Package: dynCorr (via r-universe)

September 17, 2024

Title Dynamic Correlation Package

Version 1.1.0

Date 2017-12-10

Description Computes dynamical correlation estimates and percentile bootstrap confidence intervals for pairs of longitudinal responses, including consideration of lags and derivatives.

Depends lpridge, R (>= 2.10), stats

License GPL-2

Encoding UTF-8

LazyData true

RoxygenNote 6.0.1

Suggests knitr, rmarkdown

NeedsCompilation no

Author Joel Dubin [aut, cre], Mike Li [aut], Dandi Qiao [aut], Hans-Georg Müller [aut]

Maintainer Joel Dubin <jdubin@uwaterloo.ca>

Date/Publication 2017-12-10 21:54:38 UTC

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

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

RemoteRef HEAD

RemoteSha c20b6b6e8fa56385a474b901bcffb9f96b305aed

Contents

bootstrapCI *Bootstrap Confidence Interval*

Description

Computes percentile bootstrap (BS) confidence intervals for dynamical correlation for pairs of longitudinal responses, including consideration of lags and derivatives, following a local polynomial regression smoothing step.

Usage

```
bootstrapCI(dataFrame, depVar, indepVar, subjectVar, function.choice,
            width.range, width.place, min.obs, points.length, points.by,
            boundary.trunc, byOrder, max.dynCorrLag, B, percentile,
            by.deriv.only, seed)
```

Arguments

`dataFrame`	The data frame that contains the dependent variables/responses, the independent variable (often time), and the subject/individual identification; there should be one row entry for each combination of subject/individual and indepVar (often time).
`depVar`	Dependent variables/responses; at least two are necessary for purposes of calculating at least one dynamical correlation estimate of interest; there should be a unique column within depVar for each response.
`indepVar`	Independent variable, typically the discrete recorded time points at which the dependent variables were collected; note that this is the independent variable for purposes of curve creation leading into estimating the dynamical correlations between pairs of dependent variables; must be contained in a single column.
`subjectVar`	Column name of the individuals; there should be one row entry for each combination of subject/individual and indepVar.
`function.choice`	A vector of length 3 to indicate which derivatives desired for local polynomial curve smoothing; 1st entry is for 0th derivative (i.e., original function), 2nd entry is for 1st derivative, 3rd is for 2nd derivative; 1=yes, 0=no. e.g., c(1,0,1) would be specified if interest is in looking at derivative 0 and derivative 2, c(1,0,0) for looking at original function (0th derivative) only, etc.
`width.range`	Bandwidth for local polynomial regression curve smoothing for each dependent variable/response; it can be a list that specifies a distinct bandwidth two-element vector for each response, or a single two-element vector that is used for each of the responses — the program is currently set up to allow linearly increasing or decreasing bandwidths, specified by this two-element vector with the increase (or decrease) occurring from the first argument in width.place to its second argument; the lpepa function within the lpridge package is called, which uses Epanecknikov kernel weighting, and the specifications of bandwidth in width.range

	will be used there; the default bandwidth is the range of indepVar (usually time) divided by 4, i.e., a constant global bandwidth.
`width.place`	Endpoints for width change assuming a non-constant bandwidth requested — the program is currently set up to allow linearly increasing or decreasing bandwidths, specified in width.range, with the increase (or decrease) occurring from the first argument in width.place to its second argument; it can be a list that specifies different endpoints for each response, or a single vector of endpoints that is used for each of the responses; default is no endpoints specified which means constant global bandwidth throughout the range of indepVar.
`min.obs`	Minimum oberservation (follow-up period) required. If specified, individuals whose follow-up period shorter than min.obs will be removed from calculation. Default = NA (use whole dataset provided).
`points.length`	Number of indep (time) points for dynamic correlation calculation for each response of each individual. This is the number of points between time 0 and minimum of maximum of individual follow-up periods (max_common_obs). Note that each individual's full follow-up time span is used in the local polynomial regression curve smoothing step, but only the first points.length number of time points (from time 0 to max_common_obs) is used in the following dynamic correlation calculation. Default points.length value is set to 100; points.length takes precedence unless points.by is specified.
`points.by`	Interval between indep (time) points for local polynomial regression curve smoothing for each response of each individual. Both integer and non-integer value could be specified, and time grid will be computed accordingly. Note that points.length takes precedence (default = 100) unless points.by is specified.
`boundary.trunc`	Indicate the boundary of indepVar that should be truncated after smoothing; this may be done in case of concerns about estimating dynamical correlation at the boundaries of indepVar; this is a two element vector, where the first argument is how much to truncate from the right of the minimum value of indep.var and the second argument is how much to truncate from the left of the maximum value of indep var, within an individual and specific response; default is no truncation, i.e., c(0,0).
`byOrder`	A vector that specifies the order of the variables/responses and derivatives (if any) to be the leading variable in the calculations; this will have an effect on how lag.input is to be interpreted; default is to use the order as specified in the depVar argument.
`max.dynCorrLag`	A specified lag value at which you would like to obtain a percentile BS dynamical correlation interval estimate; only single lag values are allowed (due to computational considerations), and the lag value considered might be that which is output from a lag analysis using the dynamicCorrelation function (see Example 2 on the dynamicCorrelation help page).
`B`	The number of samples used for the bootstrap.
`percentile`	The percentile used to construct confidence intervals for dynamic correlations.
`by.deriv.only`	If TRUE, the inter-dynamical correlations between different derivatives are not computed, which can save computation time (e.g., when function.choice=c(1,0,1) is specified and by.deriv.only=T, then only dynamical correlations will be calculated within (and not across) the 0th and 2nd derivative estimates, respectively); default is TRUE.

