Package 'crsuggest'

May 8, 2026

Type Package

Title Obtain Suggested Coordinate Reference System Information for Spatial Data

Version 0.4

Date 2022-07-06

Description Uses data from the 'EPSG' Registry to look up suitable coordinate reference system transformations for spatial datasets in R. Returns a data frame with 'CRS' codes that can be used for 'CRS' transformation and mapping projects. Please see the 'EPSG' Dataset Terms of Use at <https://epsg.org/terms-of-use.html> for more information.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

LazyDataCompression xz

Imports sf (>= 0.9), units, dplyr, purrr, mapview

Suggests mapboxapi, tigris, tidycensus, ggplot2

RoxygenNote 7.2.0

Depends R (>= 3.3)

NeedsCompilation no

Author Kyle Walker [aut, cre]

Maintainer Kyle Walker <kyle@walker-data.com>

Repository CRAN

Date/Publication 2022-07-06 17:00:02 UTC

Contents

`crsuggest` *Get information on suggested coordinate reference systems for spatial data*

Description

Uses data from the EPSG Registry to look up suitable coordinate reference system transformations for spatial datasets in R. Returns a data frame with CRS codes that can be used for CRS transformation and mapping projects. Please see the EPSG Dataset Terms of Use at https://epsg.org/terms-of-use.html for more information.

Author(s)

Kyle Walker

`crs_sf` *Dataset of area extent polygons with CRS information*

Description

Used by the `crs_suggest()` function to look up suitable coordinate reference systems for input spatial data. Terms of use are available at https://epsg.org/terms-of-use.html.

Usage

```
data(crs_sf)
```

Format

An object of class `sf` (inherits from `tbl_df`, `tbl`, `data.frame`) with 6144 rows and 7 columns.

Details

Dataset of area extent polygons that correspond to CRS codes

See Also

https://epsg.org/terms-of-use.html

guess_crs *Guess the CRS of a dataset that is missing CRS information*

Description

This function will "guess" possible coordinate reference systems for spatial data that are lacking a CRS definition. Input data, which must be of class `"sf"` (or which can be converted to sf) might be objects created from CSV files that use projected coordinates or objects created from shapefiles loaded with `sf::st_read()` that are missing .prj files. The function requires a "target location" that the user knows to be within the general area of the input dataset. It then identifies suitable coordinate reference systems for that area and "tests out" those CRSs for the input data by analyzing the distance between the dataset and the known location when that CRS is used. Those distances are returned by the function in a column `dist_km`; short distances represent better guesses for the CRS whereas longer distances suggest that the CRS wouldn't work.

Usage

```
guess_crs(input, target_location, units = NULL, n_return = 10, input_sf = NULL)
```

Arguments

`input`	A spatial dataset of class `"sf"`, `"Spatial*"`, `"RasterLayer"`, `"SpatVector"`, or `"SpatRaster"` in a projected coordinate reference system that is missing CRS information. For example, you may have loaded in a shapefile without a .prj file, or your input data has no CRS definition attached.
`target_location`	A coordinate pair of form `c(longitude, latitude)` or an address/location that you know is located within your input sf object. If the mapboxapi package is installed, you can supply a location name (e.g. an address or a city) instead of a coordinate pair.
`units`	If known, the units of your projected coordinate system (e.g. `"m"` for meters or `"us-ft"` for US feet). This is not required but will make the guesses more accurate.
`n_return`	The number of possible CRS choices to return; defaults to 10. A higher number than that may include CRS options that are unlikely to work with your data. Use the returned `dist_km` column to judge whether the CRS guess makes sense for your data.
`input_sf`	Deprecated; use `input` instead.

Value

A tibble of CRS guesses for your data, sorted in ascending order of distance between your target location and the input sf object's centroid when in that CRS.

Examples

```
## Not run:
library(crsuggest)
library(sf)
# An example data frame of projected coordinates with no CRS information included
locations <- data.frame(
  X = c(2312654.74514528, 2357493.02092003, 2398978.30047505, 2344378.47525209,
        2475776.26735713, 2493751.94421798, 2456797.1698781, 2448392.13089886,
        2319704.35367616, 2350119.25250331, 2449088.54659236, 2423774.3668849),
  Y = c(6966055.04531077, 6994256.06222144, 6951975.79788762, 6902972.35980149,
        6918178.81070276, 6977643.56941746, 7053989.26343385, 7024543.36487243,
        7015476.52061313, 6953350.28550116, 6945011.24615857, 6912284.16691977),
  id = 1:12
)

# Create an sf object but the CRS is not known
locations_sf <- st_as_sf(locations, coords = c("X", "Y"))

# Use `guess_crs()` to guess the CRS used for the coordinates along with a known coordinate
# in the area of interest
guesses <- guess_crs(locations_sf, target_location = c(-97.1071, 32.7356))
# Set the CRS of your data with the "best guess"
st_crs(locations_sf) <- 6584

## End(Not run)
```

`suggest_crs` *Suggest coordinate systems for an input spatial dataset*

Description

This function takes an input spatial dataset as input and makes "suggestions" for suitable coordinate reference systems that could be used for CRS transformations in spatial analysis projects. The function works by analyzing the extent of the spatial dataset and comparing it to the area extents in the EPSG's coordinate reference system database. The "suggested" coordinate reference systems are determined by minimizing the Hausdorff distances between the CRS area extents and the input dataset, subject to user preferences (such as a geographic coordinate system ID or measurement units).

