Package ‘apt’

February 25, 2016

Title Asymmetric Price Transmission

Version 2.5

Date 2016-2-25

Author Changyou Sun <cs258@msstate.edu>

Maintainer Changyou Sun <cs258@msstate.edu>

Depends R (>= 3.0.0), erer, gWidgets

Imports car, urca, copula

Suggests RGtk2, gWidgetsRGtk2, cairoDevice

Description
Asymmetric price transmission between two time series is assessed. Several functions are available for linear and nonlinear threshold cointegration, and furthermore, symmetric and asymmetric error correction model. A graphical user interface is also included for major functions included in the package, so users can also use these functions in a more intuitive way.

LazyLoad yes

NeedsCompilation no

Repository CRAN

Date/Publication 2016-02-25 23:01:55

R topics documented:

`apt-package` *Asymmetric Price Transmission*

Description

Asymmetric price transmission between two time series is assessed. The names of functions and datasets reveal the categories they belong to. A prefix of `da` is for datasets, `ci` for cointegration, and `ecm` for error correction model.

The focus is on the price transmission between *two* price variables. Therefore, objectives like fitting an error correction model for more than two variables are beyond the scope of this package.

As a teaching demo, a graphical user interface is also developed for the main functions in this package. Several packages are needed to run the two included GUIs: `guiCor` and `guiApt`. See the note at `guiCor` for installation detail.

Details

Package:	apt
Type:	Package
Version:	2.5
Date:	2016-2-25
Depends:	R (>= 3.0.0), erer, gWidgets
Imports:	car, urca, copula
Suggests:	RGtk2, gWidgetsRGtk2, cairoDevice
License:	GPL
LazyLoad:	yes

Author(s)

Changyou Sun (`<cs258@msstate.edu>`)

`ciTarFit` *Fitting Threshold Cointegration*

Description

Fit a threshold cointegration regression between two time series.

Usage

```
ciTarFit(y, x, model = c('tar','mtar'), lag = 1, thresh = 0,
  small.win = NULL)
```

Arguments

`y`	dependent or left-side variable for the long-run model; must be a time series object.
`x`	independent or right-side variable for the long-run model; must be a time series object.
`model`	a choice of two models: `tar` or `mtar`; the default is `tar`.
`lag`	number of lags for the threshold cointegration regression.
`thresh`	a threshold value (default of zero).
`small.win`	value of a small window for fitting the threshold cointegration regression; used mainly for lag selection in `ciTarLag`.

Details

This is the main function for threshold autoregression regression (TAR) in assessing the nonlinear threshold relation between two time series variables. It can be used to estimate four types of threshold cointegration regressions. These four types are TAR with a threshold value of zero; consistent TAR with a nonzero threshold; MTAR (momentum TAR) with a threshold value of zero; and consistent MTAR with a nonzero threshold. The option of small window will be used in lag selection because a comparison of AIC and BIC values should be based on the same number of regression observations.

Value

Return a list object of class `"ciTarFit"` with these components:

`y`	dependend variable
`x`	independent variable
`model`	model choice
`lag`	number of lags
`thresh`	threshold value
`data.LR`	data used in the long-run regression
`data.CI`	data used in the threshold cointegration regression
`z`	residual from the long-run regression
`lz`	lagged residual from the long-run regression
`ldz`	lagged residual with 1st difference from long-run model
`LR`	long-run regression

CI	threshold cointegration regression
f.phi	test with a null hypothesis of no threshold cointegration
f.apt	test with a null hypothesis of no asymmetric price transmission in the long run
sse	value of sum of squared errors
aic	value of Akaike Information Criterion
bic	value of Bayesian Information Criterion.

Methods

One method is defined as follows:

print: Four main outputs from threshold cointegration regression are shown: long-run regression between the two price variables, threshold cointegration regression, hypothesis test of no cointegration, and hypothesis test of no asymmetric adjustment.

