object of class ‘clc‘.

See Also

Other CLC class functions: as_raster(), copy_to(), cut_to_extent(), get_colors.clc(), get_levels.clc(), get_raster(), plot_clc(), prepare_plot(), save_to()

Examples

```
# ex1
source_gpkg <- system.file("extdata", "clc.gpkg", package = "clc")
clo <- clc(source = source_gpkg, layer_name = "clc")

## Not run:
# ex2
conn <- RPostgres::dbConnect(
  RPostgres::Postgres(),
  dbname = 'exampledb',
  host = 'localhost',
  port = '5432',
  user = 'user',
  password = 'password'
)
clo <- clc(source = conn, layer_name = "clc")

## End(Not run)
```

---

clc_codes *CLC Codes*

Description

Each code represents a specific category at the most detailed level (Level 3) of the CLC system.

Usage

```
clc_codes
```

Format

A vector of strings.

copy_to *Copy a Style to a GeoPackage or PostGIS Database*

Description

This function copies a style to the specified layers in a GeoPackage file or a PostGIS database. The destination is determined by the ‘to‘ argument.

Usage

```
copy_to(clo, to, database, schema, layers)

## S3 method for class 'clc'
copy_to(clo, to, database = NULL, schema = "public", layers = NULL)
```

Arguments

clo	A ‘clc‘ object.
to	A data destination for the output. This can be: - A string representing the path to a GeoPackage file. - A ‘DBI‘ database connection object to a PostGIS database, created using [RPostgres::dbConnect()].
database	A string, database name, only in case the destination is in PostGIS.
schema	A string, schema name, only in case the destination is in PostGIS. Defaults to ‘’public’‘.
layers	An optional character vector specifying the names of layers in the destination to which the styles should be applied. If ‘NULL‘ (default), applies the style to all layers.

Value

clo A ‘clc‘ object.

See Also

Other CLC class functions: as_raster(), clc(), cut_to_extent(), get_colors.clc(), get_levels.clc(), get_raster(), plot_clc(), prepare_plot(), save_to()

Examples

```
source_gpkg <- system.file("extdata", "clc.gpkg", package = "clc")
clo <- clc(source = source_gpkg, layer_name = "clc")

out_gpkg <- tempfile(fileext = ".gpkg")
clo <- clo |>
  save_to(out_gpkg)

# ex1
```

```
clo <- clo |>
  copy_to(out_gpkg, layers = 'clc')

## Not run:
conn <- RPostgres::dbConnect(
  RPostgres::Postgres(),
  dbname = 'exampledb',
  host = 'localhost',
  port = '5432',
  user = 'user',
  password = 'password'
)
clo <- clo |>
  save_to(conn, 'exampledb')

# ex2
clo <- clo |>
  copy_to(conn, 'exampledb', layers = 'clc')

## End(Not run)
```

`cut_to_extent` *Clip the Layer with a Polygon*

Description

This function clips the object layer using a polygon layer. It handles CRS transformations automatically if necessary, ensuring the output is in the same CRS as the input polygon.

Usage

```
cut_to_extent(clo, polygon)

## S3 method for class 'clc'
cut_to_extent(clo, polygon)
```

Arguments

`clo`	A ‘clc‘ object.
`polygon`	An ‘sf‘ object representing the polygon layer used for clipping.

Value

A ‘clc‘ object.

See Also

Other CLC class functions: `as_raster()`, `clc()`, `copy_to()`, `get_colors.clc()`, `get_levels.clc()`, `get_raster()`, `plot_clc()`, `prepare_plot()`, `save_to()`

Examples

```
source_gpkg <- system.file("extdata", "clc.gpkg", package = "clc")
clo <- clc(source = source_gpkg, layer_name = "clc")

polygon <- sf::st_read(source_gpkg, layer = 'lanjaron', quiet = TRUE)

clo2 <- clo |>
        cut_to_extent(polygon)
```

`get_colors.clc` *Retrieve Colors from a CLC Style Definition*

Description

This function extracts the color values associated with a CLC style definition. It returns a character vector containing the ‘color‘ field from the CLC style definition.

Usage

```
## S3 method for class 'clc'
get_colors(clo)

get_colors(clo)

## S3 method for class 'clc_category'
get_colors(clo)

## S3 method for class 'clc_raster'
get_colors(clo)
```

Arguments

`clo`	A ‘clc_category‘ object.

Value

A character vector of colors.