`seed`	The seed used in generating the BS samples.

Details

This function will provide smooth estimates (curves/functions or their derivatives) of responses of interest, then generate dynamical correlation estimates for each pair of responses. Lags of interest can be specified, using the lag.input argument. For smoothing, the function uses local polynomial smoothing using Epanecknikov kernel weighting (by calling the function lpepa within the lpridge package). The default global bandwidth is generated by taking the range of indepVar (usually time) and dividing by 4. This, by default, will be a constant global bandwidth, but proper specification of the width.range and width.place arguments can allow for a more flexible bandwidth choice, including different specification for each response in depVar.

Whereas the dynamicCorrelation program, which produces only point estimates, is fast, the bootstrapCI program is slow in its current form, as quite a bit of processing is required for each bootstrap sample. Details of the two-stage bootstrap algorithm can be found in Dubin and Muller (2005). We will attempt to boost computing speed in future versions.

In addition, as pointed out in Dubin and Muller (2005), a downward shift of the two-stage bootstrap CI toward 0 is intentional, in order to account for error in the curve creation step. However, it should be noted that greater than expected downward shifts may occur for dynamical correlations that are high in magnitude. A future version of this function will attempt to make this bootstrap approach more robust in this situation.

Author(s)

Joel A. Dubin, Mike Li, Dandi Qiao, Hans-Georg Müller

See Also

lpepa

Examples

```
## Example 1: using default smoothing parameters, obtain bootstrap CI
##            estimates for all three pairs of responses, for original
##            function only. Note that B=200 or greater should be
##            considered for real data analysis.

examp1.bs <- bootstrapCI(dataFrame = dynCorrData,
                         depVar = c('resp1', 'resp2', 'resp3'),
                         indepVar = 'time',
                         subjectVar = 'subject',
                         points.by = 1,
                         function.choice = c(1,0,0),
                         B = 2, percentile = c(0.025, 0.975), seed = 5)
examp1.bs

## Example 2: using default smoothing parameters, obtain bootstrap CI
##            estimates for all three pairs of responses, for original
##            function only, at -10 days lag, at .01 and .99 percentiles.
##           Note that B=200 or greater should be considered for real
##           data analysis.

examp2.bs <- bootstrapCI(dataFrame = dynCorrData,
                         depVar = c('resp1', 'resp2', 'resp3'),
                         indepVar = 'time',
                         subjectVar = 'subject',
                         points.by = 1,
                         function.choice = c(1,0,0), max.dynCorrLag = -10,
                         B = 2, percentile = c(0.01, 0.99), seed = 7)
examp2.bs
```

---

dynamicCorrelation *Dynamic Correlation*

Description

Computes dynamical correlation estimates for pairs of longitudinal responses, including consideration of lags and derivatives, following a local polynomial regression smoothing step.

Usage

```
dynamicCorrelation(dataFrame, depVar, indepVar, subjectVar,
                   function.choice, width.range, width.place,
                   min.obs, points.length, points.by, boundary.trunc,
                   lag.input, byOrder, by.deriv.only, full.lag.output)
```