Author(s)

Changyou Sun (<cs258@msstate.edu>)

References

Balke, N.S., and T. Fomby. 1997. Threshold cointegration. International Economic Review 38(3):627-645.

Enders, W., and C.W.J. Granger. 1998. Unit-root tests and asymmetric adjustment with an example using the term structure of interest rates. Journal of Business & Economic Statistics 16(3):304-311.

Enders, W., and P.L. Siklos. 2001. Cointegration and threshold adjustment. Journal of Business and Economic Statistics 19(2):166-176.

See Also

summary.ciTarFit; ciTarLag for lag selection; and ciTarThd for threshold selection.

Examples

```
# see example at daVich
```

---

ciTarLag *Lag Selection for Threshold Cointegration Regression*

Description

Select the best lag for threshold cointegration regression by AIC and BIC

Usage

```
ciTarLag(y, x, model = c("tar", "mtar"), maxlag = 4,
thresh = 0, adjust = TRUE)
```

Arguments

`y`	dependent or left-side variable for the long-run regression.
`x`	independent or right-side variable for the long-run regression.
`model`	a choice of two models, either tar or mtar.
`maxlag`	maximum number of lags allowed in the search process.
`thresh`	a threshold value.
`adjust`	logical value (default of TRUE) of whether to adjust the window widths so all regressions by lag have the same number of observations

Details

Estimate the threshold cointegration regressions by lag and then select the best regression by AIC or BIC value. The longer the lag, the smaller the number of observations availabe for estimation. If the windows of regressions by lag are not ajusted, the maximum lag is usually the best lag by AIC or BIC. Theorectially, AIC and BIC from different models should be compared on the basis of the same observation numbers (Ender 2004). `adjust` shows the effect of this adjustment on the estimation window. By default, the value of `adjust` should be TRUE.

Value

Return a list object of class "ciTarLag" with the following components:

`path`	a data frame of model criterion values by lag, including `lag` for the current lag, `totObs` for total observations in the raw data, `coinObs` for observations used in the cointegration regression, `sse` for the sum of squared errors, `aic` for AIC value, `bic` for BIC value, `LB4` for the p-value of Ljung_Box Q statistic with 4 autocorrelation coefficients, `LB8` with 8 coefficients, `LB12` for Q statistic with 12 coefficients
`out`	a data frame of the final model selection, including the values of model, maximum lag, threshold value, best lag by AIC, best lag by BIC

Methods

Two methods are defined as follows:

`print`**:** This shows the `out` component in the returned list.

`plot`**:** This demonstrates the trend of AIC and BIC changes of threshold cointegration regressions by lag. It facilitates the selection of the best lag for a threshold cointegration model.

Author(s)

Changyou Sun (<cs258@msstate.edu>)

References

Enders, W. 2004. Applied Econometric Time Series. John Wiley & Sons, Inc., New York. 480 P.

Enders, W., and C.W.J. Granger. 1998. Unit-root tests and asymmetric adjustment with an example using the term structure of interest rates. Journal of Business & Economic Statistics 16(3):304-311.

See Also

ciTarFit; and ciTarThd;

Examples

```
# see example at daVich
```

---

ciTarThd *Threshold Selection for Threshold Cointegration Regression*

Description

Select the best threshold for threshold cointegration regression by sum of squared errors

Usage

```
ciTarThd(y, x, model = c('tar', 'mtar'), lag = 1,
th.range = 0.15, digits = 3)
```

Arguments

y	dependent or left-side variable for the long-run regression.
x	independent or right-side variable for the long-run regression.
model	a choice of two models, either tar or mtar.
lag	number of lags.
th.range	the percentage of observations to be excluded from the search.
digits	number of digits used in rounding outputs.

Details

The best threshold is determined by fitting the regression for possible threshold values, sorting the results by sum of squared errors (SSE), and selecting the best with the lowest SSE. To have sufficient observations on either side of the threshold value, certain percentage of observations on the top and bottoms are excluded from the search path. This is usually set as 0.15 by the `th.range` (Chan 1993).

