

# Package ‘cointReg’

August 29, 2016

**Type** Package

**Title** Parameter Estimation and Inference in a Cointegrating Regression

**Date** 2016-06-14

**Version** 0.2.0

**Description** Cointegration methods are widely used in empirical macroeconomics and empirical finance. It is well known that in a cointegrating regression the ordinary least squares (OLS) estimator of the parameters is super-consistent, i.e. converges at rate equal to the sample size  $T$ . When the regressors are endogenous, the limiting distribution of the OLS estimator is contaminated by so-called second order bias terms, see e.g. Phillips and Hansen (1990) <DOI:10.2307/2297545>. The presence of these bias terms renders inference difficult. Consequently, several modifications to OLS that lead to zero mean Gaussian mixture limiting distributions have been proposed, which in turn make standard asymptotic inference feasible. These methods include the fully modified OLS (FM-OLS) approach of Phillips and Hansen (1990) <DOI:10.2307/2297545>, the dynamic OLS (D-OLS) approach of Phillips and Loretan (1991) <DOI:10.2307/2298004>, Saikkonen (1991) <DOI:10.1017/S0266466600004217> and Stock and Watson (1993) <DOI:10.2307/2951763> and the new estimation approach called integrated modified OLS (IM-OLS) of Vogelsang and Wagner (2014) <DOI:10.1016/j.jeconom.2013.10.015>. The latter is based on an augmented partial sum (integration) transformation of the regression model. IM-OLS is similar in spirit to the FM- and D-OLS approaches, with the key difference that it does not require estimation of long run variance matrices and avoids the need to choose tuning parameters (kernels, bandwidths, lags). However, inference does require that a long run variance be scaled out. This package provides functions for the parameter estimation and inference with all three modified OLS approaches. That includes the automatic bandwidth selection approaches of Andrews (1991) <DOI:10.2307/2938229> and of Newey and West (1994) <DOI:10.2307/2297912> as well as the calculation of the long run variance.

**URL** <https://github.com/aschersleben/cointReg>

**BugReports** <https://github.com/aschersleben/cointReg/issues>

**License** GPL-3

**Imports** checkmate ( $\geq 1.6.0$ ), MASS, matrixStats ( $\geq 0.14.1$ )

**RoxygenNote** 5.0.1

**Suggests** testthat, knitr, rmarkdown

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Philipp Aschersleben [aut, cre],  
Martin Wagner [aut] (Author of underlying MATLAB code.)

**Maintainer** Philipp Aschersleben <aschersleben@statistik.tu-dortmund.de>

**Repository** CRAN

**Date/Publication** 2016-06-14 11:58:42

## R topics documented:

cointReg-package . . . . .	2
checkDoptions . . . . .	3
checkObject . . . . .	4
checkVars . . . . .	6
cointReg . . . . .	7
cointRegD . . . . .	9
cointRegFM . . . . .	11
cointRegIM . . . . .	13
getBandwidth . . . . .	15
getLeadLag . . . . .	17
getLongRunVar . . . . .	18
getLongRunWeights . . . . .	19
getModD . . . . .	20
makeLeadLagMatrix . . . . .	21
plot.cointReg . . . . .	22
print.cointReg . . . . .	23
<b>Index</b>	<b>25</b>

---

cointReg-package	<i>The cointReg package</i>
------------------	-----------------------------

---

## Description

Parameter Estimation and Inference in a Cointegrating Regression

**Details**

See the vignette:

```
vignette("cointReg")
```

See the DESCRIPTION:

```
help(package = cointReg)
```

See the README:

<https://github.com/aschersleben/cointReg/blob/master/README.md>

Open the package documentation page:

```
package?cointReg
```

Further information and bug reporting:

<https://github.com/aschersleben/cointReg>

**Functions**

- `cointReg`(method = c("FM", "D", "IM"), ...)
 

General function to estimate parameters of the given model. Three methods are possible; they can be chosen directly by using one of the following functions:

  - `cointRegFM`: Fully Modified OLS
  - `cointRegD`: Dynamic OLS
  - `cointRegIM`: Integrated Modified OLS
- `print`

Print clear results.
- `plot`

Plot the residuals of a cointReg model.
- Helper functions:
  - Checking inputs and arguments:  
`checkObject`, `checkVars`
  - Calculation of bandwidth and long run variance:  
`getBandwidth`, `getBandwidthAnd`, `getBandwidthNW`  
`getLongRunVar`, `getLongRunWeights`
  - Additional D-OLS functions:  
`getLeadLag`, `makeLeadLagMatrix`, `getModD`, `checkDoptions`

---

checkDoptions

*Check list D.options.*

---

**Description**

Checking the list D.options, that is an argument of `cointRegD`.

**Usage**

```
checkDoptions(n.lag = NULL, n.lead = NULL, kmax = c("k4", "k12"),
  info.crit = c("AIC", "BIC"))
```

**Arguments**

n.lag, n.lead	[NULL   numeric(1)] Have to be "integerish" and > 0.
kmax	[NULL   character(1)] One of "k4" or "k12".
info.crit	[NULL   character(1)] One of "AIC" or "BIC".

**Value**

list . List with the checked and (if necessary) converted arguments.

