Package ‘convergEU’

January 21, 2024

**Type** Package  
**Title** Monitoring Convergence of EU Countries  
**Version** 0.7.3.2  
**Depends** R (>= 4.0.0)  
**Maintainer** Eleonora Peruffo <eleonora.peruffo@eurofound.europa.eu>

**URL**  
https://www.eurofound.europa.eu/system/files/2022-04/introduction-to-the-convergeu-package-0.6.4-tutorial-v2-apr2022.pdf,  
https://www.ajs.or.at/index.php/ajs/article/view/1468

**Description** Indicators and measures by country and time describe what happens at economic and social levels. This package provides functions to calculate several measures of convergence after imputing missing values. The automated downloading of Eurostat data, followed by the production of country fiches and indicator fiches, makes possible to produce automated reports. The Eurofound report (<doi:10.2806/68012>) “Upward convergence in the EU: Concepts, measurements and indicators”, 2018, is a detailed presentation of convergence.

**Imports** dplyr, tibble, ggplot2, eurostat, tidyr, rlang, utils, purrr, caTools, broom, stringr, rmarkdown, ggpubr, leaflet, tidyselect

**Suggests** devtools, formattable, gridExtra, knitr, kableExtra, magrittr, readr, readxl, tidyverse, rvest, testthat, utf8

**License** GPL-3 | BSD_3_clause + file LICENSE

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.0

**VignetteBuilder** knitr
R topics documented:

abso_change ............................................. 3
average_clust ............................................. 4
beta_conv .................................................. 6
beta_conv_graph ........................................... 8
check_country ............................................. 9
check_data ............................................... 10
coeu_grad ................................................ 11
coeu_gradV ................................................ 12
compo_cond_EUS .......................................... 12
convergEU_glb ........................................... 13
country_ranking .......................................... 14
dbEUF2018meta ............................................ 15
dbEurofound ............................................... 16
dbMetaEUStat .............................................. 18
delta_conv ................................................ 19
demea_change ............................................. 20
departure_best ........................................... 23
departure_best_plot ..................................... 24
departure_mean .......................................... 26
dev_mean_plot ............................................. 27
download_indicator_EUS ................................. 29
down_lo_EUS .............................................. 30
dow_soc_scor_boa ........................................ 32
dow_soc_scor_boa ........................................ 32
emp_20_64_MS ............................................. 32
extract_indicator_ECG .................................... 33
extract_indicator_EIGE ................................... 34
extract_indicator_EUF .................................... 35
gamma_conv ............................................. 37
gamma_conv_msteps .................................... 38
go_indica_fi .............................................. 39
go_ms_fi ................................................... 41
graph_departure ......................................... 42
Absolute change

Description

Given a dataframe of quantitative indicators along time, the absolute change is calculated. A time variable must be present and sorted. Missing values are not allowed. All other columns are indicator values in each considered country.

Usage

abso_change(tavDes, time_0, time_t, all_within = TRUE, timeName = "time")

Arguments

tavDes the sorted dataframe time by countries. No other variables besides time and countries’ indicator must be present.
time_0 reference time
time_t focus time strictly larger than time_0
all_within is TRUE is several times are considered within the specified interval (default), otherwise FALSE; the reference time remains time_0.
timeName the name of the variable that contains time information
Value

a list of absolute changes for each country, the sum of absolute values and the average per pairs of years.