Value

Return a list object of class "ciTarThd" with the following components:

model	model choice
lag	number of lags
th.range	the percentage of observations excluded
th.final	the best threshold value
ssef	the best (i.e., lowest) value of SSE

`obs.tot`	total number of observations in the raw data
`obs.CI`	number of observations used in the threshold cointegration regression
`basic`	a brief summary of the major outputs
`path`	a data frame of the search record (number of regression, threshold value, SSE, AIC, and BIC values).

Methods

Two methods are defined as follows:

`print`**:** This shows the `basic` component in the returned list object.

`plot`**:** plotting three graphs in one window; they reveals the relationship between SSE (sum of squared errors), AIC, BIC and the threshold values. The best threshold value is associated with the lowest SSE value.

Author(s)

Changyou Sun (`<cs258@msstate.edu>`)

References

Chan, K.S. 1993. Consistency and limiting distribution of the least squares estimator of a threshold autoregressive model. The Annals of Statistics 21(1):520-533.

Enders, W., and C.W.J. Granger. 1998. Unit-root tests and asymmetric adjustment with an example using the term structure of interest rates. Journal of Business & Economic Statistics 16(3):304-311.

See Also

`ciTarFit`; and `ciTarLag`

Examples

```
# see example at daVich
```

---

daVich *Import prices and values of wooden beds from Vietnam and China*

Description

This data set contains two unit import prices (dollar per piece) and values (million dollars) of wooden beds from Vietnam and China to the United States.

`price.vi`	Monthly price over Janurary 2002 to Janurary 2010 from Vietnam.
`price.ch`	Monthly price over Janurary 2002 to Janurary 2010 from China.
`price.vi`	Monthly value over Janurary 2002 to Janurary 2010 from Vietnam.
`price.ch`	Monthly value over Janurary 2002 to Janurary 2010 from China.

Usage

```
data(daVich)
```

Format

A monthly time series from Janurary 2002 to Janurary 2010 with 97 observations for each of the four series.

Details

Under the Harmonized Tariff Schedule (HTS) system, the commodity of wooden beds is classified as HTS 9403.50.9040. The monthly cost-insurance-freight values in dollar and quantities in piece are reported by country from U.S. ITC (2010). The unit price (dollar per piece) is calculated as the ratio of value over quantity by country.

Source

U.S. ITC, 2010. Interactive tariff and trade data web. US International Trade Commision (Assecced on March 1, 2010).

References

Sun, C. 2011. Price dynamics in the import wooden bed market of the United States. Forest Policy and Economics 13(6): 479-487.

