Package ‘ConvergenceClubs’

October 12, 2022

Title Finding Convergence Clubs

Description Functions for clustering regions that form convergence clubs, according to the definition of Phillips and Sul (2009) <doi:10.1002/jae.1080>. A package description is available in Sichera and Pizzuto (2019).

Version 2.2.4

Date 2022-06-13

URL https://CRAN.R-project.org/package=ConvergenceClubs

BugReports https://github.com/rhobis/ConvergenceClubs/issues

Depends R (>= 4.0.0)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Imports lmtest (>= 0.9-35), sandwich (>= 2.3-4)

NeedsCompilation no

Author Roberto Sichera [aut, cre, cph],
Pietro Pizzuto [aut]

Maintainer Roberto Sichera <rob.sichera@gmail.com>

Repository CRAN

Date/Publication 2022-06-13 22:00:24 UTC

R topics documented:

`ConvergenceClubs-package`

ConvergenceClubs: Finding Convergence Clubs

Description

Functions for clustering regions that form convergence clubs, according to the definition of Phillips and Sul (2009) <doi:10.1002/jae.1080>. A package description is available in Sichera and Pizzuto (2019).

Main functions

The package's main functions are `findClubs` and `mergeClubs`. The former finds clubs of convergence, given a dataset with units in rows and years in columns, returning an object of class `convergence.clubs`. The latter takes as argument an object of class `convergence.clubs` and applies the clustering procedure to the convergence clubs contained in the argument, according to either Phillips and Sul (2009) or von Lyncker and Thoennessen (2017) procedure.

Author(s)

Maintainer: Roberto Sichera `<rob.sichera@gmail.com>` [copyright holder]

Authors:

- Pietro Pizzuto `<pietro.pizzuto02@unipa.it>`

References

Andrews, D. W., 1991. Heteroskedasticity and autocorrelation consistent covariance matrix estimation. Econometrica: Journal of the Econometric Society, 817-858.

Phillips, P. C.; Sul, D., 2007. Transition modeling and econometric convergence tests. Econometrica 75 (6), 1771-1855.

Phillips, P. C.; Sul, D., 2009. Economic transition and growth. Journal of Applied Econometrics 24 (7), 1153-1185.

von Lyncker, K.; Thoennessen, R., 2017. Regional club convergence in the EU: evidence from a panel data analysis. Empirical Economics 52 (2), 525-553

Sichera, R.; Pizzuto, P., 2019. ConvergenceClubs: A Package for Performing the Phillips and Sul's Club Convergence Clustering Procedure. The R Journal.

See Also

Useful links:

- https://CRAN.R-project.org/package=ConvergenceClubs
- Report bugs at https://github.com/rhobis/ConvergenceClubs/issues

---

`computeH` *Compute H values*

Description

Computes H values (cross-sectional variance) according to the clustering algorithm by Phillips and Sul (2007, 2009)

Usage

```
computeH(X, quantity = "H", id)
```

Arguments

`X`	matrix or dataframe containing data (preferably filtered, in order to remove business cycles)
`quantity`	string indicating the quantity that should be returned. The options are `"H"`, the default, only the vector of cross-sectional variance is returned; `"h"`, only the matrix of transition path h is return; `"both"`, a list containing both h and H is returned.
`id`	optional; row index of regions for which H values are to be computed; if missing, all regions are used

Details

The cross sectional variation H_{it} is computed as the quadratic distance measure for the panel from the common limit and under the hypothesis of the model should converge to zero as *t* tends towards infinity:

$$H_t = N^{-1} \sum_{i=1}^{N} (h_{it} - 1)^2 \to 0, \quad t \to \infty$$

where

$$h_{it} = \frac{\log y_{it}}{(N^{-1} \sum_{i=1}^{N} log\, y_{it})}$$

Value

A numeric vector, a matrix or a list, depending on the value of `quantity`

References

Phillips, P. C.; Sul, D., 2007. Transition modeling and econometric convergence tests. Econometrica 75 (6), 1771-1855.

Phillips, P. C.; Sul, D., 2009. Economic transition and growth. Journal of Applied Econometrics 24 (7), 1153-1185.