References


Examples

# Example 1
# Sorted dataframe in the format years by countries:
require(tibble)
testTB <- dplyr::tribble(~years, ~countryA , ~countryB, ~countryC,
2000, 0.8, 2.7, 3.9,
2001, 1.2, 3.2, 4.2,
2002, 0.9, 2.9, 4.1,
2003, 1.3, 2.9, 4.0,
2004, 1.2, 3.1, 4.1,
2005, 1.2, 3.0, 4.0)

# Absolute change for each country with time_0=2000 and time_t=2005:
mySTB< abso_change(tavDes=testTB,time_0=2000, time_t=2005, timeName ="years")

# The component "res" is a list of absolute changes for each country,
# the sum of absolute values and the average per pairs of years:
names(mySTB$res)

# Absolute change for each country with time_0=2002 and time_t=2005:
mySTB1< abso_change(tavDes=testTB,time_0=2002, time_t=2005, timeName="years")

# If all_within is FALSE, only times 2002 and 2005 are considered:
mySTB2< abso_change(tavDes=testTB,time_0=2002, time_t=2005, all_within =FALSE, timeName="years")

# Example 2
# Absolute changes of Member States for the emp_20_64_MS Eurofound dataset:
data(emp_20_64_MS)
mySTB3 <- abso_change(emp_20_64_MS,time_0 = 2005,time_t = 2010,timeName = "time")
mySTB4 <- abso_change(emp_20_64_MS,time_0 = 2007,time_t = 2012,timeName = "time")
average_clust

Description

The computation is based on clusters defined in objects created by invoking *convergEU_glb()*.
At now only cluster labels contained into *convergEU_glb()* are possible.

Usage

average_clust(myTB, timeName = "time", cluster = "EU27")

Arguments

myTB time by member states dataset.
timeName name of the variable that contains time.
cluster the label defining a cluster; one string selected within the following: "EU12", "EU15", "EU19", "EU25", "EU27_2007", "EU28", "EU27_2020", "Eurozone", "EA", "all" (for all countries in the dataset).

Details

The cluster specification is based on labels: "EU27_2020", "EU27_2007", "EU25", "EU19", "EU15", "EU12", "EA", "Eurozone", "all". The option cluster = "all" indicates that all countries in the dataset have to be considered.

Value

The dataset with the average of clustered countries.

References


Examples

# Example 1
# Unweighted average of Member States for cluster "EU12":
myAC1<average_clust(emp_20_64_MS,timeName = "time",cluster = "EU12")

# Visualize results for Italy:
myAC1$res[,c(1,17)]

# Visualize results for the first five member states:
myAC1$res[,c(1:6)]

# Example 2
# Unweighted average of Member States for cluster "EU25":
myAC2<average_clust(emp_20_64_MS,timeName = "time",cluster = "EU25")

# Visualize results for France:
myAC2$res[,c(1,13)]
# Visualize results for the first six member states:
myAC2$res[,c(1:7)]

# Example 3
# Unweighted average of countries for cluster "EU27":
myAC<-average_clust(emp_20_64_MS,timeName = "time",cluster = "EU27")

# Visualize results for Germany:
myAC$res[,c(1,7)]

# Visualize results for the first five member states:
myAC$res[,c(1:6)]

---

**beta_conv**  
*Beta-convergence statistic*

**Description**

Given a dataframe of quantitative indicators along time, the unconditional beta convergence is a statistic capturing some important features. A time variable must be present and sorted. Missing values are not allowed. All other columns are indicator values in each considered country.

**Usage**

```r
def beta_conv(  
tavDes,  
time_0,  
time_t,  
all_within = FALSE,  
timeName = "time",  
useTau = TRUE,  
useCon = FALSE  
)
```

**Arguments**

- **tavDes**: the sorted dataframe time by countries on the original scale. No other variable besides time and countries' indicator must be present.
- **time_0**: reference time.
- **time_t**: target time strictly larger than time_0.
- **all_within**: is FALSE if just two different years are considered (default); if more than two years are desired within the specified interval then it must be TRUE ; the reference time remains time_0.
- **timeName**: the name of the variable that contains time information.
- **useTau**: if TRUE the log ratio of indicator values is divided for the elapsed time (years).
- **useCon**: if TRUE replaces 0 with a minimum constant value.
Value

a list with the value of beta-conv, by OLS (least-squares), the transformed data and standard statistical tests.