Examples

```
# The following codes reproduce the main results in Sun (2011 FPE).
# All the codes have been tested and should work.

# 1. Data preparation ___________________________________________________________

library(urca); data(daVich)
head(daVich); tail(daVich); str(daVich)
prVi <- y <- daVich[, 1]
prCh <- x <- daVich[, 2]


# 2. EG cointegration  __________________________________________________________

LR <- lm(y ~ x); summary(LR)
(LR.coef <- round(summary(LR)$coefficients, 3))
(ry <-  ts(residuals(LR), start=start(prCh), end=end(prCh), frequency =12))
summary(eg <- ur.df(ry,  type=c("none"), lags=1)); plot(eg)
(eg4  <- Box.test(eg@res, lag = 4, type="Ljung") )
(eg8  <- Box.test(eg@res, lag = 8, type="Ljung") )
(eg12 <- Box.test(eg@res, lag = 12, type="Ljung"))


## Not run:
```

```
# 3. TAR + Cointegration  _______________________________________________________

# best threshold
t3 <- ciTarThd(y=prVi, x=prCh, model="tar", lag=0)
(th.tar <- t3$basic); plot(t3)
for (i in 1:12) {                 # 20 seconds
  t3a <- ciTarThd(y=prVi, x=prCh, model="tar", lag=i)
  th.tar[i+2] <- t3a$basic[,2]
}
th.tar

t4 <- ciTarThd(y=prVi, x=prCh, model="mtar", lag=0); (th.mtar <- t4$basic)
plot(t4)
for (i in 1:12) {
  t4a <- ciTarThd(y=prVi, x=prCh, model="mtar", lag=i)
  th.mtar[i+2] <- t4a$basic[,2]
 }
th.mtar

# The following threshold values are specific to this data. They HAVE TO be
# revised for another data set. Otherwise, various errors will occur.
  t.tar <- -8.041; t.mtar <- -0.451     # lag = 0 to 4
# t.tar <- -8.701 ; t.mtar <- -0.451    # lag = 5 to 12

# lag selection
mx <- 12
(g1 <-ciTarLag(y=prVi, x=prCh, model="tar", maxlag=mx, thresh= 0));     plot(g1)
(g2 <-ciTarLag(y=prVi, x=prCh, model="mtar",maxlag=mx, thresh= 0));     plot(g2)
(g3 <-ciTarLag(y=prVi, x=prCh, model="tar", maxlag=mx, thresh=t.tar));  plot(g3)
(g4 <-ciTarLag(y=prVi, x=prCh, model="mtar",maxlag=mx, thresh=t.mtar)); plot(g4)


# Table 3 Results of EG and threshold cointegration tests
# Note: Some results in Table 3 in the published paper were incorrect because
#  of a mistake made when the paper was done in 2009. I found the mistake when
#  the package was build in later 2010. For example, for the consistent MTAR,
#  the coefficient for the positive term was reported as -0.251 (-2.130) but
#  it should be -0.106 (-0.764), as cacluated from below codes.
#  The main conclusion about the research issue is still qualitatively the same.
vv <- 3
(f1 <- ciTarFit(y=prVi, x=prCh, model="tar",  lag=vv, thresh=0     ))
(f2 <- ciTarFit(y=prVi, x=prCh, model="tar",  lag=vv, thresh=t.tar ))
(f3 <- ciTarFit(y=prVi, x=prCh, model="mtar", lag=vv, thresh=0     ))
(f4 <- ciTarFit(y=prVi, x=prCh, model="mtar", lag=vv, thresh=t.mtar))

r0 <- cbind(summary(f1)$dia, summary(f2)$dia, summary(f3)$dia,
  summary(f4)$dia)
diag <- r0[c(1:4, 6:7, 12:14, 8, 9, 11), c(1,2,4,6,8)]
rownames(diag) <- 1:nrow(diag); diag

e1 <- summary(f1)$out; e2 <- summary(f2)$out
e3 <- summary(f3)$out; e4 <- summary(f4)$out; rbind(e1, e2, e3, e4)
```

```
ee <- list(e1, e2, e3, e4); vect <- NULL
for (i in 1:4) {
  ef <- data.frame(ee[i])
  vect2 <- c(paste(ef[3, "estimate"], ef[3, "sign"], sep=""),
             paste("(", ef[3, "t.value"], ")",       sep=""),
             paste(ef[4, "estimate"], ef[4, "sign"], sep=""),
             paste("(", ef[4, "t.value"], ")",       sep=""))
  vect <- cbind(vect, vect2)
}
item <- c("pos.coeff","pos.t.value", "neg.coeff","neg.t.value")
ve <- data.frame(cbind(item, vect)); colnames(ve) <- colnames(diag)
( res.CI <- rbind(diag, ve)[c(1:2, 13:16, 3:12), ] )
rownames(res.CI) <- 1:nrow(res.CI)


# 4. APT + ECM   ________________________________________________________________

(sem <- ecmSymFit(y=prVi, x=prCh, lag=4)); names(sem)
aem <- ecmAsyFit(y=prVi, x=prCh,lag=4, model="mtar", split=TRUE, thresh=t.mtar)
aem
(ccc <- summary(aem))
coe <- cbind(as.character(ccc[1:19, 2]),
  paste(ccc[1:19, "estimate"], ccc$signif[1:19], sep=""), ccc[1:19, "t.value"],
  paste(ccc[20:38,"estimate"], ccc$signif[20:38],sep=""), ccc[20:38,"t.value"])
colnames(coe) <- c("item", "China.est", "China.t", "Vietnam.est","Vietnam.t")

(edia <- ecmDiag(aem, 3))
(ed <- edia[c(1,6,7,8,9), ])
ed2 <- cbind(ed[,1:2], "_", ed[,3], "_")
colnames(ed2) <- colnames(coe)

(tes <- ecmAsyTest(aem)$out)
(tes2 <- tes[c(2,3,5,11,12,13,1), -1])
tes3 <- cbind(as.character(tes2[,1]),
  paste(tes2[,2], tes2[,6], sep=''), paste("[", round(tes2[,4],2), "]", sep=''),
  paste(tes2[,3], tes2[,7], sep=''), paste("[", round(tes2[,5],2), "]", sep=''))
colnames(tes3) <- colnames(coe)

(coe  <- data.frame(apply(coe, 2, as.character), stringsAsFactors=FALSE))
(ed2  <- data.frame(apply(ed2, 2, as.character), stringsAsFactors=FALSE))
(tes3 <- data.frame(apply(tes3,2, as.character), stringsAsFactors=FALSE))
table.4 <- data.frame(rbind(coe, ed2, tes3))
options(width=150); table.4; options(width=80)


## End(Not run)
```