Arguments

`dataFrame`	The data frame that contains the dependent variables/responses, the independent variable (often time), and the subject/individual identification; there should be one row entry for each combination of subject/individual and indepVar (often time).
`depVar`	Dependent variables/responses; at least two are necessary for purposes of calculating at least one dynamical correlation estimate of interest; there should be a unique column within depVar for each response.
`indepVar`	Independent variable, typically the discrete recorded time points at which the dependent variables were collected; note that this is the independent variable for purposes of curve creation leading into estimating the dynamical correlations between pairs of dependent variables; must be contained in a single column.
`subjectVar`	Column name of the individuals; there should be one row entry for each combination of subject/individual and indepVar.
`function.choice`	A vector of length 3 to indicate which derivatives desired for local polynomial curve smoothing; 1st entry is for 0th derivative (i.e., original function), 2nd entry is for 1st derivative, 3rd is for 2nd derivative; 1=yes, 0=no. e.g., c(1,0,1) would

	be specified if interest is in looking at derivative 0 and derivative 2, c(1,0,0) for looking at original function (0th derivative) only, etc.
`width.range`	Bandwidth for local polynomial regression curve smoothing for each dependent variable/response; it can be a list that specifies a distinct bandwidth two-element vector for each response, or a single two-element vector that is used for each of the responses — the program is currently set up to allow linearly increasing or decreasing bandwidths, specified by this two-element vector with the increase (or decrease) occurring from the first argument in width.place to its second argument; the lpepa function within the lpridge package is called, which uses Epanecknikov kernel weighting, and the specifications of bandwidth in width.range will be used there; the default bandwidth is the range of indepVar (usually time) divided by 4, i.e., a constant global bandwidth.
`width.place`	Endpoints for width change assuming a non-constant bandwidth requested — the program is currently set up to allow linearly increasing or decreasing bandwidths, specified in width.range, with the increase (or decrease) occurring from the first argument in width.place to its second argument; it can be a list that specifies different endpoints for each response, or a single vector of endpoints that is used for each of the responses; default is no endpoints specified which means constant global bandwidth throughout the range of indepVar.
`min.obs`	Minimum oberservation (follow-up period) required. If specified, individuals whose follow-up period shorter than min.obs will be removed from calculation. Default = NA (use whole dataset provided).
`points.length`	Number of indep (time) points for dynamic correlation calculation for each response of each individual. This is the number of points between time 0 and minimum of maximum of individual follow-up periods (max_common_obs). Note that each individual's full follow-up time span is used in the local polynomial regression curve smoothing step, but only the first points.length number of time points (from time 0 to max_common_obs) is used in the following dynamic correlation calculation. Default points.length value is set to 100; points.length takes precedence unless points.by is specified.
`points.by`	Interval between indep (time) points for local polynomial regression curve smoothing for each response of each individual. Both integer and non-integer value could be specified, and time grid will be computed accordingly. Note that points.length takes precedence (default = 100) unless points.by is specified.
`boundary.trunc`	Indicate the boundary of indepVar that should be truncated after smoothing; this may be done in case of concerns about estimating dynamical correlation at the boundaries of indepVar; this is a two element vector, where the first argument is how much to truncate from the right of the minimum value of indep.var and the second argument is how much to truncate from the left of the maximum value of indep var, within an individual and specific response; default is no truncation, i.e., c(0,0).
`lag.input`	Values of lag to be considered; can be a vector of requested lags, for which a dynamical correlation estimate will be produced for each pair of responses at each lag; a positive value of lag.input means that the first entry for the dynamical correlation leads (occurs before) the second entry — conversely, a negative value means that the second entry for the correlation leads the first entry; default is no lag at all considered.

`byOrder`	A vector that specifies the order of the variables/responses and derivatives (if any) to be the leading variable in the calculations; this will have an effect on how lag.input is to be interpreted; default is to use the order as specified in the depVar argument.
`by.deriv.only`	If TRUE, the inter-dynamical correlations between different derivatives are not computed, which can save computation time (e.g., when function.choice=c(1,0,1) is specified and by.deriv.only=T, then only dynamical correlations will be calculated within (and not across) the 0th and 2nd derivative estimates, respectively); default is TRUE.
`full.lag.output`	If TRUE, the dynamical correlation values for each pair of responses and requested derivative will be stored in vectors corresponding to different lag values, which enables plotting the correlations as a function of lag values; all the vectors will be stored in the returned attribute resultMatrix; default is FALSE.

Details

This function will provide smooth estimates (curves/functions or their derivatives) of longitudinal responses of interest per individual, then generate dynamical correlation estimates for each pair of responses. Lags of interest can be specified, using the lag.input argument. For smoothing, the function uses local polynomial smoothing using Epanecknikov kernel weighting (by calling the function lpepa within the lpridge package). The default global bandwidth is generated by taking the range of indepVar (usually time) and dividing by 4. This, by default, will be a constant global bandwidth, but proper specification of the width.range and width.place arguments can allow for a more flexible bandwidth choice, including different specification for each response in depVar. Note that the correlation will only be calculated as far as min of max observation time (max_common_obs) across all individuals, by default, hence caution should be taken if there is large heterogeneity among max observation times across individuals.

Details of the methodology for dynamical correlation can be found in Dubin and Muller (2005).