See Also

Other CLC class functions: `as_raster()`, `clc()`, `copy_to()`, `cut_to_extent()`, `get_levels.clc()`, `get_raster()`, `plot_clc()`, `prepare_plot()`, `save_to()`

Examples

```
source_gpkg <- system.file("extdata", "clc.gpkg", package = "clc")
clo <- clc(source = source_gpkg, layer_name = "clc")

levels <- clo |>
  get_colors()
```

`get_levels.clc` *Retrieve Levels from a CLC Style Definition*

Description

This function extracts the levels values associated with a CLC style definition. It returns a data frame that contains the fields ‘id‘, ‘description‘, and ‘color‘ from the CLC style definition.

Usage

```
## S3 method for class 'clc'
get_levels(clo)

get_levels(clo)

## S3 method for class 'clc_category'
get_levels(clo)

## S3 method for class 'clc_raster'
get_levels(clo)
```

Arguments

`clo`	A CLC object.

Value

A data frame with columns: - ‘id‘: The identifier of the category. - ‘description‘: A textual description of the category. - ‘color‘: The color associated with the category.

See Also

Other CLC class functions: `as_raster()`, `clc()`, `copy_to()`, `cut_to_extent()`, `get_colors.clc()`, `get_raster()`, `plot_clc()`, `prepare_plot()`, `save_to()`

Examples

```
source_gpkg <- system.file("extdata", "clc.gpkg", package = "clc")
clo <- clc(source = source_gpkg, layer_name = "clc")

levels <- clo |>
  get_levels()
```

`get_raster` *Retrieve a Raster Representation of CLC*

Description

Retrieve a raster representation (‘terra::SpatRaster‘) from a CLC object.

Usage

```
get_raster(clo)

## S3 method for class 'clc_raster'
get_raster(clo)
```

Arguments

`clo`	A ‘clc_raster‘ object.

Value

A ‘terra::SpatRaster‘ object.

See Also

Other CLC class functions: `as_raster()`, `clc()`, `copy_to()`, `cut_to_extent()`, `get_colors.clc()`, `get_levels.clc()`, `plot_clc()`, `prepare_plot()`, `save_to()`

Examples

```
source_gpkg <- system.file("extdata", "clc.gpkg", package = "clc")
clo <- clc(source = source_gpkg, layer_name = "clc")

r <- clo |>
     as_raster(resolution = 50)

clc_r <- r |>
         get_raster()
```

plot_clc *Plot CLC Layer*

Description

Plot CLC data stored in objects of supported classes. The function adapts the plot based on the class of the input data (vectorial or raster format).

Usage

```
plot_clc(clo, ...)

## S3 method for class 'clc'
plot_clc(clo, ...)

## S3 method for class 'clc_raster'
plot_clc(clo, ...)
```

Arguments

clo	An object containing CLC data. This must be an instance of a supported class, such as: - A vectorial CLC data object (e.g., ‘clc‘ object). - A raster CLC data object (e.g., ‘clc_raster‘).
...	Additional arguments passed to the ‘terra::plot‘ function.

Details

For the raster version, the ‘terra::plot‘ function is used with the ‘col‘ parameter configured, while all other parameters supported by the function can also be defined (using ‘...‘).

For the vector version, ‘ggplot2::ggplot‘ is used, and by using the ‘prepare_plot‘ function instead of this one (‘plot_clc‘), further customization can be applied as needed.

Value

A ‘ggplot2‘ object or a ‘terra‘ plot.

See Also

prepare_plot

Other CLC class functions: as_raster(), clc(), copy_to(), cut_to_extent(), get_colors.clc(), get_levels.clc(), get_raster(), prepare_plot(), save_to()

Examples

```
source_gpkg <- system.file("extdata", "clc.gpkg", package = "clc")
clo <- clc(source = source_gpkg, layer_name = "clc")

temp_file <- tempfile(fileext = ".png")
png(filename = temp_file, width = 800, height = 600)

clo |>
  plot_clc()

dev.off()
```

`prepare_plot` *Prepare a Plot for CLC Vectorial Data*

Description

Generates a 'ggplot2' object to visualize CLC Vectorial data. The function processes the data stored in a 'clc' object, ensuring that the codes field is mapped correctly to the categories and their associated styles.

Usage

```
prepare_plot(clo)

## S3 method for class 'clc'
prepare_plot(clo)
```

Arguments

`clo`	A 'clc' object.