ecmAsyFit *Fitting Asymmetric Error Correction Model*

Description

Estimate an asymmetric error correction model (ECM) for two time series.

Usage

```
ecmAsyFit(y, x, lag = 1, split = TRUE,
  model = c("linear", "tar", "mtar"), thresh = 0)
```

Arguments

`y`	dependent or left-side variable for the long-run regression.
`x`	independent or right-side variable for the long-run regression.
`lag`	number of lags for variables on the right side.
`split`	a logical value (default of TRUE) of whether the right-hand variables should be split into positive and negative parts.
`model`	a choice of three models: linear, tar , or mtar cointegration.
`thresh`	a threshold value; this is only required when the model is specified as 'tar' or 'mtar.'

Details

There are two specficiations of an asymmetric ECM. The first one is how to calculate the error correction terms. One way is through linear two-step Engle Granger approach, as specificied by `model="linear"`. The other two ways are threshold cointegration by either 'tar' or 'mtar' with a threshold value. The second specification is related to the possible asymmetric price transmission in the lagged price variables, as specified in `split = TRUE`. Note that the linear cointegration specification is a special case of the threshold cointegration. A model with `model="linear"` is the same as a model with `model="tar", thresh = 0`.

Value

Return a list object of class "ecm" and "ecmAsyFit" with the following components:

`y`	dependend variable
`x`	independent variable
`lag`	number of lags
`split`	logical value of whether the right-hand variables are split
`model`	model choice
`IndVar`	data frame of the right-hand variables used in the ECM
`name.x`	name of the independent variable
`name.y`	name of the dependent variable
`ecm.y`	ECM regression for the dependent variable
`ecm.x`	ECM regression for the independent variable
`data`	all the data combined for the ECM
`thresh`	thresh value for TAR and MTAR model

Methods

Two methods are defined as follows:

`print`: showing the key outputs.

`summary`: summarizing thekey outputs.

Author(s)

Changyou Sun (<cs258@msstate.edu>)

References

Enders, W., and C.W.J. Granger. 1998. Unit-root tests and asymmetric adjustment with an example using the term structure of interest rates. Journal of Business & Economic Statistics 16(3):304-311.

See Also

`print.ecm`; `summary.ecm`; `ecmDiag`; and `ecmAsyTest`.

Examples

```
# see example at daVich
```

`ecmAsyTest` *Hypothesis Tests on Asymmetric Error Correction Model*

Description

Conduct several F-tests on the coefficients from asymmetric ECM.

Usage

```
ecmAsyTest(w, digits = 3)
```

Arguments

`w`	an object of 'ecmAsyFit' class.
`digits`	number of digits used in rounding outputs.