If one of n.lag and n.lead is NULL, only kmax and info.crit will be not NULL.

**See Also**

Other check: [checkObject](#), [checkVars](#)

**Examples**

```
checkDoptions(n.lag = 3, n.lead = 4)
checkDoptions(info.crit = "BIC")
checkDoptions()

# It's not sufficient to include only one of "n.lag" and "n.lead":
checkDoptions(n.lag = 2)
```

---

checkObject	<i>Variable check for single objects.</i>
-------------	---

---

**Description**

Checking the variable and convert it for internal use, if necessary. (Also used by the `cointmonitorR` package.)

**Usage**

```
checkObject(obj, obj.name, ..., out = "return", .env)
```

**Arguments**

obj	[any] Variable or value to check and convert.
obj.name	[character(1)] Name of the object to check. If missing, the name of obj has to be one of the possible names (see details).

...	[any] An alternative to the use of the <code>obj</code> and <code>obj.name</code> arguments is to directly give the name and the variable to be checked via <code>name = variable</code> arguments (see examples). In the case of more than one ... argument, <a href="#">checkVars</a> will be called internally.
out	[character] Whether to "return" or to "assign" the checked (and converted) object. Also possible: <code>c("return", "assign")</code> .
.env	[environment] Environment to which to assign the converted <code>obj</code> (usually the same on that contains <code>obj</code> , if it's a variable). Required, if argument <code>out</code> contains "assign".

### Details

Possible values of `obj.name` to check:

"y", "x.stat": Of type `numeric`, `matrix` or `data.frame`. Only the first row/column will be used.  
Converted to object: column matrix

"y.fm", "x.coimt", "deter": Of type `numeric`, `matrix` or `data.frame`.  
Converted to object: column matrix

"m": Of type `numeric(1)`, has to be greater than 0.

"model": One of `c("FM", "D", "IM")`.

"signif.level": Of type `numeric(1)`, has to be in the interval `[0.01, 0.1]`.

"trend", "return.stats", "return.input", "demeaning", "t.test": Converted to object: `logical(1)`.

"kernel": One of `c("ba", "bo", "da", "pa", "qs", "th", "tr")`.

"bandwidth": One of `c("and", "nw")`.

"selector": One or both `c(1, 2)`.

### Value

The checked and converted argument is assigned to the given environment (`.env`) and/or returned (depending on the argument `out`).

### See Also

Other check: [checkOptions](#), [checkVars](#)

### Examples

```
x = matrix(1:20, nrow = 2)
x2 = checkObject(x, "x.coimt")
x2

env = environment()
y = 1:10
checkObject(y, out = "assign", .env = env)
```

```

y

# example for the use of the ... argument:
det = rbind(1, 1:10)
x3 = sin(10:20)
det2 = checkObject(deter = det)
det2
(checkObject(deter = det, x.stat = x3))

```

---

checkVars

*Multiple variable checks for certain values.*

---

### Description

Checking the arguments and convert them for internal use, if necessary.

### Usage

```
checkVars(..., out = "assign", .env)
```

### Arguments

...	[any] Variables to check, see details.
out	[character] Whether to "return" or to "assign" the checked (and converted) object. Also possible: c("return", "assign").
.env	[environment] Environment to which to assign the converted obj (usually the same on that contains obj, if it's a variable). Required, if argument out contains "assign".

### Details

See [checkObject](#) for details.

### Value

The checked and converted arguments are assigned to the given environment (.env) or invisibly returned as a list.

### See Also

Other check: [checkOptions](#), [checkObject](#)

**Examples**

```

env = environment()
x.data = data.frame(a = 1:10, b = 20:11)
y.data = 1:10
checkVars(x.coint = x.data, y = y.data, .env = env)
x.coint
y

test = checkVars(x.coint = x.data, y = y.data, out = "return")
str(test)

# If the variables already have the "right" name,
# there's no need to do something like
# checkVars(kernel = kernel, bandwidth = bandwidth) -
# just call checkVars without specifying the arguments:
kernel = "q"
bandwidth = "a"
(checkVars(kernel, bandwidth, out = "return"))

```

cointReg

*Estimation and Inference for cointegrating regressions***Description**

Computes either the Phillips and Hansen (1990) Fully Modified OLS estimator, or the Saikkonen (1990) Dynamic OLS estimator, or the Vogelsang and Wagner (2014) Integrated Modified OLS estimator.

**Usage**

```
cointReg(method = c("FM", "D", "IM"), x, y, ...)
```

**Arguments**

method	[character(1)] Select the method for the estimation of your cointegration model: <ul style="list-style-type: none"> <li>• "FM": FM-OLS (default), see details at <a href="#">cointRegFM</a></li> <li>• "D": D-OLS, see details at <a href="#">cointRegD</a></li> <li>• "IM": IM-OLS, see details at <a href="#">cointRegIM</a></li> </ul>
x	[numeric   matrix   data.frame] RHS variables on which to apply the model estimation.
y	[numeric   matrix   data.frame] LHS variable(s) on which to apply the model estimation. Usually one-dimensional, but a matrix or data.frame with more than one column is also possible (only for FM-OLS).

...