References


Examples

# Example 1:
# Dataframe in the format years by countries:
require(tibble)
myTB1 <- tibble::tribble(
  ~years, ~UK, ~DE, ~IT,
  1990,  998, 1250, 332,
  1988, 1201,  868, 578,
  1989, 1150,  978, 682,
  1991, 1600, 1350, 802
)
# Sort the time variable:
newdata <- myTB1[order(myTB1$years),]
# Beta convergence statistic by considering just two times, e.g. 1989 and 1991:
myBC1 <- beta_conv(newdata,1989,1991,timeName="years")
# Visualize the summary of the results (estimated coefficients, standard errors, p-values):
myBC1$res$summary
# Visualize the adjusted R-squared:
myBC1$res$adj.r.squared
# Beta convergence statistic by considering more than two times:
myBC2 <- beta_conv(newdata,1988,1991,all_within=TRUE,timeName="years")

# Example 2:
# Dataframe in the format years by countries, time variable already sorted:
testTB <- tribble(
  ~time, ~countryA, ~countryB, ~countryC,
  2000,  0.8,  2.7,  3.9,
  2001,  1.2,  3.2,  4.2,
  2002,  0.9,  2.9,  4.1,
  2003,  1.3,  2.9,  4.0,
  2004,  1.2,  3.1,  4.1,
  2005,  1.2,  3.0,  4.0
)
myBC3 <- beta_conv(testTB, time_0 = 2000, time_t = 2005, timeName = "time")
myBC4 <- beta_conv(testTB, time_0 = 2000, time_t = 2005, all_within = TRUE, timeName = "time")
# Example 3
# Beta convergence for the emp_20_64_MS Eurofound dataset:
data(emp_20_64_MS)
empBC <- beta_conv(emp_20_64_MS, time_0 = 2002, time_t = 2006, timeName = "time")

# Summary of the model results:
empBC$res$summary

# Adjusted R-squared:
empBC$res$adj.r.squared

beta_conv_graph

**Graphical representation based on beta convergence**

**Description**

A ggplot of transformed data and a straight line for the results obtained for beta-convergence

**Usage**

```
beta_conv_graph(betaRes, indiName = NA, time_0 = NA, time_t = NA)
```

**Arguments**

- `betaRes`: the output obtained from beta_conv function.
- `indiName`: name of the considered indicator as a string.
- `time_0`: starting time.
- `time_t`: ending time.

**Value**

A ggplot object to be displayed or saved using ggsave.

**References**


**Examples**

```
# Example 1
# Beta convergence for the emp_20_64_MS Eurofound dataset in the period 2002-2006:
data(emp_20_64_MS)
empBC <- beta_conv(emp_20_64_MS, time_0 = 2002, time_t = 2006, timeName = "time")
```
Check a dataset (tibble) for the presence of countries

Description
A given list of countries is contained into a dataset (tibble). If not, an object signaling this error is returned.

Usage
check_country(myTB, clusterCode = "EU27")

Arguments
myTB dataset (tibble) to be checked
clusterCode string to denote which countries should be in the dataset

Value
TRUE if they are inside, FALSE otherwise

References

Examples
# Check the dataset "emp_20_64_MS" for the presence of countries in cluster EU27:
check_country(emp_20_64_MS, clusterCode="EU27")

# Check absence for EU27:
check_country(emp_20_64_MS[,,-(6:8)], clusterCode="EU27")
# Check the dataset "emp_20_64_MS" for the presence of countries in cluster EU25:
check_country(emp_20_64_MS, clusterCode="EU25")

# Check the dataset "emp_20_64_MS" for the presence of countries in cluster EU12:
check_country(emp_20_64_MS, clusterCode="EU12")

check_data  Make tests on a dataset (dataframe and tibbles)

Description

A dataset can’t have qualitative variables, neither vector of strings nor missing values for computing convergence measures. A time variable should also be present, and if the name is passed then a check on the time order is performed. The object returned states if the dataset is ready for calculations, and if it is not, the error component states why checking failed.