Examples

```
data("filteredGDP")

h <- computeH(filteredGDP[,-1], quantity="h")
H <- computeH(filteredGDP[,-1], quantity="H")
b <- computeH(filteredGDP[,-1], quantity="both")
```

---

coreG *Find core (primary) group*

Description

Find the Core (primary) group according to step 2 of the clustering algorithm by Phillips and Sul (2007, 2009)

Usage

```
coreG(
  X,
  dataCols,
  time_trim,
  threshold = -1.65,
  HACmethod = c("FQSB", "AQSB"),
  type = c("max", "all")
)
```

Arguments

`X`	dataframe containing data (preferably filtered data in order to remove business cycles). Data must not contain any NA or NaN values, otherwise the clustering procedure will be stopped with an error.
`dataCols`	integer vector with the column indices of the data
`time_trim`	a numeric value between 0 and 1, representing the portion of time periods to trim when running log t regression model. Phillips and Sul (2007, 2009) suggest to discard the first third of the period.
`threshold`	numeric value indicating the threshold to be used to perform the one-tail t test; default is -1.65.

`HACmethod` string indicating whether a Fixed Quadratic Spheric Bandwidth (`HACmethod="FQSB"`) or an Adaptive Quadratic Spheric Bandwidth (`HACmethod="AQSB"`) should be used for the truncation of the Quadratic Spectral kernel in estimating the *log t* regression model with heteroskedasticity and autocorrelation consistent standard errors. The default method is "FQSB".

`type` one of "max" or "all"; "max" includes only the region with maximum t-value. The default option is "max"; "all" includes all units that pass the test t in the core formation (step 2).

Details

According to the second step of the Phillips and Sul clustering algorithm (2007, 2009), the *log t* regression should be run for the first k units $2 < k < N$ maximizing k under the condition that $t - value > -1.65$. In other words, the core group size k^* is chosen as follows:

$$k^* = argmax_k\{t_k\}$$

subject to

$$\min t_k > -1.65$$

Such behavior is obtained with `type="max"`; if `type="all"`, all units that satisfy $t_k > -1.65$ are added to core group.

If the condition $t_k > -1.65$ does not hold for $k = 2$ (the first two units), the algorithm drops the first unit and repeats the same procedure for the next pair of units. If $t_k > -1.65$ does not hold for any couple of units, the whole panel diverges.

Value

A numeric vector containing the row indices of the units included in the core group; if a core group cannot be found, returns `FALSE`.

References

Phillips, P. C.; Sul, D., 2007. Transition modeling and econometric convergence tests. Econometrica 75 (6), 1771-1855.

Phillips, P. C.; Sul, D., 2009. Economic transition and growth. Journal of Applied Econometrics 24 (7), 1153-1185.

`dim.convergence.clubs` `dim` *method for S3 object* `convergence.clubs`

Description

`dim` method for S3 object `convergence.clubs`

Usage

```
## S3 method for class 'convergence.clubs'
dim(x, ...)
```

Arguments

`x`	an object of class `convergence.clubs`.
`...`	other parameters to pass to function `summary()`.

Value

an integer vector with two values: the first one indicates the number of clubs, the second one indicates the number of divergent units

---

`estimateMod` *Log-t test for convergence*

Description

Estimates the *log-t* regression model proposed by Phillips and Sul (2007, 2009) in order to investigate the presence of convergence by adopting the Andrews estimator of long-run variance (fixed or adaptive bandwidth of the kernel).

Usage

```
estimateMod(H, time_trim = 1/3, HACmethod = c("FQSB", "AQSB"))
```

Arguments

`H`	vector of H values
`time_trim`	a numeric value between 0 and 1, representing the portion of time periods to trim when running *log t* regression model. Phillips and Sul (2007, 2009) suggest to discard the first third of the period.
`HACmethod`	string indicating whether a Fixed Quadratic Spectral Bandwidth (`HACmethod="FQSB"`) or an Adaptive Quadratic Spectral Bandwidth (`HACmethod="AQSB"`) should be used for the truncation of the Quadratic Spectral kernel in estimating the *log t* regression model with heteroskedasticity and autocorrelation consistent standard errors. The default method is "FQSB".

Details

The following linear model is estimated:

$$\log \frac{H_1}{H_t} - 2\log(\log t) = \alpha + \beta \log t + u_t$$

Heteroskedasticity and autocorrelation consistent (HAC) standard errors are used with Quadratic Spectral kernel (Andrews, 1991), If `HACmethod="FQSB"`, a fixed bandwidth parameter is applied, while with `HACmethod="AQSB"` an adaptive bandwidth parameter is employed.