[any]

Arguments passed to the corresponding cointReg function, like:

- x, y, deter: data to include in the model
- kernel, bandwidth: parameters for calculating the long-run variance
- n.lead, n.lag, kmax, info.crit: D-OLS specific arguments.
- selector, t.test: IM-OLS specific arguments.
- check: Whether to check (and if necessary convert) the arguments. See [checkVars](#) for further information.

### Value

cointReg object.

### References

- Phillips, P.C.B. and B. Hansen (1990): "Statistical Inference in Instrumental Variables Regression with I(1) Processes," *Review of Economic Studies*, 57, 99–125, [DOI:10.2307/2297545](#).
- Phillips, P.C.B. and M. Loretan (1991): "Estimating Long Run Economic Equilibria," *Review of Economic Studies*, 58, 407–436, [DOI:10.2307/2298004](#).
- Saikkonen, P. (1991): "Asymptotically Efficient Estimation of Cointegrating Regressions," *Econometric Theory*, 7, 1–21, [DOI:10.1017/S0266466600004217](#).
- Stock, J.H. and M.W. Watson (1993): "A Simple Estimator of Cointegrating Vectors in Higher Order Integrated Systems," *Econometrica*, 61, 783–820, [DOI:10.2307/2951763](#).
- Vogelsang, T.J. and M. Wagner (2014): "Integrated Modified OLS Estimation and Fixed-b Inference for Cointegrating Regressions," *Journal of Econometrics*, 148, 741–760, [DOI:10.1016/j.jeconom.2013.10.015](#).

### See Also

Other cointReg: [cointRegD](#), [cointRegFM](#), [cointRegIM](#), [plot.cointReg](#), [print.cointReg](#)

### Examples

```
set.seed(1909)
x1 = cumsum(rnorm(100, mean = 0.05, sd = 0.1))
x2 = cumsum(rnorm(100, sd = 0.1)) + 1
x3 = cumsum(rnorm(100, sd = 0.2)) + 2
x = cbind(x1, x2, x3)
y = x1 + x2 + x3 + rnorm(100, sd = 0.2) + 1
deter = cbind(level = 1, trend = 1:100)
cointReg("FM", x = x, y = y, deter = deter, kernel = "ba",
        bandwidth = "and")

# Compare the results of all three models:
res = sapply(c("FM", "D", "IM"), cointReg, x = x, y = y, deter = deter)
do.call(cbind, lapply(res, "[[", "theta"))
```



cointRegD

*Dynamic OLS***Description**

Computes the Saikkonen (1990) Dynamic OLS estimator.

**Usage**

```
cointRegD(x, y, deter, kernel = c("ba", "pa", "qs", "tr"),
  bandwidth = c("and", "nw"), n.lead = NULL, n.lag = NULL,
  kmax = c("k4", "k12"), info.crit = c("AIC", "BIC"), demeaning = FALSE,
  check = TRUE, ...)
```

**Arguments**

x	[numeric   matrix   data.frame] RHS variables on which to apply the D-OLS estimation (see Details).
y	[numeric   matrix   data.frame] LHS variable(s) on which to apply the D-OLS estimation (see Details). Has to be one-dimensional. If matrix, it may have only one row or column, if data.frame just one column.
deter	[numeric   matrix   data.frame   NULL] Deterministic variable to include in the equation (see Details). If it's NULL or missing, no deterministic variable is included in the model.
kernel	[character(1)] The kernel function to use for calculating the long-run variance. Default is Bartlett kernel ("ba"), see Details for alternatives.
bandwidth	[character(1)   integer(1)] The bandwidth to use for calculating the long-run variance. Default is Andrews (1991) ("and"), an alternative is Newey West (1994) ("nw").
n.lead, n.lag	[numeric(1)   NULL] Numbers of Leads and Lags (see Details). Default is NULL.
kmax	[character(1)] Maximal value for lags and leads if generated automatically (see Details). Default is "k4".
info.crit	[character(1)] Information criterion to use for the automatical calculation of lags and leads. Default is "AIC".
demeaning	[logical] Demeaning of residuals in <a href="#">getLongRunVar</a> . Default is FALSE.
check	[logical] Whether to check (and if necessary convert) the arguments. See <a href="#">checkVars</a> for further information.
...	Arguments passed to <a href="#">getBandwidthNW</a> .

## Details

The equation for which the FM-OLS estimator is calculated:

$$y = \delta \cdot D + \beta \cdot x + u$$

with  $D$  as the deterministic matrix. Then  $\theta = (\delta', \beta')$  is the full parameter vector.

Information about the D-OLS specific arguments:

`n.lag`, `n.lead` A positive number to set the number of lags and leads. If at least one of them is equal to NULL (default), the function `getLeadLag` will be used to calculate them automatically (see Choi and Kurozumi (2012) for details). In that case, the following two arguments are needed.

`kmax` Maximal value for lags and leads, when they are calculated automatically. If "k4", then the maximum is equal to `floor(4 * (x.T/100)^(1/4))`, else it's `floor(12 * (x.T/100)^(1/4))` with `x.T` is equal to the data's length. One of "k4" or "k12". Default is "k4".

`info.crit` Information criterion to use for the automatical calculation of lags and leads. One of "AIC" or "BIC". Default is "AIC".

