Package: conleyreg (via r-universe)

October 24, 2026

Type Package

Title Estimations using Conley Standard Errors

Version 0.1.9

Description Functions calculating Conley (1999) <doi:10.1016/S0304-4076(98)00084-0> standard errors. The package started by merging and extending multiple packages and other published scripts on this econometric technique. It strongly emphasizes computational optimization. Details are available in the function documentation and in the vignette.

License MIT + file LICENSE

Encoding UTF-8

URL https://github.com/cdueben/conleyreg

BugReports https://github.com/cdueben/conleyreg/issues

Imports stats, sf, Rcpp, data.table, lmtest, foreach, parallel, doParallel, Rdpack, fixest, Matrix, lwgeom, s2, methods

Suggests rmarkdown, knitr

LinkingTo Rcpp, RcppArmadillo

RdMacros Rdpack

SystemRequirements GNU make

RoxygenNote 7.3.3

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 <cdueben.ml+cran@proton.me>

Config/pak/sysreqs make

Repository https://cran.r-universe.dev

Date/Publication 2026-01-07 19:20:07 UTC

RemoteUrl https://github.com/cran/conleyreg

RemoteRef HEAD

RemoteSha 6c5be2df72f5fb97c9e5798021a7ce9b2c888e21

Contents

conleyreg 2
dist_mat 7
rnd_locations 10

Index **12**

`conleyreg` *Conley standard error estimations*

Description

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

Usage

```
conleyreg(
  formula,
  data,
  dist_cutoff,
  model = c("ols", "logit", "probit", "poisson"),
  unit = NULL,
  time = NULL,
  lat = NULL,
  lon = NULL,
  kernel = c("bartlett", "uniform"),
  lag_cutoff = 0,
  intercept = TRUE,
  verbose = TRUE,
  ncores = NULL,
  par_dim = c("cross-section", "time", "r", "cpp"),
  dist_comp = NULL,
  crs = NULL,
  st_distance = FALSE,
  dist_which = NULL,
  sparse = FALSE,
  batch = TRUE,
  batch_ram_opt = NULL,
  float = FALSE,
  rowwise = FALSE,
  reg_ram_opt = FALSE,
  dist_mat = NULL,
```

```
  dist_mat_conv = TRUE,
  vcov = FALSE,
  gof = FALSE
)
```

Arguments

formula	regression equation as formula or character string. Avoid interactions and transformations inside the equation. I.e. use `y ~ x1 + x1_2, data = dplyr::mutate(data, x1_2 = x1^2)` instead of `y ~ x1 + x1^2, data = data)`.
data	input data. Either (i) in non-spatial data frame format (includes tibbles and data tables) with columns denoting coordinates or (ii) in sf format. In case of an sf object, all non-point geometry types are converted to spatial points, based on the feature's centroid. When using a non-spatial data frame format with projected, i.e. non-longlat, coordinates, `crs` must be specified. Note that the projection can influence the computed distances, which is a general phenomenon in GIS software and not specific to `conleyreg`. The computationally fastest option is to use a data table with coordinates in the crs in which the distances are to be derived (longlat for spherical and projected for planar), and with time and unit set as keys in the panel case. An sf object as input is the slowest option.
dist_cutoff	the distance cutoff in km
model	the applied model. Either `"ols"` (default), `"logit"`, `"probit"` or `"poisson"`. `"logit"`, `"probit"`, and `"poisson"` 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. Only needs to be specified, if data is not cross-sectional.
lat	the variable specifying the latitude
lon	the variable specifying the longitude
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	logical specifying whether to include an intercept. Defaults to `TRUE`. Fixed effects models omit the intercept automatically.
verbose	logical 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.
par_dim	the dimension along which the function parallelizes in panel applications. Can be set to `"cross-section"` (default) or `"time"`. With the former option, the function parallelizes the spatial correlation code in C++ using OpenMP and the serial correlation part in R using the parallel package. With the latter option, it is the other way around. Use `"r"` and `"cpp"` to define parallelization based on

the language rather than the dimension. Some MAC users do not have access to OpenMP by default. par_dim is then always set to "r". Thus, depending on the application, the function can be notably faster on Windows and Linux than on MACs.

dist_comp — choice between "spherical" and "planar" distance computations. When unspecified, the input data determines the method: longlat uses spherical (Haversine) distances, alternatives (projected data) use planar (Euclidean) distances. When inserting projected data but specifying dist_comp = "spherical", the data is transformed to longlat. Combining unprojected data with dist_comp = "planar" transforms the data to an azimuthal equidistant format centered at the data's centroid.

crs — the coordinate reference system, if the data is projected. Object of class crs or input string to sf::st_crs. This parameter can be omitted, if the data is in longlat format (EPSG: 4326), i.e. not projected. If the projection does not use meters as units, this function converts to units to meters.