Value

A named vector containing information about the model used to run the t-test on the units in the club: beta coefficient, standard deviation, t-statistics and p-value.

References

Andrews, D. W., 1991. Heteroskedasticity and autocorrelation consistent covariance matrix estimation. Econometrica: Journal of the Econometric Society, 817-858.

Phillips, P. C.; Sul, D., 2007. Transition modeling and econometric convergence tests. Econometrica 75 (6), 1771-1855.

Phillips, P. C.; Sul, D., 2009. Economic transition and growth. Journal of Applied Econometrics 24 (7), 1153-1185.

`filteredGDP` *Filtered per-capita GDP of 152 Countries from 1970 to 2003*

Description

A dataset containing the per-capita GDP of 152 Countries over 34 years (Phillips and Sul, 2009). Data were filtered in order to remove business cycles.

Usage

```
data(filteredGDP)
```

Format

A data frame with 152 rows and 35 variables.

Details

ID Country names (character);

Y1970, ..., Y2003 per-capita GDP from year 1970 to 2003 (filtered in order to remove business cycles).

References

Phillips, P. C.; Sul, D., 2009. Economic transition and growth. Journal of Applied Econometrics 24 (7), 1153-1185.

`findClubs` *Finds convergence clubs*

Description

Find convergence clubs by means of Phillips and Sul clustering procedure.

Usage

```
findClubs(
  X,
  dataCols,
  unit_names = NULL,
  refCol,
  time_trim = 1/3,
  HACmethod = c("FQSB", "AQSB"),
  cstar = 0,
  cstar_method = c("fixed", "incremental"),
  cstar_increment = 0.1,
  cstar_cap = 3
)
```

Arguments

`X`	dataframe containing data (preferably filtered data in order to remove business cycles). Data must not contain any NA or NaN values, otherwise the clustering procedure will be stopped with an error. In order for the clustering procedure to work, data must be arranged such that each row represents a unit and each column represents a time period (year, month, ...)
`dataCols`	integer vector with the column indices of the data
`unit_names`	integer scalar indicating, if present, the index of a column with codes of the units
`refCol`	integer scalar indicating the index of the column to use for ordering data
`time_trim`	a numeric value between 0 and 1, representing the portion of time periods to trim when running *log-t* regression model. Phillips and Sul (2007, 2009) suggest to discard the first third of the period.
`HACmethod`	string indicating whether a Fixed Quadratic Spectral Bandwidth (`HACmethod="FQSB"`) or an Adaptive Quadratic Spectral Bandwidth (`HACmethod="AQSB"`) should be used for the truncation of the Quadratic Spectral kernel in estimating the *log-t* regression model with heteroskedasticity and autocorrelation consistent standard errors. The default method is "FQSB".
`cstar`	numeric scalar, indicating the threshold value of the sieve criterion (c^*) to include units in the detected core (primary) group (step 3 of Phillips and Sul (2007, 2009) clustering algorithm). The default value is 0.

`cstar_method` a string specifying whether cstar should be mantained fixed (`cstar_method="fixed"`) or increased iteratively until the whole club satisfies the condition $tvalue > -1.65$ (`cstar_method="incremental"`). Default is `cstar_method="fixed"` (see Details).

`cstar_increment` a positive value specifying the increment to cstar, only valid if `cstar_method="incremental"` (see Details); the default value is 0.1.

`cstar_cap` scalar indicating the maximum value up to which `cstar` can be increased; the default value is 3.