## Value

`cointReg` . List with components:

`beta` [numeric ] coefficients of the regressors  
`delta` [numeric ] coefficients of the deterministic  
`theta` [numeric ] combined coefficients of beta and delta  
`sd.theta` [numeric ] standard errors for theta  
`t.theta` [numeric ] t-values for theta  
`p.theta` [numeric ] p-values for theta  
`theta.all` [numeric ] combined coefficients of beta, delta and the auxiliary leads-and-lags regressors  
`residuals` [numeric ] D-OLS residuals (length depends on leads and lags)  
`omega.u.v` [numeric ] conditional long-run variance based on OLS residuals  
`varmat` [matrix ] variance-covariance matrix  
`Omega` [list ] the whole long-run variance matrix and parts of it  
`bandwidth` [list ] number and name of the calculated bandwidth  
`kernel` [character ] abbr. name of kernel type  
`lead.lag` [list ] leads-and-lags parameters

## References

- Phillips, P.C.B. and M. Loretan (1991): "Estimating Long Run Economic Equilibria," *Review of Economic Studies*, 58, 407–436, DOI:10.2307/2298004.
- Saikkonen, P. (1991): "Asymptotically Efficient Estimation of Cointegrating Regressions," *Econometric Theory*, 7, 1–21, DOI:10.1017/S0266466600004217.
- Stock, J.H. and M.W. Watson (1993): "A Simple Estimator of Cointegrating Vectors in Higher Order Integrated Systems," *Econometrica*, 61, 783–820, DOI:10.2307/2951763.

**See Also**

Other cointReg: [cointRegFM](#), [cointRegIM](#), [cointReg](#), [plot.cointReg](#), [print.cointReg](#)

Other D-OLS: [getLeadLag](#), [getModD](#), [makeLeadLagMatrix](#)

**Examples**

```
set.seed(1909)
x1 <- cumsum(rnorm(100, mean = 0.05, sd = 0.1))
x2 <- cumsum(rnorm(100, sd = 0.1)) + 1
x3 <- cumsum(rnorm(100, sd = 0.2)) + 2
x <- cbind(x1, x2, x3)
y <- x1 + x2 + x3 + rnorm(100, sd = 0.2) + 1
deter <- cbind(level = 1, trend = 1:100)
test <- cointRegD(x, y, deter, n.lead = 2, n.lag = 2,
                 kernel = "ba", bandwidth = "and")

print(test)
test2 <- cointRegD(x, y, deter, kmax = "k4", info.crit = "BIC",
                  kernel = "ba", bandwidth = "and")

print(test2)
```

---

cointRegFM

*Fully Modified OLS*


---

**Description**

Computes the Phillips and Hansen (1990) Fully Modified OLS estimator.

**Usage**

```
cointRegFM(x, y, deter, kernel = c("ba", "pa", "qs", "tr"),
           bandwidth = c("and", "nw"), demeaning = FALSE, check = TRUE, ...)
```

**Arguments**

x	[numeric   matrix   data.frame] RHS variables on which to apply the FM-OLS estimation (see Details).
y	[numeric   matrix   data.frame] LHS variable(s) on which to apply the FM-OLS estimation (see Details). Usually one-dimensional, but a matrix or data.frame with more than one column is also possible.
deter	[numeric   matrix   data.frame   NULL] Deterministic variable to include in the equation (see Details). If it's NULL or missing, no deterministic variable is included in the model.
kernel	[character(1)] The kernel function to use for calculating the long-run variance. Default is Bartlett kernel ("ba"), see Details for alternatives.

bandwidth	[character(1)   integer(1)] The bandwidth to use for calculating the long-run variance. Default is Andrews (1991) ("and"), an alternative is Newey West (1994) ("nw").
demeaning	[logical] Demeaning of residuals in <code>getLongRunVar</code> . Default is FALSE.
check	[logical] Whether to check (and if necessary convert) the arguments. See <code>checkVars</code> for further information.
...	Arguments passed to <code>getBandwidthNW</code> .

### Details

The equation for which the FM-OLS estimator is calculated:

$$y = \delta \cdot D + \beta \cdot x + u$$

with  $D$  as the deterministic matrix. Then  $\theta = (\delta', \beta')$  is the full parameter vector.

The calculation of t-values and the variance-covariance matrix is only possible, if  $y$  is one-dimensional.

### Value

`cointReg` . List with components:

delta [numeric | matrix ] coefficients as vector / matrix  
 beta [numeric | matrix ] coefficients as vector / matrix  
 theta [numeric | matrix ] combined coefficients of beta and delta as vector / matrix  
 sd.theta [numeric ] standard errors for theta  
 t.theta [numeric ] t-values for theta  
 p.theta [numeric ] p-values for theta  
 residuals [numeric ] FM-OLS residuals (first value is always missing)  
 omega.u.v [numeric ] conditional long-run variance based on OLS residuals.  
 varmat [matrix ] variance-covariance matrix  
 Omega [list ] the whole long-run variance matrix and parts of it  
 beta.OLS [numeric | matrix ] OLS coefficients as vector / matrix  
 delta.OLS [numeric | matrix ] OLS coefficients as vector / matrix  
 u.OLS [numeric ] OLS residuals  
 bandwidth [list ] number and name of bandwidth  
 kernel [character ] abbr. name of kernel type

### References

- Phillips, P.C.B. and B. Hansen (1990): "Statistical Inference in Instrumental Variables Regression with I(1) Processes," *Review of Economic Studies*, 57, 99–125, DOI:10.2307/2297545.