st_distance — logical specifying whether distances should be computed via sf::st_distance (TRUE) or via conleyreg's internal, computationally optimized distance functions (FALSE). The default (FALSE) produces the same distances as sf::st_distance does with S2 enabled. I.e. it uses Haversine (great circle) distances for longlat data and Euclidean distances otherwise. Cases in which you might want to set this argument to TRUE are e.g. when you want enforce the GEOS approach to computing distances or when you are using an peculiar projection, for which the sf package might include further procedures. Cross-sectional parallelization is not available when st_distance = TRUE and the function automatically switches to parallelization along the time dimension, if the data is a panel and ncores != 1. Third and fourth dimensions, termed Z and M in sf, are not accounted for in any case. Note that sf::st_distance is considerably slower than conleyreg's internal distance functions.

dist_which — the type of distance to use when st_distance = TRUE. If unspecified, the function defaults to great circle distances for longlat data and to Euclidean distances otherwise. See sf::st_distance for options.

sparse — logical specifying whether to use sparse rather than dense (regular) matrices in distance computations. Defaults to FALSE. Only has an effect when st_distance = FALSE. 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 — logical specifying whether distances are inserted into a sparse matrix element by element (FALSE) or all at once as a batch (TRUE). Defaults to TRUE. This argument only has an effect when st_distance = FALSE and sparse = TRUE. Batch insertion is faster than element-wise insertion, but requires more memory.

batch_ram_opt — the degree to which batch insertion should be optimized for RAM usage. Can be set to one out of the three levels: "none", "moderate" (default), and "heavy". Higher levels imply lower RAM usage, but also lower speeds.

float — logical specifying whether distance matrices should use the float (TRUE) rather than the double (FALSE) data type. Floats are less precise and than doubles and

thereby occupy less space than doubles do. They should only be used when the machine's RAM is insufficient for both the dense and the sparse matrix cases, as they affect the precision of distance values. The `float` option only has an effect in Bartlett kernel cases because uniform kernel applications store the data in a smaller integer data type.

`rowwise` logical specifying whether to store individual rows of the distance matrix only, instead of the full matrix. If `TRUE`, the function uses these rows directly in the standard error correction. This option's advantage is that it induces the function to store only N x `ncores` cells, instead of the full N x N matrix, lowering RAM requirements. The disadvantage is that the function needs to compute twice as many distance values as in the default case (`FALSE`), since the symmetry of the matrix is not utilized. It hence sacrifices speed for lower RAM utilization. This parameter only has an effect in cross-sectional and unbalanced panel applications with `st_distance = FALSE` and `sparse = FALSE`.

`reg_ram_opt` logical specifying whether the regression should be optimized for RAM usage. Defaults to `FALSE`. Changing it to `TRUE` slows down the function. This argument only affects the baseline estimation, not the standard error correction.

`dist_mat` a distance matrix. Pre-computing a distance matrix and passing it to this argument is only more efficient than having `conleyreg` derive it, if you execute `conleyreg` multiple times with the same input data. In that case, it is recommended to compute the distance matrix via `dist_mat`, which is optimized for this purpose and also evaluates various other steps that are identical across regressions on the same input data. Generally, you must not modify the input data between deriving the distance matrix and running `conleyreg`. That includes dropping observations or changing values of the unit, time, or coordinate variables. In cross-sectional settings, you must not re-order rows either. If you compute distances through a function other than `dist_mat`, there are a few additional issues to account for. (i) In the panel scenario, you must order observations by time and unit in ascending order. I.e. cells [1, 2] and [2, 1] of the distance matrix must refer to the distance between unit 1 and unit 2, cells [2, 3] and [3, 2] to the distance between unit 2 and unit 3 etc. The unit numbers in this example refer to their rank when they are sorted. (ii) `dist_cutoff` does not refer to kilometers, but to the units of the matrix. (iii) While in a balanced panel you only enter one matrix that is applied to all periods, you supply distances as a list of matrices in the unbalanced case. The matrices must be sorted, with the first list element containing the first period's distance matrix etc. (iv) Zeros in sparse matrices are interpreted as values above the distance cutoff and NaN values are interpreted as zeros. (v) The matrices in the list must all be of the same type - all dense or all sparse. (vi) Distance matrices must only contain non-missing, finite numbers (and NaN in the case of sparse matrices).

`dist_mat_conv` logical specifying whether to convert the distance matrix to a list, if the panel turns out to be unbalanced because of missing values. This setting is only relevant, if you enter a balanced panel's distance matrix not derived via `dist_mat`. If `TRUE` (the default), the function only drops rows with missing values. If `FALSE`, the function maintains the panel's balanced character by dropping units with missing values in at least one period from the entire data set.