Details

In order to investigate the presence of convergence clubs according to the Phillips and Sul clustering procedure, the following steps are implemented:

1. (Cross section last observation ordering): Sort units in descending order according to the last panel observation of the period;
2. (Core group formation): Run the *log-t* regression for the first k units $(2 < k < N)$ maximizing k under the condition that t-value is > -1.65. In other words, chose the core group size k* as follows:

   $$k^* = argmax_k\{t_k\}$$

   subject to

   $$min\{t_k\} > -1.65$$

   If the condition $t_k > -1.65$ does not hold for $k = 2$ (the first two units), drop the first unit and repeat the same procedure. If $t_k > -1.65$ does not hold for any units chosen, the whole panel diverges;
3. (Sieve the data for club membership): After the core group is detected, run the *log-t* regression for the core group adding (one by one) each unit that does not belong to the latter. If t_k is greater than a critical value c^* add the new unit in the convergence club. All these units (those included in the core group k^* plus those added) form the first convergence club. Note that Phillips and Sul (2007) suggest to make sure $t_k > -1.65$ for the subconvergence group obtained. Otherwise, repeat the procedure by increasing the value of the c^* parameter until the condition $t_k > -1.65$ is satisfied for the subconvergence group;
4. (Recursion and stopping rule): If there are units for which the previous condition fails ($t_k < c^*$), gather all these units in one group and run the *log-t* test to see if the condition $t_k > -1.65$ holds. If the condition is satisfied, conclude that there are two convergence clubs. Otherwise, step 1 to 3 should be repeated on the same group to determine whether there are other subgroups that constitute convergence clubs. If no k in step 2 satisfies the condition $t_k > -1.65$, the remaining units diverge.

Note that the clustering procedure may return groups with $t_k < -1.65$, which are not really convergence clubs. In this case, following step 3 of the clustering procedure there are two options: (i) allow an iterative increase of the cstar parameter until the subconvergence club satisfies the condition $t_k > -1.65$. In this case it should the argument `cstar_method` should be set to `"incremental"` and a positive argument for the `cstar_increment` argument should be chosen; (ii) increase the value of the cstar in order to increase the discriminatory power of the *log-t* test in the formation of each club.

Information about clubs, divergent units and the c^* used for each club can be easily displayed by means of the summary() function, for which the package provides a specific method for the convergence.clubs class.

Value

Ad object of class convergence.clubs, containing a list of Convergence Clubs, for each club a list is return with the following objects: id, a vector containing the row indices of the units in the club; model, a list containing information about the model used to run the t-test on the units in the club; unit_names, a vector containing the names of the units of the club (optional, only included if parameter unit_names is given)

References

Andrews, D. W., 1991. Heteroskedasticity and autocorrelation consistent covariance matrix estimation. Econometrica: Journal of the Econometric Society, 817-858.

Phillips, P. C.; Sul, D., 2007. Transition modeling and econometric convergence tests. Econometrica 75 (6), 1771-1855.

Phillips, P. C.; Sul, D., 2009. Economic transition and growth. Journal of Applied Econometrics 24 (7), 1153-1185.

See Also

mergeClubs, Merges a list of clubs created by findClubs;

mergeDivergent, Merges divergent units according to the algorithm proposed by von Lyncker and Thoennessen (2017)

Examples

```
data("filteredGDP")


# Cluster Countries using GDP from year 1970 to year 2003
clubs <- findClubs(filteredGDP,  dataCols=2:35, unit_names = 1, refCol=35,
                   time_trim = 1/3, HACmethod = "FQSB",
                   cstar = 0,
                   cstar_method = 'incremental',
                   cstar_increment = 0.1)



## Not run:
# Cluster Countries using GDP from year 1970 to year 2003
clubs <- findClubs(filteredGDP,  dataCols=2:35, unit_names = 1, refCol=35,
                   time_trim = 1/3, HACmethod = "AQSB", cstar = 0)


## End(Not run)
```

GDP *Per-capita GDP of 152 Countries from 1970 to 2003*

Description

A dataset containing the per-capita GDP of 152 Countries over 34 years (Phillips and Sul, 2009).

Usage

```
data(GDP)
```

Format

A data frame with 152 rows and 35 variables.

Details

ID Country names (character);

Y1970, ..., Y2003 per-capita GDP from year 1970 to 2003.

References

Phillips, P. C.; Sul, D., 2009. Economic transition and growth. Journal of Applied Econometrics 24 (7), 1153-1185.

mergeClubs *Merge convergence clubs*

Description

Merges a list of clubs created with the function findClubs by either Phillips and Sul method or von Lyncker and Thoennessen procedure.