**See Also**

Other cointReg: [cointRegD](#), [cointRegIM](#), [cointReg](#), [plot.cointReg](#), [print.cointReg](#)

**Examples**

```
set.seed(1909)
x1 = cumsum(rnorm(100, mean = 0.05, sd = 0.1))
x2 = cumsum(rnorm(100, sd = 0.1)) + 1
x3 = cumsum(rnorm(100, sd = 0.2)) + 2
x = cbind(x1, x2, x3)
y = x1 + x2 + x3 + rnorm(100, sd = 0.2) + 1
deter = cbind(level = 1, trend = 1:100)
test = cointRegFM(x, y, deter, kernel = "ba", bandwidth = "and")
print(test)
```

---

cointRegIM

*Integrated Modified OLS*


---

**Description**

Computes the Vogelsang and Wagner (2014) Integrated Modified OLS estimator.

**Usage**

```
cointRegIM(x, y, deter, selector = 1, t.test = TRUE, kernel = c("ba",
  "pa", "qs", "tr"), bandwidth = c("and", "nw"), check = TRUE, ...)
```

**Arguments**

<code>x</code>	[numeric   matrix   data.frame] RHS variables on which to apply the IM-OLS estimation (see Details).
<code>y</code>	[numeric   matrix   data.frame] LHS variable(s) on which to apply the IM-OLS estimation (see Details). Has to be one-dimensional. If matrix, it may have only one row or column, if data.frame just one column.
<code>deter</code>	[numeric   matrix   data.frame   NULL] Deterministic variable to include in the equation (see Details). If it's NULL or missing, no deterministic variable is included in the model.
<code>selector</code>	[numeric] Choose the regression type: 1, 2, or c(1, 2) (see Details). Default is 1.
<code>t.test</code>	[logical] Whether to calculate t-values for the coefficients of the first regression. Default is TRUE. Attention: Needs more calculation time, because an additional FM-OLS model has to be fitted to get the long-run variance.

kernel	[character(1)] The kernel function to use for calculating the long-run variance. Default is Bartlett kernel ("ba"), see Details for alternatives.
bandwidth	[character(1)   integer(1)] The bandwidth to use for calculating the long-run variance. Default is Andrews (1991) ("and"), an alternative is Newey West (1994) ("nw").
check	[logical] Whether to check (and if necessary convert) the arguments. See <a href="#">checkVars</a> for further information.
...	Arguments passed to <a href="#">getBandwidthNW</a> .

### Details

The equation for which the IM-OLS estimator is calculated (type 1):

$$S_y = \delta \cdot S_D + \beta \cdot S_x + \gamma \cdot x + u$$

where  $S_y$ ,  $S_x$  and  $S_D$  are the cumulated sums of  $y$ ,  $x$  and  $D$  (with  $D$  as the deterministic matrix). Then  $\theta = (\delta', \beta', \gamma')'$  is the full parameter vector.

The equation for which the IM-OLS estimator is calculated (type 2):

$$S_y = \delta \cdot S_D + \beta \cdot S_x + \gamma \cdot x + \lambda \cdot Z + u$$

where  $S_y$ ,  $S_x$  and  $S_D$  are the cumulated sums of  $y$ ,  $x$  and  $D$  (with  $D$  as the deterministic matrix) and  $Z$  as defined in equation (19) in Vogelsang and Wagner (2015). Then  $\theta = (\delta', \beta', \gamma', \lambda)'$  is the full parameter vector.

### Value

cointReg . List with components:

delta	[numeric ] coefficients of the deterministic (cumulative sum $S_{deter}$ )
beta	[numeric ] coefficients of the regressors (cumulative sum $S_x$ )
gamma	[numeric ] coefficients of the regressors (original regressors $x$ )
theta	[numeric ] combined coefficients of beta, delta
sd.theta	[numeric ] standard errors for the theta coefficients
t.theta	[numeric ] t-values for the theta coefficients
p.theta	[numeric ] p-values for the theta coefficients
theta.all	[numeric ] combined coefficients of beta, delta, gamma
residuals	[numeric ] IM-OLS residuals. Attention: These are the first differences of $S_u$ – the original residuals are stored in u.plus.
u.plus	[numeric ] IM-OLS residuals, not differenced. See residuals above.
omega.u.v	[numeric ] conditional long-run variance based on OLS residuals, via cointRegFM (in case of argument t.test is TRUE) or NULL
varmat	[matrix ] variance-covariance matrix

Omega [matrix ] NULL (no long-run variance matrix for this regression type)  
 bandwidth [list ] number and name of bandwidth if `t.test = TRUE`  
 kernel [character ] abbr. name of kernel type if `t.test = TRUE`  
 delta2 [numeric ] coefficients of the deterministic (cumulative sum  $S_{deter}$ ) for regression type 2  
 beta2 [numeric ] coefficients of the regressors (cumulative sum  $S_x$ ) for regression type 2  
 gamma2 [numeric ] coefficients of the regressors (original regressors  $x$ ) for regression type 2  
 lambda2 [numeric ] coefficients of the Z regressors for regression type 2  
 theta2 [numeric ] combined coefficients of beta2, delta2, gamma2 and lambda2 for regression type 2  
 u.plus2 [numeric ] IM-OLS residuals for regression type 2

## References

- Vogelsang, T.J. and M. Wagner (2014): "Integrated Modified OLS Estimation and Fixed-b Inference for Cointegrating Regressions," *Journal of Econometrics*, 148, 741–760, DOI:10.1016/j.jeconom.2013.10.015.