Usage

check_data(tavDes, timeName = NA)

Arguments

tavDes the dataframe under examination
timeName a string with the name of the time variable, optional

Value

an object stating if errors are present

References


Examples

# Example 1
# Tibble dataset with missing values:
require(tibble)
myTB1 <- tibble::tribble(
  ~time, ~veval,
  1988, 1201,
  1989, NA,
  1990, 998,
  1991, NA
)
# Check dataset:
check_data(myTB1)

# Example 2
# Dataset with no missing values, no qualitative variables, and variable time present:
myTB2 <- tibble::tribble(~time, ~veval,
                         1988, 1201,
                         1989, 450,
                         1990, 998,
                         1991, 675)
check_data(myTB2)

# Check the "emp_20_64_MS" Eurofound dataset:
data(emp_20_64_MS)
check_data(emp_20_64_MS, timeName="time")

---

coeu_grad

*Auxiliary function for gradients and delta2*

**Description**

Intermediate calculation to define patterns.

**Usage**

coeu_grad(mEU2, mEU1, mMS2, mMS1, time2, time1)

**Arguments**

- mEU2: average at time 2, EU
- mEU1: average at time 1, EU
- mMS2: average at time 2, Member State
- mMS1: average at time 1, Member State
- time2: time 2
- time1: time 1

**Value**

A list with components time length, grad of member state, grad of EU average and the delta squared difference at a pair of times.
coeu_gradV  

Auxiliary function to provide a different object as input

Description

See function coeu_grad for details.

Usage

coeu_gradV(mEU, mMS, time)

Arguments

- **mEU**: averages at time1 and time 2
- **mMS**: indicator for a member country at time1 and time2
- **time**: the two times considered, sorted in ascending order

Value

a list with components time length, grad of member state, grad of EU average and the delta squared difference at a pair of times.

compo_cond_EUS  

Auxiliary function compo_cond_EUS

Description

Not exported

Usage

compo_cond_EUS(myScarica)

Arguments

- **myScarica**: a bulk downloaded tibble from Eurostat

Value

a tag based on indicator-specific conditioning variables
Description

This is a list of constants and setups for the package. In this function that generates global static objects and tables, cluster of countries are stored with their corresponding labels as well as indicators information and labels.

Usage

convergEU_glb()

Details

Note that EU27 refers to Member States after the 1st February 2020, while EU28 is a valid tag up to 31 March 2020. String EU27_2020 and EU27_2007 as defined by Eurofound are also available.

The following clusters of countries are stored: EU12, EU15, EU25, EU27, EA, Euroarea. Current Member States are elements of EU27_2020. The cluster geoRefEUF is composed of both Member States and other countries (neighboring countries). The component "metaEUstat" contains the indicators’ information, while the component "paralintags" is for defining patterns for the Member States.

Value

a list of constants and objects for package convergEU

References


Examples

# Member States in the cluster Euroarea:
convergEU_glb()$Euroarea

# Cluster EU12 of Member States:
convergEU_glb()$EU12

# Cluster EU27 of Member States after 31 Jan 2020:
convergEU_glb()$EU27

# Cluster EU28 of Member States up to Jan 2020:
convergEU_glb()$EU28

# The countries in the cluster geoRefEUF:
country_ranking

```r
convergEU_glb()$geoRefEUF

# Metainformation on indicators of the European Union:
convergEU_glb()$metaEUStat
```

---

**country_ranking**  
*Ranking of EU countries by time*

**Description**

Countries are ranked for each time according to two types of indicators: higher is the best (highBest) or lower is the best (lowBest).

**Usage**

```r
country_ranking(
  myTB,
  timeName = "time",
  time_0 = NA,
  time_t = NA,
  typeInd = "highBest"
)
```

**Arguments**

- **myTB**  
The dataframe time by countries (complete and sorted by increasing time).
- **timeName**  
The name of the variable that contains time information.
- **time_0**  
Starting time to consider; if NA all times considered.
- **time_t**  
Last time to consider; if NA all times considered.
- **typeInd**  
"highBest" is the default, "lowBest" is the alternative

**Value**

A list with component res which contains ranking by each considered year.