Value

A 'ggplot2' object ready for rendering.

See Also

`plot_clc`

Other CLC class functions: `as_raster()`, `clc()`, `copy_to()`, `cut_to_extent()`, `get_colors.clc()`, `get_levels.clc()`, `get_raster()`, `plot_clc()`, `save_to()`

Examples

```
source_gpkg <- system.file("extdata", "clc.gpkg", package = "clc")
clo <- clc(source = source_gpkg, layer_name = "clc")

p <- clo |>
  prepare_plot()

levels <- clo |>
  get_levels()

p <- p +
  ggplot2::scale_fill_manual(
    values = stats::setNames(levels$color, levels$id),
    labels = stats::setNames(levels$description, levels$id),
    name = ""
  ) +
  ggplot2::theme(
    legend.position = "right",
    legend.key.height = ggplot2::unit(2, "cm"),
    legend.title = ggplot2::element_text(size = 12),
    legend.text = ggplot2::element_text(size = 10)
  ) +
  ggplot2::theme_minimal()

temp_file <- tempfile(fileext = ".png")
png(filename = temp_file, width = 800, height = 600)

p

dev.off()
```

---

safe_clip_multipoligon

Safely Clip a Multipolygon Vector Layer

Description

This function clips a 'MULTIPOLYGON' vector layer using a polygon layer, handling specific issues that might arise with geometries encoded incorrectly or containing unknown WKB types. It serves as a fallback when the 'clip_vector()' function fails due to errors like 'ParseException: Unknown WKB type 12', which is associated with *MULTIPOLYGON* types.

Usage

```
safe_clip_multipoligon(vector, polygon)
```

Arguments

vector	A ‘sf‘ multipolygon vector layer to be clipped.
polygon	A ‘sf‘ polygon layer used as the clipping geometry.

Details

The function ensures that the input layer is correctly encoded as ‘MULTIPOLYGON‘ and uses GDAL utilities for re-encoding if necessary. The output is projected to the CRS of the clipping polygon.

This solution is inspired by a discussion on handling WKB type errors in R: <https://gis.stackexchange.com/questions/389814
st-centroid-geos-error-unknown-wkb-type-12>.

Value

A ‘sf‘ vector layer with the clipped geometries.

Examples

```
gpkg_path <- system.file("extdata", "clc.gpkg", package = "clc")

clc <- sf::st_read(gpkg_path, layer = "clc")
lanjaron <- sf::st_read(gpkg_path, layer = "lanjaron")

clc_clipped <- safe_clip_multipoligon(clc, lanjaron)
```

---

save_to *Save a Layer and its Style to a GeoPackage or PostGIS Database*

Description

This function saves a layer and its style to a GeoPackage file or a PostGIS database. The destination is determined by the ‘to‘ argument.

Usage

```
save_to(clo, to, database, schema, layer_name)

## S3 method for class 'clc'
save_to(clo, to, database = NULL, schema = "public", layer_name = NULL)
```

Arguments

clo	A ‘clc‘ object.
to	A data destination for the output. This can be: - A string representing the path to a GeoPackage file. - A ‘DBI‘ database connection object to a PostGIS database, created using [RPostgres::dbConnect()].
database	A string, database name, only in case the destination is in PostGIS.
schema	A string, schema name, only in case the destination is in PostGIS. Defaults to ‘’public’‘.
layer_name	A character string specifying the name of the layer in the output. If ‘NULL‘, the name of the input ‘layer‘ is used.

Details

The function overwrites the table if it already exists.

Value

clo A ‘clc‘ object.

See Also

Other CLC class functions: as_raster(), clc(), copy_to(), cut_to_extent(), get_colors.clc(), get_levels.clc(), get_raster(), plot_clc(), prepare_plot()

Examples

```
source_gpkg <- system.file("extdata", "clc.gpkg", package = "clc")
clo <- clc(source = source_gpkg, layer_name = "clc")

# ex1
out_gpkg <- tempfile(fileext = ".gpkg")

sink(tempfile())
clo <- clo |>
  save_to(out_gpkg)
sink()

## Not run:
# ex2
conn <- RPostgres::dbConnect(
  RPostgres::Postgres(),
  dbname = 'exampledb',
  host = 'localhost',
  port = '5432',
  user = 'user',
  password = 'password'
)
clo <- clo |>
  save_to(conn, 'exampledb')
```

```
## End(Not run)
```

Index