Details

There are two ECM equations for the two price series. In each equation, four types of hypotheses are tested; equilibrium adjustment path symmetry on the error correction terms (H1), Granger causality test (H2), distributed lag symmetry at each lag (H3), and cumulative asymmetry of all lags (H4). The latter two tests are only feasible and availabe for models with split variables. The number of H3 tests is equal to the number of lags.

Value

Return a list object with the following components:

H1ex	H01 in equation x: equilibrium adjustment path symmetry
H1ey	H01 in equation y: equilibrium adjustment path symmetry
H2xx	H02 in equation x: x does not Granger cause x
H2yx	H02 in equation y: x does not Granger cause y
H2xy	H02 in equation x: y does not Granger cause x
H2yy	H02 in equation y: y does not Granger cause y
H3xx	H03 in equation x: distributed lag symmetry of x at each lag
H3yx	H03 in equation y: distributed lag symmetry of x at each lag
H3xy	H03 in equation x: distributed lag symmetry of y at each lag
H3yy	H03 in equation y: distributed lag symmetry of y at each lag
H4xx	H04 in equation x: cumulative asymmetry of x for all lags
H4yx	H04 in equation y: cumulative asymmetry of x for all lags
H4xy	H04 in equation x: cumulative asymmetry of y for all lags
H4yy	H04 in equation y: cumulative asymmetry of y for all lags
out	summary of the four types of hypothesis tests

Methods

One method is are defined as follows:

print: This shows the out component in the returned list object.

Author(s)

Changyou Sun (<cs258@msstate.edu>)

References

Frey, G., and M. Manera. 2007. Econometric models of asymmetric price transmission. Journal of Economic Surveys 21(2):349-415.

See Also

ecmAsyFit and ecmDiag.

Examples

```
# see example at daVich
```

ecmDiag *Diagnostic Statitics for Symmetric or Asymmetric ECMs*

Description

Report a set of diagnostic statistics for symmetric or asymmetric error correction models

Usage

```
ecmDiag(m, digits = 2)
```

Arguments

m	an object of class ecm from the function of ecmAsyFit or ecmSymFit.
digits	number of digits used in rounding outputs.

Details

Compute several diagnostic statistics for each ECM equation. This is mainly used to assess the serial correlation in the residuals and model adequacy.

Value

Return a data frame object with the following components by equation: R-squared, Adjusted R-squared, F-statistic, Durbin Watson statistic, p-value for DW statistic, AIC, BIC, and p-value of Ljung_Box Q statistics with 4, 8, 12 autocorrelation coefficients.

Author(s)

Changyou Sun (<cs258@msstate.edu>)

References

Enders, W. 2004. Applied Econometric Time Series. John Wiley & Sons, Inc., New York. 480 P.

See Also

ecmAsyFit; ecmSymFit; and ecmDiag.

Examples

```
# see example at daVich
```

`ecmSymFit` *Fitting symmetric Error Correction Model*

Description

Estimate a symmetric error correction model (ECM) for two time series.

Usage

```
ecmSymFit(y, x, lag = 1)
```

Arguments

`y`	dependent or left-side variable for the long-run regression.
`x`	independent or right-side variable for the long-run regression.
`lag`	number of lags for variables on the right side.

Details

The package `apt` focuses on price transmission between two series. This function estimates a standard error correction model for two time series. While it can be extended for more than two series, it is beyond the objective of the package now.

Value

Return a list object of class "ecm" and "ecmSymFit" with the following components:

`y`	dependend variable
`x`	independent variable
`lag`	number of lags
`data`	all the data combined for the ECM
`IndVar`	data frame of the right-hand variables used in the ECM
`name.x`	name of the independent variable
`name.y`	name of the dependent variable
`ecm.y`	ECM regression for the dependent variable
`ecm.x`	ECM regression for the independent variable

Author(s)

Changyou Sun (`<cs258@msstate.edu>`)

References

Enders, W. 2004. Applied Econometric Time Series. John Wiley & Sons, Inc., New York. 480 P.