Usage

```
mergeClubs(
  clubs,
  time_trim,
  mergeMethod = c("PS", "vLT"),
  threshold = -1.65,
  mergeDivergent = FALSE,
  estar = -1.65
)
```

Arguments

`clubs`	an object of class `convergence.clubs` (created by findClubs function)
`time_trim`	a numeric value between 0 and 1, representing the portion of time periods to trim when running log t regression model; if omitted, the same value used for `clubs` is used.
`mergeMethod`	character string indicating the merging method to use. Methods available are `"PS"` for Phillips and Sul (2009) and `"vLT"` for von Lyncker and Thoennessen (2017).
`threshold`	a numeric value indicating the threshold to be used with the t-test.
`mergeDivergent`	logical, if TRUE, indicates that merging of divergent units should be tried.
`estar`	a numeric value indicating the threshold e^* to test if divergent units may be included in one of the new convergence clubs. To be used only if `mergeDivergent=TRUE`

Details

Phillips and Sul (2009) suggest a "club merging algorithm" to avoid over determination due to the selection of the parameter c^*. This algorithm suggests to merge for adjacent groups. In particular, it works as follows:

1. Take the first two groups detected in the basic clustering mechanism and run the log-t test. If the t-statistic is larger than -1.65, these groups together form a new convergence club;
2. Repeat the test adding the next group and continue until the basic condition (t-statistic > -1.65) holds;
3. If convergence hypothesis is rejected, conclude that all previous groups converge, except the last one. Hence, start again the test merging algorithm beginning from the group for which the hypothesis of convergence did not hold.

On the other hand, von Lyncker and Thoennessen (2017), propose a modified version of the club merging algorithm that works as follows:

1. Take all the groups detected in the basic clustering mechanism (P) and run the t-test for adjacent groups, obtaining a (M × 1) vector of convergence test statistics t (where $M = P - 1$ and $m = 1, \dots, M$);
2. Merge for adjacent groups starting from the first, under the conditions $t(m) > -1.65$ and $t(m) > t(m+1)$. In particular, if both conditions hold, the two clubs determining $t(m)$ are merged and the algorithm starts again from step 1, otherwise it continues for all following pairs;
3. For the last element of vector M (the value of the last two clubs) the only condition required for merging is $t(m = M) > -1.65$.

Value

Ad object of class `convergence.clubs`, containing a list of Convergence Clubs, for each club a list is return with the following objects: `id`, a vector containing the row indices of the units in the club; `model`, a list containing information about the model used to run the t-test on the units in the club; `unit_names`, a vector containing the names of the units of the club (optional, only included if parameter `unit_names` is given)

References

Phillips, P. C.; Sul, D., 2007. Transition modeling and econometric convergence tests. Econometrica 75 (6), 1771-1855.

Phillips, P. C.; Sul, D., 2009. Economic transition and growth. Journal of Applied Econometrics 24 (7), 1153-1185.

von Lyncker, K.; Thoennessen, R., 2017. Regional club convergence in the EU: evidence from a panel data analysis. Empirical Economics 52 (2), 525-553

See Also

`findClubs`, finds convergence clubs by means of Phillips and Sul clustering procedure.

`mergeDivergent`, merges divergent units according to the algorithm proposed by von Lyncker and Thoennessen (2017).

Examples

```
data("filteredGDP")

# Cluster Countries using GDP from year 1970 to year 2003
clubs <- findClubs(filteredGDP, dataCols=2:35, unit_names = 1, refCol=35,
                   time_trim = 1/3, cstar = 0, HACmethod = "FQSB")
summary(clubs)

# Merge clusters
mclubs <- mergeClubs(clubs, mergeMethod='PS', mergeDivergent=FALSE)
summary(mclubs)

mclubs <- mergeClubs(clubs, mergeMethod='vLT', mergeDivergent=FALSE)
summary(mclubs)
```

`mergeDivergent` *Merge divergent units*

Description

Merges divergent units according the algorithm proposed by von Lyncker and Thoennessen (2017)

Usage

```
mergeDivergent(clubs, time_trim, estar = -1.65)
```

Arguments

- `clubs` an object of class `convergence.clubs` (created by `findClub` or `mergeClubs` function)
- `time_trim` a numeric value between 0 and 1, representing the portion of time periods to trim when running log t regression model; if omitted, the same value used for `clubs` is used.
- `estar` a numeric value indicating the threshold e^* to test if divergent units may be included in one of the new convergence clubs. To be used only if `mergeDivergent=TRUE`.