`vcov` logical specifying whether to return variance-covariance matrix (`TRUE`) rather

than the default `lmtest::coeftest` matrix of coefficient estimates and standard errors (`FALSE`).

`gof` logical specifying whether to return goodness of fit measures. Defaults to `FALSE`. If `TRUE`, the function produces a list.

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.

This function has reasonable defaults. If your machine has insufficent RAM to allocate the default dense matrices, try sparse matrices. If the RAM error persists, try setting a lower `dist_cutoff`, use floats, select a uniform kernel, experiment with `batch_ram_opt`, `reg_ram_opt`, or `batch`.

Consult the vignette, `vignette("conleyreg_introduction", "conleyreg")`, for a more extensive discussion.

Value

Returns a `lmtest::coeftest` matrix of coefficient estimates and standard errors by default. Can be changed to the variance-covariance matrix by specifying `vcov = TRUE`.

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

```
## Not run:
# Generate cross-sectional example data
data <- rnd_locations(100, output_type = "data.frame")
data$y <- sample(c(0, 1), 100, replace = TRUE)
data$x1 <- stats::runif(100, -50, 50)

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

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

# Estimate ols model with fixed effects
data$x2 <- sample(1:5, 100, replace = TRUE)
conleyreg(y ~ x1 | x2, data, 1000, lat = "lat", lon = "lon")
```

```
# Estimate ols model using panel data
data$time <- rep(1:10, each = 10)
data$unit <- rep(1:10, times = 10)
conleyreg(y ~ x1, data, 1000, unit = "unit", time = "time", lat = "lat", lon = "lon")

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

## End(Not run)
```

`dist_mat` *Distance matrix estimation*

Description

This function estimates the distance matrix separately from Conley standard errors. Such step can be helpful when running multiple Conley standard error estimations based on the same distance matrix. A pre-requisite of using this function is that the data must not be modified between applying this function and inserting the results into `conleyreg`.

Usage

```
dist_mat(
  data,
  unit = NULL,
  time = NULL,
  lat = NULL,
  lon = NULL,
  dist_comp = NULL,
  dist_cutoff = NULL,
  crs = NULL,
  verbose = TRUE,
  ncores = NULL,
  par_dim = c("cross-section", "time", "r", "cpp"),
  sparse = FALSE,
  batch = TRUE,
  batch_ram_opt = NULL,
  dist_round = FALSE,
  st_distance = FALSE,
  dist_which = NULL
)
```

Arguments

data	input data. Either (i) in non-spatial data frame format (includes tibbles and data tables) with columns denoting coordinates or (ii) in sf format. In case of an sf object, all non-point geometry types are converted to spatial points, based on the feature's centroid. When using a non-spatial data frame format the with projected, i.e. non-longlat, coordinates, `crs` must be specified. Note that the projection can influence the computed distances, which is a general phenomenon in GIS software and not specific to conleyreg. The computationally fastest option is to use a data table with coordinates in the crs in which the distances are to be derived (longlat for spherical and projected for planar), and with time and unit set as keys in the panel case. An sf object as input is the slowest option.
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. Only needs to be specified, if data is not cross-sectional.
lat	the variable specifying the latitude
lon	the variable specifying the longitude
dist_comp	choice between `spherical` and `planar` distance computations. When unspecified, the input data determines the method: longlat uses spherical (Haversine) distances, alternatives (projected data) use planar (Euclidean) distances. When inserting projected data but specifying `dist_comp = "spherical"`, the data is transformed to longlat. Combining unprojected data with `dist_comp = "planar"` transforms the data to an azimuthal equidistant format centered at the data's centroid.
dist_cutoff	the distance cutoff in km. If not specified, the distances matrices contain all bilateral distances. If specified, the cutoff most be as least as large as the largest distance cutoff in the Conley standard error corrections in which you use the resulting matrix. If you e.g. specify distance cutoffs of 100, 200, and 500 km in the subsequent `conleyreg` calls, `dist_cutoff` in this function must be set to at least 500. `dist_cutoff` allows to pre-compute distance matrices also in applications where a full distance matrix would not fit into the computer's memory - conditional on that `sparse = TRUE`.
crs	the coordinate reference system, if the data is projected. Object of class crs or input string to `sf::st_crs`. This parameter can be omitted, if the data is in longlat format (EPSG: 4326), i.e. not projected. If the projection does not use meters as units, this function converts to units to meters.
verbose	logical 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.
par_dim	the dimension along which the function parallelizes in unbalanced panel applications. Can be set to `"cross-section"` (default) or `"time"`. Use `"r"` and `"cpp"` to define parallelization based on the language rather than the dimension. In this function, `"r"` is equivalent to `"time"` and parallelizes along the time dimension

using the parallel package. "cross-section" is equivalent to "cpp" and parallelizes along the cross-sectional dimension using OpenMP in C++. Some MAC users do not have access to OpenMP by default. par_dim is then always set to "r". Thus, depending on the application, the function can be notably faster on Windows and Linux than on MACs. When st_distance = TRUE, par_dim defaults to "time".