**References**

Examples

# Example 1
# Sorted dataframe in the format years by countries:
require(tibble)
myTB <- tibble::tribble(~years, ~UK, ~DE, ~IT,
  1988, 1201, 868, 578,
  1989, 1150, 978, 682,
  1990, 998, 1250, 332,
  1991, 1600, 1350, 802)

# Country ranking according to the indicator higher is the best:
res <- country_ranking(myTB,timeName="years")

# Country ranking according to the indicator lower is the best:
res1 <- country_ranking(myTB,timeName="years", typeInd="lowBest")

# Country ranking for some years only:
myres <- country_ranking(myTB,timeName="years", time_0=1989,time_t=1990,typeInd="lowBest")

# Example 2
# Ranking of the Member States for the "emp_20_64_MS" dataset
data(emp_20_64_MS)
myCR<-country_ranking(emp_20_64_MS,timeName = "time", time_0 = 2007, time_t = 2010)

# Visualize the results for the first five countries:
myCR$res[1:6]

---

**dbEUF2018meta**  
*Metainformation on Eurofound dataset*

**Description**

Metainformation about data provided by Eurofound currently up to 2018. Metainformation is provided for two dimensions: quality of life and working conditions. For each dimension, metainformation for several indicators is reported, e.g. coding in database, official code, measurement unit, source organization, disaggregation and bookmark URL. Variable names often end with characters denoting scales: The following convention holds for names of variables: "_p" percentage, "_i" index, "_pop" persons, "_h" hours, "_eur" euros, "_pps" purchasing power standards, "_y" years.

**Usage**

data(dbEUF2018meta)
Format
A dataset with 13 rows and 10 columns

Source

References

Examples

```r
data(dbEUF2018meta)
names(dbEUF2018meta)

## Not run:
View(dbEUF2018meta)

## End(Not run)

# Visualize metainformation on the indicators stored in the dataset:
dbEUF2018meta$INDICATOR

# Visualize the indicators coding in database:
dbEUF2018meta$Code_in_database

# Visualize the indicators official code:
dbEUF2018meta$Official_code
```

---

<table>
<thead>
<tr>
<th>dbEurofound</th>
<th>Eurofound dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Description

Source data provided by Eurofound currently up to 2018. Variable names often end with characters denoting scales. The following convention holds for names of variables: 
- "_p" percentage, 
- "_i" index, 
- "_pop" persons, 
- "_h" hours, 
- "_eur" euros, 
- "_pps" purchasing power standards, 
- "_y" years.

Usage

data(dbEurofound)
dbEurofound

Format

A tibble dataset with 17 columns

- **time** time
- **geo** geo
- **geo_label** geo_label
- **sex** gender
- **lifesatisf** Mean_life_satisfaction
- **health** Mean_health_status
- **goodhealth_p** Percentage_of_people_with_good_or_very_good_health
- **trustlocal** Mean_level_of_trust_in_local_government
- **volunt** Level_of_involvement_in_volunteering
- **volunt_p** Percentage_of_people_involved_in_volunteering
- **caring_h** Hours_per_week_spent_in_informal_care
- **socialexc_i** Social_Exclusion_Index
- **JQIskill_i** JQI_Skills_and_discretion_index
- **JQIenviron_i** JQI_Physical_environment_index
- **JQIintensity_i** JQI_Intensity_index
- **JQItime_i** JQI_Working_time_quality_index
- **exposdiscr_p** Exposition_to_discrimination

Details

Further details and metainformation on these data are contained into the dataset *dbEUF2018meta*, say *data(dbEUF2018meta)* in R.