## See Also

Other `cointReg`: [cointRegD](#), [cointRegFM](#), [cointReg](#), [plot.cointReg](#), [print.cointReg](#)

## Examples

```

set.seed(1909)
x1 = cumsum(rnorm(100, mean = 0.05, sd = 0.1))
x2 = cumsum(rnorm(100, sd = 0.1)) + 1
x3 = cumsum(rnorm(100, sd = 0.2)) + 2
x = cbind(x1, x2, x3)
y = x1 + x2 + x3 + rnorm(100, sd = 0.2) + 1
deter = cbind(level = 1, trend = 1:100)
test = cointRegIM(x, y, deter, selector = c(1, 2), t.test = TRUE,
                 kernel = "ba", bandwidth = "and")
print(test)

```

---

getBandwidth

*Automatic Bandwidth Selection*

---

## Description

Automatic bandwidth selection of Andrews (1991) and of Newey and West (1994).

## Usage

```

getBandwidth(u, bandwidth = c("and", "nw"), kernel, ..., check = TRUE)

getBandwidthAnd(u, kernel = c("ba", "pa", "qs", "th", "tr"), check = TRUE)

getBandwidthNW(u, kernel = c("ba", "pa", "qs"), inter = FALSE,
               u.weights = NULL, check = TRUE)

```

**Arguments**

u	[numeric] Data on which to apply the bandwidth selection.
bandwidth	[character(1)] The bandwidth selection method to use. Default is Andrews (1991) ("and"), an alternative is Newey West (1994) ("nw").
kernel	[character(1)] The kernel function to use for selecting the bandwidth. Default is Bartlett kernel ("ba"), see Details for alternatives.
...	Arguments passed to getBandwidthNW.
check	[logical] Whether to check (and if necessary convert) the arguments. See <a href="#">checkVars</a> for further information.
inter	[logical] The first column will be ignored, if TRUE (intercept). Default is FALSE.
u.weights	[numeric] How to weight the columns of u. If NULL (default), uses identical weights for all columns.

**Details**

For Andrews (1991), the AR(1) individual version is implemented.

The kernel that is used for calculating the long-run variance can be one of the following:

- "ba": Bartlett kernel
- "pa": Parzen kernel
- "qs": Quadratic Spectral kernel
- "th": Tukey-Hanning kernel (only if bandwidth = "and")
- "tr": Truncated kernel (only if bandwidth = "and")

**Value**

numeric(1) . Bandwidth

**Functions**

- `getBandwidthAnd`: Automatic bandwidth selection of Andrews (1991).
- `getBandwidthNW`: Automatic bandwidth selection of Newey and West (1994).

**References**

- Andrews, D.W.K. (1991): "Heteroskedasticity and Autocorrelation Consistent Covariance Matrix Estimation," *Econometrica*, 59, 817–854, [DOI:10.2307/2938229](https://doi.org/10.2307/2938229).
- Newey, W.K. and K.D. West (1994): "Automatic Lag Selection in Covariance Matrix Estimation", *Review of Economic Studies*, 61, 631–653, [DOI:10.2307/2297912](https://doi.org/10.2307/2297912).



**See Also**[getLongRunVar](#)**Examples**

```

set.seed(1909)
x <- rnorm(100)
getBandwidth(x, kernel = "ba")
getBandwidth(x, bandwidth = "nw", kernel = "ba")

x2 <- arima.sim(model = list(ar = c(0.7, 0.2)), innov = x, n = 100)
getBandwidth(x2, kernel = "qs")
getBandwidth(x2, bandwidth = "nw", kernel = "qs")

```

getLeadLag

*Leads and Lags***Description**

Generates "optimal" numbers of leads and lags for the Dynamic OLS estimator.

**Usage**

```

getLeadLag(x, y, deter, max.lag, max.lead, ic = c("AIC", "BIC"),
  symmet = FALSE, check = FALSE)

```

**Arguments**

x	[numeric   matrix   data.frame] RHS variables on which to apply the D-OLS estimation (see Details).
y	[numeric   matrix   data.frame] LHS variable(s) on which to apply the D-OLS estimation (see Details). Has to be one-dimensional. If matrix, it may have only one row or column, if data.frame just one column.
deter	[numeric   matrix   data.frame   NULL] Deterministic variable to include in the equation (see Details). If it's NULL or missing, no deterministic variable is included in the model.
max.lead, max.lag	[numeric(1)] Maximal numbers of leads and lags, have to be non-negative integer values.
ic	[character(1)] Information criterion (one of "AIC" or "BIC").
symmet	[logical(1)] If TRUE, only looks for equal leads and lags.
check	[logical] Whether to check (and if necessary convert) the arguments. See <a href="#">checkVars</a> for further information.