sparse — logical specifying whether to use sparse rather than dense (regular) matrices in distance computations. Defaults to FALSE. Only has an effect when st_distance = FALSE. 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. The function always uses dense matrices when dist_cutoff is not specified.

batch — logical 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 st_distance = FALSE and sparse = TRUE. Batch insertion is faster than element-wise insertion, but requires more memory.

batch_ram_opt — the degree to which batch insertion should be optimized for RAM usage. Can be set to one out of the three levels: "none", "moderate" (default), and "heavy". Higher levels imply lower RAM usage, but also lower speeds.

dist_round — logical specifying whether to round distances to full kilometers. This further reduces memory consumption and can be a solution when even sparse matrices cannot accomodate the data. Note, though, that this rounding introduces a bias.

st_distance — logical specifying whether distances should be computed via sf::st_distance (TRUE) or via conleyreg's internal, computationally optimized distance functions (FALSE). The default (FALSE) produces the same distances as sf::st_distance does with S2 enabled. I.e. it uses Haversine (great circle) distances for longlat data and Euclidean distances otherwise. Cases in which you might want to set this argument to TRUE are e.g. when you want enforce the GEOS approach to computing distances or when you are using a peculiar projection, for which the sf package might include further procedures. Cross-sectional parallelization is not available when st_distance = TRUE and the function automatically switches to parallelization along the time dimension, if the data is a panel and ncores != 1. Third and fourth dimensions, termed Z and M in sf, are not accounted for in any case. Note that sf::st_distance is considerably slower than conleyreg's internal distance functions.

dist_which — the type of distance to use when st_distance = TRUE. If unspecified, the function defaults to great circle distances for longlat data and to Euclidean distances otherwise. See sf::st_distance for options.

Details

This function runs the distance matrix estimations separately from the Conley standard error correction. You can pass the resulting object to the dist_mat argument in conleyreg, skipping the distance matrix computations and various checks in that function. Pre-computing the distance matrix is only more efficient than deriving it via conleyreg when estimating various models that use the same distance matrices. The input data must not be modified between calling this function and

inserting the results into conleyreg. Do not reorder the observations, add or delete variables, or undertake any other operation on the data.

Value

Returns an object of S3 class conley_dist. It contains modified distance matrices, the used dist_cutoff, a sparse matrix identifier, and information on the potential panel structure. In the cross-sectional case and the balanced panel case, the distances are stored in one matrix, while in unbalanced panel applications, distances come as a list of matrices. The function optimizes the distance matrices with respect to computational performance, setting distances beyond dist_cutoff to zero and actual off-diagonal zeros to NaN. Hence, these objects are only to be used in conleyreg.

Examples

```
## Not run:
# Generate cross-sectional example data
data <- rnd_locations(100, output_type = "data.frame")
data$y <- sample(c(0, 1), 100, replace = TRUE)
data$x1 <- stats::runif(100, -50, 50)

# Compute distance matrix in cross-sectional case
dm <- dist_mat(data, lat = "lat", lon = "lon")

# Compute distance matrix in panel case
data$time <- rep(1:10, each = 10)
data$unit <- rep(1:10, times = 10)
dm <- dist_mat(data, unit = "unit", time = "time", lat = "lat", lon = "lon")

# Use distance matrix in conleyreg function
conleyreg(y ~ x1, data, 1000, dist_mat = dm)

## End(Not run)
```

---

rnd_locations *Random location drawing*

Description

This function draws random locations in longlat format.

Usage

```
rnd_locations(
  nobs,
  xmin = -180,
  xmax = 180,
  ymin = -90,
  ymax = 90,
```

```
  output_type = c("data.table", "data.frame", "sf")
)
```

Arguments

nobs	number of observations
xmin	minimum longitude
xmax	maximum longitude
ymin	minimum latitude
ymax	maximum latitude
output_type	type of output object. Either "data.table" (default), "data.frame", or "sf".

Details

By default, this function draws a global sample of random locations. You can restrict it to a certain region via specifying xmin, xmax, ymin, and ymax. The function draws from a uniform spatial distribution that assumes the planet to be a perfect sphere. The spherical assumption is common in GIS functions, but deviates from Earth's exact shape.

Value

Returns a data.table, data.frame, or sf object of unprojected (longlat) points.

Examples

```
data <- rnd_locations(1000)
```

Index

conleyreg, 2

dist_mat, *5*, 7

rnd_locations, 10