Details

von Lyncker and Thoennessen (2017) claim that units identified as divergent by the basic clustering procedure by Phillips and Sul might not necessarily still diverge in the case of new convergence clubs detected with the club merging algorithm. To test if divergent units may be included in one of the new convergence clubs, they propose the following algorithm:

1. Run a log t-test for all diverging units, and if $t_k > -1.65$ all these units form a convergence club (This step is implicitly included in Phillips and Sul basic algorithm);
2. Run a log t-test for each diverging units and each club, creating a matrix of t-values with dimensions $d \times p$, where each row d represents a divergent region and each column p a convergence club;
3. Take the highest $t > e^*$ and add the respective region to the respective club and restart from the step 1. the authors suggest to use $e^* = t = -1.65$;
4. The algorithm stops when no t-value > e* is found in step 3, and as a consequence all remaining units are considered divergent.

Value

A list of Convergence Clubs, for each club a list is return with the following objects: `id`, a vector containing the row indices of the units in the club; `model`, a list containing information about the model used to run the t-test on the units in the club; `unit_names`, a vector containing the names of the units of the club (optional, only included if it is present in the `clubs` object given in input).

References

Phillips, P. C.; Sul, D., 2007. Transition modeling and econometric convergence tests. Econometrica 75 (6), 1771-1855.

Phillips, P. C.; Sul, D., 2009. Economic transition and growth. Journal of Applied Econometrics 24 (7), 1153-1185.

von Lyncker, K.; Thoennessen, R., 2017. Regional club convergence in the EU: evidence from a panel data analysis. Empirical Economics 52 (2), 525-553

See Also

`mergeClubs`, Merges a list of clubs created by `findClubs`;

Examples

```
data("filteredGDP")

#Cluster Countries using GDP from year 1970 to year 2003
clubs <- findClubs(filteredGDP, dataCols=2:35, unit_names = 1, refCol=35,
                   time_trim = 1/3, cstar = 0, HACmethod = "FQSB")
summary(clubs)

# Merge clusters and divergent units
mclubs <- mergeClubs(clubs, mergeDivergent=TRUE)
summary(mclubs)
```

`plot.convergence.clubs` *Plot method for S3 class* `convergence.clubs`

Description

Plot the transition paths of units in the convergence clubs and the average transition paths of those clubs.

Usage

```
## S3 method for class 'convergence.clubs'
plot(
  x,
  y = NULL,
  nrows = NULL,
  ncols = NULL,
  clubs,
  avgTP = TRUE,
  avgTP_clubs,
  y_fixed = FALSE,
  legend = FALSE,
  save = FALSE,
  filename,
  path,
  width = 20,
  height = 15,
  device = c("pdf", "png", "jpeg"),
  res,
  plot_args,
  legend_args,
  breaks,
  ...
)
```

Arguments

x	an object of class convergence.clubs.
y	unused, added for compatibility with function plot
nrows	number of rows of the graphical layout, if NULL, it is automatically defined
ncols	number of columns of the graphical layout, if NULL, it is automatically defined
clubs	numeric scalar or vector, indicating for which clubs the transition path plot should be generated. Optional, if omitted, plots for all clubs are produced. If clubs=NULL, transition path are not plotted for any club.
avgTP	logical, indicates if a plot with the average transition paths of each convergence club should be produced. Default is TRUE.
avgTP_clubs	numeric scalar or vector, indicating for which clubs the average transition path should be displayed. Optional, if omitted, average transition paths for all clubs are plotted.
y_fixed	logical, should the scale of the y axis be the same for all plots? Default is FALSE.
legend	logical, should a legend be displayed? Default is FALSE.
save	logical, should the plot be saved as a file?
filename	optional, a string indicating the name of the file where the plot should be saved; must include the extension (e.g. "plot.pdf")
path	optional, a string representing the path of the directory where the plot should be saved; the path should not end with a slash symbol ("/")
width	the image width when saving the plot, in inches.
height	the image height when saving the plot, in inches.
device	string indicating the format to be used to save the plot; one of "pdf", "png" or "jpeg". The default is "pdf".
res	the resolution of the image, in ppi; only used with device="png" and device="jpeg"
plot_args	optional, a named list with the graphical parameters for the plot, see Details section.
legend_args	optional, a named list with the graphical parameters for the legend, see Details section.
breaks	a vector of integer values representing the columns (time periods) to be plotted. Accepted values are integers from 1 to T, that is the number of time periods included in the convergence procedure. Optional, if omitted, all periods are plotted.
...	other parameters to pass to function plot().