**Value**

numeric(2) . "Optimal" numbers of leads and lags.

**See Also**

Other D-OLS: [cointRegD](#), [getModD](#), [makeLeadLagMatrix](#)

**Examples**

```
set.seed(1909)
y <- matrix(cumsum(rnorm(100)), ncol = 1)
x <- matrix(rep(y, 4) + rnorm(400, mean = 3, sd = 2), ncol = 4)
deter <- cbind(1, 1:100)
cointReg::getLeadLag(x = x, y = y, deter = deter, max.lag = 5,
                    max.lead = 5, ic = "AIC", symmet = FALSE)
```

---

getLongRunVar

*Long-Run Variance*

---

**Description**

This function computes the long-run variance Omega, the one sided long-run variance Delta (starting with lag 0) and the variance Sigma from an input matrix of residuals.

**Usage**

```
getLongRunVar(u, bandwidth = c("and", "nw"), kernel = c("ba", "bo", "da",
"pa", "qs", "tr"), demeaning = FALSE, check = TRUE, ...)
```

**Arguments**

u	[numeric   matrix] Data on which to apply the calculation of the long-run variance.
bandwidth	[numeric(1)] The bandwidth to use for calculating the long-run variance as a positive intergerish value.
kernel	[character(1)] The kernel function to use for selecting the bandwidth. Default is Bartlett kernel ("ba"), see Details for alternatives.
demeaning	[logical] Demeaning of the data before the calculation (default is FALSE).
check	[logical] Whether to check (and if necessary convert) the arguments. See <a href="#">checkVars</a> for further information.
...	Arguments passed to getBandwidthNW.

**Details**

The bandwidth can be one of the following:

- "ba": Bartlett kernel
- "bo": Bohmann kernel
- "da": Daniell kernel
- "pa": Parzen kernel
- "qs": Quadratic Spectral kernel
- "tr": Truncated kernel

**Value**

list with components:

Omega [matrix ] Long-run variance matrix

Delta [matrix ] One-sided long-run variance matrix

Sigma [matrix ] Variance matrix

**See Also**

[getBandwidth](#)

**Examples**

```
set.seed(1909)
x <- rnorm(100)
band <- getBandwidthAnd(x, kernel = "ba")
getLongRunVar(x, kernel = "ba", bandwidth = band)
# shorter:
getLongRunVar(x, kernel = "ba", bandwidth = "and")

x2 <- arima.sim(model = list(ar = c(0.7, 0.2)), innov = x, n = 100)
x2 <- cbind(a = x2, b = x2 + rnorm(100))
getLongRunVar(x2, kernel = "ba", bandwidth = "nw")
```

---

getLongRunWeights      *Weights for Long-Run Variance*

---

**Description**

Compute the weights corresponding to some kernel functions.

**Usage**

```
getLongRunWeights(n, bandwidth, kernel)
```

**Arguments**

n	[numeric(1)] Length of weights' vector.
bandwidth	[numeric(1)] The bandwidth (as number).
kernel	[character(1)] The kernel function (see <a href="#">getLongRunVar</a> for possible values).

**Value**

list with components:

w [numeric ]	Vector of weights
upper [numeric(1) ]	Index to largest non-zero entry in w

**See Also**

[getLongRunVar](#)

**Examples**

```
lrw.ba = cointReg:::getLongRunWeights(100, kernel = "ba", bandwidth = 25)
plot(lrw.ba$w)
```

---

getModD

*Get D OLS model.*

---

**Description**

Generates an lm model for the Dynamic OLS estimator.

**Usage**

```
getModD(x, y, deter, n.lag, n.lead, check = FALSE)
```

**Arguments**

x	[matrix] RHS variables of the D OLS estimation.
y	[matrix] LHS variable(s) of the D OLS estimation.
deter	[matrix] Matrix of deterministic variables to include in the D OLS model.
n.lag, n.lead	[numeric(1)] Number of lags and leads, have to be non-negative integer values.

check [logical]  
Whether to check (and if necessary convert) the arguments. See [checkVars](#) for further information.

### Value

lm . An `lm` object, containing an additional list element (`aux`) with D-OLS specific objects:

Z [matrix ] jointed matrix of deterministic and x  
 x.delta [matrix ] differences of x  
 dx.all [matrix ] leads-and-lags matrix  
 all.trunc [matrix ] truncated version of jointed matrix of Z and dx.all  
 y.trunc [matrix ] truncated version of y

### See Also

Other D-OLS: [cointRegD](#), [getLeadLag](#), [makeLeadLagMatrix](#)

### Examples

```
set.seed(1909)
y <- matrix(cumsum(rnorm(100)), ncol = 1)
x <- matrix(rep(y, 4) + rnorm(400, mean = 3, sd = 2), ncol = 4)
deter <- cbind(1, 1:100)
cointReg::getModD(x = x, y = y, deter = deter, n.lag = 2, n.lead = 3)
```

---

makeLeadLagMatrix      *Leads-and-Lags Matrix*

---

### Description

Generates leads-and-lags matrix for the Dynamic OLS estimator.

