Package 'conleyreg'

June 28, 2021

Type Package

Title Estimations using Conley Standard Errors

Version 0.1.4

Description Merges and extends multiple packages and other published scripts calculating Conley (1999) <doi:10.1016/S0304-4076(98)00084-0> standard errors. Details are available in the function documentation and in the vignette.

License MIT + file LICENSE

Encoding UTF-8

Imports base, stats, sf, Rcpp, RcppArmadillo, data.table, lfe, lmtest, foreach, parallel, doParallel, Rdpack, fixest, Matrix, lwgeom

Suggests rmarkdown, knitr, s2

LinkingTo Rcpp, RcppArmadillo

RdMacros Rdpack

SystemRequirements GNU make

RoxygenNote 7.1.1

VignetteBuilder knitr

NeedsCompilation yes

Author Christian Düben [aut, cre],
Richard Bluhm [cph],
Luis Calderon [cph],
Darin Christensen [cph],
Timothy Conley [cph],
Thiemo Fetzer [cph],
Leander Heldring [cph]

Maintainer Christian Düben <christian.dueben@uni-hamburg.de>

Repository CRAN

Date/Publication 2021-06-28 12:10:02 UTC

R topics documented:

`conleyreg` *Conley standard error estimations*

Description

This function estimates ols, logit, and probit models with Conley standard errors.

Usage

```
conleyreg(
  formula,
  data,
  dist_cutoff,
  model = c("ols", "logit", "probit"),
  unit = NULL,
  time = NULL,
  lat = NULL,
  lon = NULL,
  kernel = c("bartlett", "uniform"),
  lag_cutoff = 0,
  intercept = TRUE,
  verbose = TRUE,
  ncores = NULL,
  dist_comp = c("precise", "fast"),
  sparse = FALSE,
  batch = FALSE
)
```

Arguments

`formula`	regression equation as formula or character string
`data`	input data, either in non-spatial data frame format (includes tibbles and data tables) with columns denoting coordinates or in sf format with a spatial points geometry. When using a non-spatial data frame format, the coordinates must be longlat. sf objects can use any projection. Note that the projection can influence the computed distances, which is a general phenomenon in GIS software and not specific to `conleyreg`.
`dist_cutoff`	the distance cutoff in km
`model`	the applied model. Either `ols` (default), `logit`, or `probit`. `logit` and `probit` are currently restricted to cross-sectional applications.
`unit`	the variable identifying the cross-sectional dimension. Only needs to be specified, if data is not cross-sectional. Assumes that units do not change their location over time.
`time`	the variable identifying the time dimension

`lat`	the variable specifying the latitude in longlat format
`lon`	the variable specifying the longitude in longlat format
`kernel`	the kernel applied within the radius. Either `bartlett` (default) or `uniform`.
`lag_cutoff`	the cutoff along the time dimension. Defaults to 0, meaning that standard errors are only adjusted cross-sectionally.
`intercept`	boolean specifying whether to include an intercept. Defaults to `TRUE`. Fixed effects models omit the intercept automatically.
`verbose`	boolean specifying whether to print messages on intermediate estimation steps. Defaults to `TRUE`.
`ncores`	the number of CPU cores to use in the estimations. Defaults to the machine's number of CPUs. Does not affect cross-sectional applications.
`dist_comp`	choice between `precise` (default) and `fast` distance computations when data is longlat. Even when choosing `precise`, you can still tweak the performance by setting the library that the `sf` package uses in distance computations. `sf::sf_use_s2(TRUE)` makes it rely on s2 which should be faster than the alternative choice of GEOS with `sf::sf_use_s2(FALSE)`. With `precise`, distances are great circle distances, with `fast` they are haversine distances. Non-longlat data is not affected by this parameter and always uses Euclidean distances.
`sparse`	boolean specifying whether to use sparse rather than dense (regular) matrices in distance computations. Defaults to `FALSE`. Only has an effect when `dist_comp = "fast"`. Sparse matrices are more efficient than dense matrices, when the distance matrix has a lot of zeros arising from points located outside the respective `dist_cutoff`. It is recommended to keep the default unless the machine is unable to allocate enough memory.
`batch`	boolean specifying whether distances are inserted into a sparse matrix element by element (`FALSE`) or all at once as a batch (`TRUE`). Defaults to `FALSE`. This argument only has an effect when `dist_comp = "fast"` and `sparse = TRUE`. Batch insertion is faster than element-wise insertion, but requires more memory.

Details

This code is an extension and modification of earlier Conley standard error implementations by (i) Richard Bluhm, (ii) Luis Calderon and Leander Heldring, (iii) Darin Christensen and Thiemo Fetzer, and (iv) Timothy Conley. Results vary across implementations because of different distance functions and buffer shapes.

Value

Returns a `lmtest::coeftest` matrix of coefficient estimates and standard errors.

References

Calderon L, Heldring L (2020). "Spatial standard errors for several commonly used M-estimators." Mimeo.

Conley TG (1999). "GMM estimation with cross sectional dependence." *Journal of Econometrics*, **92**(1), 1-45.

Conley TG (2008). "Spatial Econometrics." In Durlauf SN, Blume LE (eds.), *Microeconometrics*, 303-313. London: Palgrave Macmillan.

Examples

```
# Generate cross-sectional example data
data <- data.frame(y = sample(c(0, 1), 100, replace = TRUE),
  x1 = stats::runif(100, -50, 50),
  lat = runif(100, -90, 90),
  lon = runif(100, -180, 180))

# Estimate ols model with Conley standard errors using a 1000 km radius
conleyreg(y ~ x1, data, 1000, lat = "lat", lon = "lon")

# Estimate same model with an sf object as input
conleyreg(y ~ x1, sf::st_as_sf(data, coords = c("lon", "lat"), crs = 4326), 1000)

# Estimate same model with an sf object of another projection as input
conleyreg(y ~ x1, sf::st_transform(sf::st_as_sf(data, coords = c("lon", "lat"), crs = 4326),
  crs = "+proj=aeqd"), 1000)

# Estimate logit model
conleyreg(y ~ x1, data, 1000, "logit", lat = "lat", lon = "lon")

# Add variable
data$x2 <- sample(1:5, 100, replace = TRUE)

# Estimate ols model with fixed effects
conleyreg(y ~ x1 | x2, data, 1000, lat = "lat", lon = "lon")

# Estimate probit model with fixed effects
conleyreg(y ~ x1 | x2, data, 1000, "probit", lat = "lat", lon = "lon")

# Add panel variables
data$time <- rep(1:10, each = 10)
data$unit <- rep(1:10, times = 10)

# Estimate ols model using panel data
conleyreg(y ~ x1, data, 1000, unit = "unit", time = "time", lat = "lat", lon = "lon")
```

Index