Details

nrows and ncols are optional parameters used to define the row and column number for the plot layout. Both or just one of them may be specified. If none of them is specified, the layout dimension is chosen automatically.

If legend=TRUE and a column with units' names is available in the x object, those names are truncated to fit the plot's legend. The graphical parameter cex may be used to modify the size of the legend's labels, default is 0.8

Note that, when using RStudio, one may incur in an error if the plot window is too small. Enlarging the plot window usually solves the problem.

List of argument that could be included in `plot_args` as a list:

- `lty` numeric scalar or vector indicating the line type (values available range from 1 to 6)
- `type` a string indicating whether the points (markers) should be displayed. If 'l' no markers are displayed; if 'o' markers are displayed;
- `pch` numeric scalar or vector to specify symbols to use when plotting points (markers). If omitted, customized markers are used for each line. If fixed (e.g. pch=1) the same marker is used for each line. (Values available range from 0 to 25)
- `cex` number indicating the amount by which plotting text and symbols should be scaled relative to the default. 1=default, 1.5 is 50% larger, 0.5 is 50% smaller, and so on. Default is 1.
- `lwd` number indicating the line width. Default is 1.
- `xlab` string indicating x-axis label. If omitted, 'Time', is displayed
- `ylab` string indicating y-axis label. If omitted, 'Relative transition path', is displayed
- `cex.lab` number indicating the amount by which plotting x and y labels should be scaled relative to cex. Default is 1.
- `col` option to specify colors for each line. Colors could be specified by index, name, hexadecimal, or RGB. For example col=1, col="white", and col="#FFFFFF" are equivalent. If omitted, colors are chosen randomly.
- `col_hline` color of the horizontal line for h=1. Default is 'black'.
- `xmarks` vector with tic marks to be displayed in the x axis.
- `xlabs` vector with labels of marks to be displayed in the x axis.
- `xlabs_dir` number indicating the direction of x-axis labels. For horizontal labels xlabs_dir=0; for vertical labels xlabs_dir=2.

List of argument that could be included in `legend_args` as a list:

- `cex` number indicating the amount by which plotting text and symbols should be scaled relative to the default. Default is 0.9
- `lwd` Number indicating the line width. Default is 1.
- `y.intersp` number indicating the space between each legend entry. Default is 1.
- `max_length_labels` maximum length of the labels displayed for each legend entry.

Note that, when using *RStudio*, one may incur in an error if the plot window is too small. Enlarging the plot window usually solves the problem. We suggest to export plots in the available formats ("pdf", "png" or "jpeg") using adequate values of width and height.