See Also

`print.ecm`; `summary.ecm`; `ecmDiag`; and `ecmAsyFit`.

Examples

```
# see example at daVich
```

---

guiApt *Graphical User Interface for the apt Package*

Description

This graphical user interface (GUI) contains most functions in the `apt` package. Users can call these functions on this interface.

Usage

```
guiApt(pos = 1)
```

Arguments

`pos`	specifying where the seven tabs should be placed: bottom (`pos = 1`; default), left (`pos = 2`), top (`pos = 3`), or right (`pos = 4`).

Details

The `gWidgets` package is used to create this GUI. The objective in this application is to develop a GUI for the `apt` package and allow users to call individual functions on the interface and conduct threshold cointegration analysis in an intuitive way. The GUI has menus and toolbars at the top. Then seven tabbed notebook container widgets are used to hold messages and individual function calls.

The inputs are two single time series data, and they should be processed within R first before a GUI can be used for analyses. Outputs can be displayed within the interface, but for simplicity, the R console is used to display all outputs in this application.

See `guiCor` for package installation issues.

Value

No value returnd from this function call. A GUI is generated as the side effect.

Author(s)

Changyou Sun (<`cs258@msstate.edu`>)

References

Lawrence, M.F., and J. Verzani. 2012. Programming graphical user interfaces in R. Ed. CRC Press. 466 P.

See Also

guiCor

Examples

```
## Not run:
  library(RGtk2); library(gWidgetsRGtk2)
  options(guiToolkit = "RGtk2")  # may need this for some computers

  data(daVich); prVi <- daVich[, 1] ; prCh <- daVich[, 2]
  guiApt()
  guiApt(2) # tabs on the left side

## End(Not run)
```

---

guiCor *Graphical User Interface for Showing Variable Correlation*

Description

This graphical user interface (GUI) can show the correlation between two random variables with normal copula distribution.

Usage

```
guiCor()
```

Details

The gWidgets package is used to create this GUI. In this application, two random variables are generated from R and then plotted on a graph. Their correlation changes with each random sample. When many plots are drawn and compared, the change in correlation pattern can be revealed. To make the pattern more predictable, we need a random number generator to draw variables randomly and consistently. The rcoupla() function in the copula package is a good choice for this objective. The relation between the variables is specified through a normal copula, as defined by normalCopula(). The correlation between two normally distributed variables is controlled by a single parameter with the value range [-1, 1]. In showing these pairs of variables on the screen, a number of graphics parameter values can also be manipulated to improve the presentation, e.g., the size and color of points.

Two versions of the graph are considered. In the static version, the goal is to show one plot each time. The appearance of the plot can be affected by four choices: the number of points, their color, size, and the correlation parameter between the two variables. In the dynamic version, a for loop is employed to show many plots continuously. Only one choice is left for users, i.e., the number of plots. Other choices are fixed: the number of points is 1,000, the point size is 3, and the color is changed automatically with each plot.

Important installation notes: Several packages are needed to run this GUI: gWidgets, RGtk2, cairoDevice, and gWidgetsRGtk2. gWidgets is on the "Depends" list so it will be installed with apt together. RGtk2, cairoDevice, and gWidgetsRGtk2 are on the "Suggests" list so they will not be installed by default. Users can install them individually, when they want to use guiCor and guiApt.

RGtk2 should be installed first. It can be challenging to install this package. I am using Microsoft Windows system so the following note is specifically for Windows users. First, install this package by install.packages("RGtk2"). Then load/attach it by library(RGtk2). When a small window pops up and ask you to install GTK+, choose yes. Then shut down the whole R program, restart it, and load/attach this library again. Install other packages, i.e., cairoDevice and gWidgetsRGtk2. Finally, install apt. Submit library(apt); help("guiCor"), and copy the sample codes below to initiate the GUI. If you can go through the process as described, then that is perfect.