Author(s)

Joel A. Dubin, Mike Li, Dandi Qiao, Hans-Georg Müller

See Also

lpepa

Examples

```
## Example 1: using default smoothing parameters, obtain dynamical
##            correlation estimates for all three pairs of responses,
##            for both original function and the first derivative.
##            Note that in dynCorrData, mininum of maximum obs. time
##            = 120, results should be the same if either points.by
##            = 1 or points.length = 120 is specified.

examp1_by <- dynamicCorrelation(dataFrame = dynCorrData,
                                depVar = c('resp1', 'resp2', 'resp3'),
                                indepVar = 'time',
                                points.by = 1,
                                subjectVar = 'subject',
                                function.choice = c(1,1,0))

examp1_len <- dynamicCorrelation(dataFrame = dynCorrData,
                                 depVar = c('resp1', 'resp2', 'resp3'),
                                 indepVar = 'time',
                                 points.length = 120,
                                 subjectVar = 'subject',
                                 function.choice = c(1,1,0))
examp1_by
examp1_len

## Example 1a: re-run Example 1 using original dataset, but with min.obs
##             set to 200. Range in lengths of follow-up periods between
##             individuals is reduced.

examp1a <- dynamicCorrelation(dataFrame = dynCorrData,
                              depVar = c('resp1', 'resp2', 'resp3'),
                              indepVar = 'time',
                              min.obs = 150,
                              points.by = 1,
                              subjectVar = 'subject',
                              function.choice = c(1,0,0))
examp1a

## Example 2: using default smoothing parameters, obtain dynamical
##            correlation estimates for all three pairs of responses,
##            looking at range of lags between -10 and +10, for original
##            functions only

examp2 <- dynamicCorrelation(dataFrame = dynCorrData,
                             depVar = c('resp1', 'resp2', 'resp3'),
                             indepVar = 'time',
                             points.by = 1,
                             subjectVar = 'subject',
                             function.choice = c(1,0,0),
                             lag.input = seq(-20,20, by=1))
examp2

## note: output includes zero lag correlations, as well as maximum
##       correlation (in absolute value) in max.dynCorr and and its
##       corresponding lag value in max.dynCorrLag

## Example 3: re-rerun example 2, but set up for plotting of specified
##            lagged correlations

examp3 <- dynamicCorrelation(dataFrame = dynCorrData,
                             depVar = c('resp1', 'resp2', 'resp3'),
                             indepVar = 'time',
                             subjectVar = 'subject',
                             points.by = 1,
                             function.choice = c(1,0,0),
                             lag.input = seq(-20,10, by=1),
                             full.lag.output = TRUE)

# conduct plotting, with one panel for each pair of responses considered;
# the ylim adjustment is made here for the different magnitude of the
# correlations between the two pairs

par(mfrow=c(1,2))
plot(seq(-20,10, by=1),
     examp3$lagResultMatrix[[1]][1,],
     type='b',
     xlab = 'lag order (in days)',
     ylab = 'lagged correlations',
     ylim = c(-0.4, -0.2),
     main = 'dyncorr b/t resp1 and resp2 as a function of lag')

abline(v = examp3$max.dynCorrLag[[1]][1,2], lty = 2)

plot(seq(-20,10, by=1),
     examp3$lagResultMatrix[[1]][2,],
     type='b',
     xlab = 'lag order (in days)',
     ylab = 'lagged correlations',
     ylim = c(0.3, 0.5),
     main = 'dyncorr b/t resp1 and resp3 as a function of lag')

abline(v = examp3$max.dynCorrLag[[1]][1,3], lty = 2)

## Example 4: same as the original function piece of Example 1,
##            except now adjust the constant global bandwidth
##            from the default to 40

examp4 <- dynamicCorrelation(dataFrame = dynCorrData,
                             depVar = c('resp1', 'resp2', 'resp3'),
                             indepVar = 'time',
                             points.by = 1,
                             subjectVar = 'subject',
                             function.choice = c(1,0,0),
                             width.range = c(40, 40))
examp4
```

dynCorrData *dynCorrData*

Description

An example dataset for use in the example calls in the help files for the dynamicCorrelation and bootstrapCI functions.

This dataset contains three longitudinal responses for each subject, each measured every ten days over a varying length of follow-up depending upon subject (for each of 34 subjects). Each row contains all three responses for a given subject and a given follow-up time. The features are that the first and third responses have a moderate-sized positive dynamical correlation, while the first and second, and second and third, respectively, have lesser negative dynamical correlations. The dataset is called in the examples on the dynamicCorrelation and bootstrapCI help pages.

Usage

```
data(dynCorrData)
```

Format

A 648 by 5 data frame.

Index