Examples

```
data("filteredGDP")

clubs <- findClubs(filteredGDP, dataCols=2:35, unit_names = 1, refCol=35, time_trim = 1/3,
                   cstar = 0, HACmethod = "FQSB")
```

```
### Plot transition paths for all clubs
plot(clubs)
plot(clubs, y_fixed=TRUE)
plot(clubs, nrows=2,ncols=4)

plot(clubs, ncols=3, lty='dotdash', lwd=3, col="blue")
plot(clubs, ncols=3, y_fixed=TRUE, lty='dotdash', lwd=3, col="blue")

### Plot transition paths only for some clubs
plot(clubs, clubs=c(2,4,5))
plot(clubs, nrows=1, ncols=3, clubs=c(2,4,5), avgTP = FALSE)
plot(clubs, nrows=1, ncols=3, clubs=c(2,4,5), avgTP = FALSE, legend=TRUE)
plot(clubs, clubs=c(2,4,5), avgTP_clubs = c(1,3))
plot(clubs, clubs=c(2,4,5), avgTP_clubs = c(1,3), legend=TRUE)


### Export customized plots
#Only plot average transition paths
plot(clubs, clubs=NULL, avgTP = TRUE, legend=TRUE)

#only lines, without markers and legend
plot(clubs, save = TRUE, filename ="name.pdf" , path = tempdir(), width = 15, height = 10)

#markers and legend (up to the fourth characther is shown)
plot(clubs, legend=TRUE, plot_args=list(type='o'),
    legend_args=list(max_length_labels=4, y.intersp=1),
    save = TRUE, filename ="name.pdf", path = tempdir(), width = 15, height = 10)

#for large samples the legend could be better displayed by plotting each club
#in turn and by increasing the plot dimension (through width and height)
plot(clubs, clubs=1, avgTP=FALSE, legend=TRUE, plot_args=list(type='o'),
    legend_args=list(max_length_labels=8, y.intersp=1),
    save = TRUE, filename ="name.pdf", path = tempdir(), width = 20, height = 15)

#customize x-labels - 1
plot(clubs, legend=TRUE, plot_args=list(type='o', xmarks=seq(1,34),xlabs=seq(1970,2003),
    xlabs_dir=0), legend_args=list(max_length_labels=4, y.intersp=1),
    save = TRUE, filename ="name.pdf" , path = tempdir(), width = 15, height = 10)

#customize x-labels - 2
plot(clubs, legend=TRUE, plot_args=list(type='o', xmarks=seq(1,34,1), xlabs=seq(1970,2003,1),
    xlabs_dir=2), legend_args=list(max_length_labels=4, y.intersp=1),
    save = TRUE, filename ="name.pdf" , path = tempdir(), width = 15, height = 10)

#show only the plot with the average transition paths of each club
plot(clubs, clubs=NULL, avgTP=TRUE, legend=TRUE,
    plot_args=list(type='o', xmarks=seq(1,34), xlabs=seq(1970,2003), xlabs_dir=0),
    save = TRUE, filename ="name.pdf" , path = tempdir(), width = 15, height = 10)

#markers and legend - png format
```

```
plot(clubs, legend=TRUE, plot_args=list(type='o'),
    legend_args=list(max_length_labels=4, y.intersp=1),
    save = TRUE, filename ="name.png" , path = tempdir(), width = 15, height = 10,
    device= "png", res=100)
```

`print.convergence.clubs`
Print method for S3 object `convergence.clubs`

Description

Print method for S3 object `convergence.clubs`

Usage

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

Arguments

`x`	an object of class `convergence.clubs`.
`...`	other parameters to pass to function `summary()`.

`summary.convergence.clubs`
Summary method for S3 object `convergence.clubs`

Description

Summary method for S3 object `convergence.clubs`

Usage

```
## S3 method for class 'convergence.clubs'
summary(object, ...)
```

Arguments

`object`	an object of class `convergence.clubs`.
`...`	other parameters to pass to function `summary()`.

transition_paths — *Extract transition paths from a* `convergence.clubs` *object*

Description

Given a `convergence.clubs` object (created by either `findClubs` or `mergeClubs` function), returns a list with transition paths for each club.

Usage

```
transition_paths(
  clubs,
  include_unit_names = TRUE,
  output_type = c("list", "data.frame")
)
```

Arguments

`clubs`	an object of class `convergence.clubs` (created by either function `findClubs` or `mergeClubs`).
`include_unit_names`	logical, if TRUE (the default) adds a column with unit names (only if present in the `convergence.clubs` object passed to `clubs`).
`output_type`	string indicating if the function should output a list or a data frame. Possible options are "list" and "data.frame", default is "list".

Value

If `output_type=="list"`, a list of data frames, one for each club; each data frame will contain transition paths for the units in the correspondent club. If `output_type=="data.frame"`, a data.frame.

References

Phillips, P. C.; Sul, D., 2007. Transition modeling and econometric convergence tests. Econometrica 75 (6), 1771-1855.

Phillips, P. C.; Sul, D., 2009. Economic transition and growth. Journal of Applied Econometrics 24 (7), 1153-1185.

See Also

`findClubs`, Finds Convergence Clubs; `mergeClubs`, Merges a list of clubs created by `findClubs`; `plot.convergence.clubs`, Plots transition paths from a `convergence.clubs` object.

Examples

```
data("filteredGDP")

# Cluster Countries using GDP from year 1970 to year 2003
clubs <- findClubs(filteredGDP, dataCols=2:35, unit_names = 1, refCol=35,
                   time_trim = 1/3, cstar = 0, HACmethod = "FQSB")

# Extract Transition Paths
tp <- transition_paths(clubs)
tp <- transition_paths(clubs, output_type = 'data.frame')
```

Index