Many problems may arise in loading and attaching the RGk2 package, depending on the operating system and hardware on a computer. A typical and annoying problem is that the computer may ask you to installl the GTK+ again and again. This is because the RGk2 package cannot find the GTK+ software, or the GTK+ installed on your computer does not meet the requirement (e.g., 32 or 64 bit). GTK+ is an outside and independent software program and it has many versions. A typical error message is like this: "Failed to load RGtk2 dynamic library." If that is the case, read the error message carefully and download the suggested version of GTK+ manually from the internet, unzip it, and place all the folders in one place on your local drive. Then add a directory in the path of environmental variable through the control panel on your computer. For example, I download a zip file with the name of gtk+-bundle_2.22.1-20101229_win64.zip. The following directory is added and recognized: C:\CSprogram\myRsoftware\gtk22211win64\bin.

In sum, running the GUI needs the toolkit of GTK+, which is independent from R. It needs to be installed on a local drive and recognized by R. The package of rattle faces similar challenges as it uses this toolkit too. Search the internet for similar problems and solutions.

Value

No value returnd from this function call. A GUI is generated as the side effect.

Author(s)

Changyou Sun (<cs258@msstate.edu>)

References

Lawrence, M.F., and J. Verzani. 2012. Programming graphical user interfaces in R. Ed. CRC Press. 466 P.

Nelsen, R.B. 2006. An Introduction to Copulas. 2nd Ed. Springer. 269 P.

See Also

guiApt

Examples

```
## Not run:
  library(copula)
  library(RGtk2)
  library(cairoDevice)
```

```
    library(gWidgetsRGtk2)
    options(guiToolkit = "RGtk2")  # may need this for some computers

    guiCor()

## End(Not run)
```

---

print.ecm *Printing Outputs from Error Correction Models*

Description

Show main outputs from symmetric `ecmSymFit` or asymmetric `ecmAsyFit` error correction models.

Usage

```
## S3 method for class 'ecm'
print(x, ...)
```

Arguments

`x`	an object of class `ecm` from the function of `ecmAsyFit` or `ecmSymFit`.
`...`	additional arguments to be passed.

Details

This is the print method for `ecmAsyFit` or `ecmSymFit` to show main model outputs.

Value

Summary results of the two ECM equations are shown for the two focal price series.

Author(s)

Changyou Sun (<cs258@msstate.edu>)

See Also

`ecmSymFit` and `ecmAsyFit`.

Examples

```
# see example at daVich
```

summary.ciTarFit *Summary of Results from Threshold Cointegration Regression*

Description

This summarizes the main results from threshold cointegration regression.

Usage

```
## S3 method for class 'ciTarFit'
summary(object, digits=3, ...)
```

Arguments

`object`	an object of class 'ciTarFit'.
`digits`	number of digits for rounding outputs.
`...`	additional arguments to be passed.

Details

This wraps up the outputs from threshold cointegration regression in two data frames, one for diagnostic statistics and the other for coeffients.

Value

A list with two data frames. `dia` contains the main model specifications and hypothesis test results. `out` contains the regression results for both the long run (LR) and threshold cointegration (CI).

Author(s)

Changyou Sun (<`cs258@msstate.edu`>)

See Also

`ciTarFit`.

Examples

```
# see example at daVich
```

summary.ecm	*Summary of Results from Error Correction Model*

Description

This summarizes the main results from error correction models.

Usage

```
## S3 method for class 'ecm'
summary(object, digits=3, ...)
```

Arguments

`object`	an object of class `ecm` from the function of `ecmAsyFit` or `ecmSymFit`.
`digits`	number of digits for rounding outputs
`...`	additional arguments to be passed.

Details

This wraps up the coefficents and statistics from ECM by equation.

Value

A data frame object with coefficients and related statistics by equation.

Author(s)

Changyou Sun (`<cs258@msstate.edu>`)

See Also

`ecmSymFit` and `ecmAsyFit`.

Examples

```
# see example at daVich
```

Index