### Usage

```
makeLeadLagMatrix(x, n.lag, n.lead)
```

### Arguments

x [matrix]  
Matrix for which to generate the leads-and-lags matrix.

n.lag, n.lead [numeric(1)]  
Number of lags and leads, have to be non-negative integer values. If greater than `nrow(x)`, produces 0-rows.

### Value

matrix . Leads-and-lags matrix.

**See Also**

Other D-OLS: [cointRegD](#), [getLeadLag](#), [getModD](#)

**Examples**

```
x <- matrix(1:20, 2, byrow = TRUE)
cointReg:::makeLeadLagMatrix(x = x, n.lag = 2, n.lead = 3)
```

---

plot.cointReg                      *Plot Method for Cointegration Models (Modified OLS).*

---

**Description**

Plotting objects of class "cointReg". Currently, only the residuals will be plotted.

**Usage**

```
## S3 method for class 'cointReg'
plot(x, type, main, xlab, ylab, axes = TRUE, ...)
```

**Arguments**

x	[cointReg] Object of class "cointReg", i.e. the result of <a href="#">cointRegFM</a> , <a href="#">cointRegD</a> , or <a href="#">cointRegIM</a> .
type	[character] Plot type (from <a href="#">plot</a> ). Default is "l".
main, xlab, ylab	[character] Title and axis titles (from <a href="#">plot</a> ). Default values will be generated from the contents of x.
axes	[logical] Whether to add axes (from <a href="#">plot</a> ) to the plot.
...	[any] Further arguments passed to <a href="#">plot</a> .

**See Also**

Other cointReg: [cointRegD](#), [cointRegFM](#), [cointRegIM](#), [cointReg](#), [print.cointReg](#)

**Examples**

```

set.seed(42)
x = data.frame(x1 = cumsum(rnorm(200)), x2 = cumsum(rnorm(200)))
eps1 = rnorm(200, sd = 2)
y = x$x1 - x$x2 + 10 + eps1
deter = cbind(level = rep(1, 200))
test = cointRegFM(x = x, y = y, deter = deter)
plot(test)

```

---

```

print.cointReg      Print Method for Cointegration Models (Modified OLS).

```

---

**Description**

Printing objects of class "cointReg".

**Usage**

```

## S3 method for class 'cointReg'
print(x, ..., digits = getOption("digits"),
      all.coeffs = FALSE)

```

**Arguments**

x	[cointReg] Object of class "cointReg", i.e. the result of cointRegFM, cointRegD or cointRegIM.
...	ignored
digits	[numeric] Number of significant digits to be used.
all.coeffs	[logical] Whether to show all coefficients (i. e. the "real" regressors AND the auxiliary regressors). Default is FALSE.

**Value**

The invisible x object.

**See Also**

Other cointReg: [cointRegD](#), [cointRegFM](#), [cointRegIM](#), [cointReg](#), [plot.cointReg](#)

**Examples**

```
set.seed(42)
x = data.frame(x1 = cumsum(rnorm(200)), x2 = cumsum(rnorm(200)))
eps1 = rnorm(200, sd = 2)
y = x$x1 - x$x2 + 10 + eps1
deter = cbind(level = rep(1, 200))

test.fm = cointRegFM(x = x, y = y, deter = deter)
print(test.fm)

test.d = cointRegD(x = x, y = y, deter = deter)
print(test.d)

test.im2 = cointRegIM(x = x, y = y, deter = deter)
print(test.im2)
```



# Index

checkDoptions, [3](#), [3](#), [5](#), [6](#)  
checkObject, [3](#), [4](#), [4](#), [6](#)  
checkVars, [3–5](#), [6](#), [8](#), [9](#), [12](#), [14](#), [16–18](#), [21](#)  
cointReg, [3](#), [7](#), [11](#), [13](#), [15](#), [22](#), [23](#)  
cointReg-package, [2](#)  
cointRegD, [3](#), [7](#), [8](#), [9](#), [13](#), [15](#), [18](#), [21–23](#)  
cointRegFM, [3](#), [7](#), [8](#), [11](#), [11](#), [15](#), [22](#), [23](#)  
cointRegIM, [3](#), [7](#), [8](#), [11](#), [13](#), [13](#), [22](#), [23](#)

getBandwidth, [3](#), [15](#), [19](#)  
getBandwidthAnd, [3](#)  
getBandwidthAnd (getBandwidth), [15](#)  
getBandwidthNW, [3](#), [9](#), [12](#), [14](#)  
getBandwidthNW (getBandwidth), [15](#)  
getLeadLag, [3](#), [10](#), [11](#), [17](#), [21](#), [22](#)  
getLongRunVar, [3](#), [9](#), [12](#), [17](#), [18](#), [20](#)  
getLongRunWeights, [3](#), [19](#)  
getModD, [3](#), [11](#), [18](#), [20](#), [22](#)

lm, [21](#)

makeLeadLagMatrix, [3](#), [11](#), [18](#), [21](#), [21](#)

plot, [3](#), [22](#)  
plot.cointReg, [8](#), [11](#), [13](#), [15](#), [22](#), [23](#)  
print, [3](#)  
print.cointReg, [8](#), [11](#), [13](#), [15](#), [22](#), [23](#)