Usage

```
suggest_crs(
  input,
  type = "projected",
  limit = 10,
  gcs = NULL,
  units = NULL,
  drop_na = TRUE
)
```

Arguments

input	A spatial dataset of class "sf", "Spatial*", "RasterLayer", "SpatVector", or "SpatRaster" for which you would like coordinate reference system suggestions.
type	The output CRS type; defaults to "projected".
limit	How many results to return; defaults to 10.
gcs	(optional) The EPSG code for the corresponding geographic coordinate system of the results (e.g. 4326 for WGS 1984).
units	(optional) The measurement units of the coordinate systems in the returned results. Can be one of "m", "ft", or "ft-us".
drop_na	Whether or not to drop EPSG codes that do not appear in the PROJ database (and thus can't be used for CRS transformation). Defauts to TRUE; set to FALSE if you want to search all codes.

Value

A data frame with information about coordinate reference systems that could be suitably used for CRS transformation.

Examples

```
## Not run:

library(tigris)
library(crsuggest)

# Get a dataset of Census tracts for Nassau County, NY
nassau_tracts <- tracts("NY", "Nassau", cb = TRUE)

# tigris datasets default to the NAD1983 GCS (EPSG code 4269)
# What are some appropriate projected coordinate systems?
suggest_crs(nassau_tracts)

# Alternatively, we can require projections to have specific
# geographic coordinate systems and/or units
# For example, let's say we only want NAD83(HARN) (code 4152)
# and we want the measurement units to be US feet
suggest_crs(nassau_tracts, gcs = 4152, units = "us-ft")


## End(Not run)
```

---

suggest_top_crs *Return the CRS code for a "best-fit" projected coordinate reference system*

Description

Return the EPSG code or proj4string syntax for the top-ranking projected coordinate reference system returned by `suggest_crs()`. This function should be used with caution and is recommended for interactive work rather than in production data pipelines.

Usage

```
suggest_top_crs(input, units = NULL, inherit_gcs = TRUE, output = "epsg")
```

Arguments

`input`	An input spatial dataset of class `"sf"`, `"Spatial*"`, or `"RasterLayer"`.
`units`	(optional) The measurement units used by the returned coordinate reference system.
`inherit_gcs`	if `TRUE` (the default), the function will return a CRS suggestion that uses the geographic coordinate system of the input layer. Otherwise, the output may use a different geographic coordinate system from the input.
`output`	one of `"epsg"`, for the EPSG code, or `"proj4string"`, for the proj4string syntax.

Value

The EPSG code or proj4string for the output coordinate reference system.

Examples

```
## Not run:

# Let's say we are working with a demographic dataset from the US Census:
library(tidycensus)
library(ggplot2)
library(sf)
library(crsuggest)

tx_income <- get_acs(
  geography = "county",
  variables = "B19013_001",
  state = "TX",
  geometry = TRUE
)

# We can use `suggest_top_crs()` to return the EPSG code of the "top" suggested CRS
# for statewide mapping of Texas
tx_crs <- suggest_top_crs(tx_income)

# The returned CRS is EPSG code 3083, NAD83 / Texas Centric Albers Equal Area.
# This code can be used for visualization:

ggplot(tx_income, aes(fill = estimate)) +
  geom_sf() +
```

```
  coord_sf(crs = tx_crs)

# Alternatively, we can transform the CRS of our sf object directly:

tx_projected <- st_transform(tx_income, tx_crs)

## End(Not run)
```

`view_crs` *Quickly preview the extent of a given CRS using mapview*

Description

The crsuggest package makes coordinate reference systems *suggestions* that may not be perfect for your specific analytic use case. Use `view_crs()` to quickly view the geographic extent of a given coordinate reference system (represented by its EPSG code) and assess whether that CRS makes sense for your data.

Usage

```
view_crs(crs)
```

Arguments

`crs`	A character string representing the EPSG code of an input coordinate reference system, possibly returned by `suggest_top_crs`.

Value

an object of class `mapview` which uses the mapview package to preview the extent of a coordinate reference system.

Examples

```
## Not run:

library(tigris)
library(crsuggest)
options(tigris_use_cache = TRUE)

# Get a Census tract dataset from the tigris package
tarrant_tracts <- tracts("TX", "Tarrant", cb = TRUE, year = 2021)

# Suggest a CRS for your data
target_crs <- suggest_top_crs(tarrant_tracts, units = "m", inherit_gcs = FALSE)

# Preview the extent of the CRS
view_crs(target_crs)


## End(Not run)
```

Index