Source


References


Examples

```r
data("dbEurofound")
head(dbEurofound)

# Variable names:
names(dbEurofound)

# time ranges interval:
c(min(dbEurofound$time), max(dbEurofound$time))```
### `dbMetaEUStat`  

#### Description

Metainformation about data from Eurostat processed at Eurofound. More precisely, metainformation is provided for three dimensions: employment, socio economic and quality of life. For each dimension, metainformation for several indicators is reported, e.g. coding in database, official code, measurement unit, source organization, disaggregation and bookmark URL. Variable names often end with characters denoting scales. The following convention holds for names of variables: 

- "_p" percentage, 
- "_i" index, 
- "_pop" persons, 
- "_h" hours, 
- "eur" euros, 
- "pps" purchasing power standards, 
- "y" years.

#### Usage

```r
data(dbMetaEUStat)
```

#### Format

A tibble dataset with 56 rows and 10 columns

#### Source

[https://ec.europa.eu/eurostat/data/database](https://ec.europa.eu/eurostat/data/database)

#### References


#### Examples

```r
data(dbMetaEUStat)
names(dbMetaEUStat)
```

# Visualize indicators' information:
`dbMetaEUStat$INDICATOR`

# Visualize the indicators' coding in database:
`dbMetaEUStat$Code_in_database`

# Visualize the indicators' official coding:
`dbMetaEUStat$Official_code`
**Description**

Given a dataframe of quantitative indicators along time, the delta convergence is a statistic describing departures from best performer. A time variable may be present or not, but if it is not present then rows must be already sorted. Missing values are not allowed. If the time variable is omitted, subsequent rows are separated by one time unit.

**Usage**

```r
delta_conv(
  tavDes,
  timeName = "time",
  indiType = "highBest",
  time_0 = NA,
  time_t = NA,
  extended = FALSE
)
```

**Arguments**

- **tavDes** the dataframe time by countries.
- **timeName** the name of the variable that contains time information; if it is set to NA then no time information is exploited.
- **indiType** the indicator type; the default is "highBest", otherwise it is equal to "lowBest".
- **time_0** starting time to consider; if NA all times considered.
- **time_t** last time to consider; if NA all times considered.
- **extended** if FALSE only measures of convergence are produced, otherwise the declaration of convergence is also provided.

**Value**

A tibble with the value of delta-conv (called delta) along time, which is called 'time'.

**References**

Examples

# Example 1
# Delta convergence with time present
# Dataframe in the format time by countries:
myTB <- tibble::tribble(
  ~time, ~UK, ~DE, ~IT,
  1988, 1201, 868, 578,
  1989, 1150, 978, 682,
  1990, 998, 1250, 332
)
resDelta <- delta_conv(myTB)

# Example 2
# Delta convergence with scrambled time order (time present):
myTB2 <- tibble::tribble(
  ~time, ~UK, ~DE, ~IT,
  1990, 998, 1250, 332,
  1988, 1201, 868, 578,
  1989, 1150, 978, 682
)
resDelta1 <- delta_conv(myTB2)

# Example 3
# Delta convergence, scrambled time and different name for the time variable:
myTB2 <- tibble::tribble(
  ~years, ~UK, ~DE, ~IT,
  90, 998, 1250, 332,
  88, 1201, 868, 578,
  89, 1150, 978, 682
)
resDelta2 <- delta_conv(myTB2, timeName="years")

# Example 4
# Delta convergence for the emp_20_64_MS Eurofound dataset:
data("emp_20_64_MS")
# check name of the time variable:
names(emp_20_64_MS)

# Calculate delta convergence:
resDelta3 <- delta_conv(emp_20_64_MS)

# Obtain measures of delta-convergence and the declaration of convergence:
resDelta4 <- delta_conv(emp_20_64_MS, extended = TRUE)

demea_change

Calculate changes of deviations from the mean
Description
Deviations from the mean of a collection of countries is calculated for each year. Then differences at subsequent times are calculated within each member state. Finally negative differences are added over years within member state, and similarly positive differences are added over years within member state. The output is made by datasets with intermediate calculations, and by the component statistics which is member state by statistics.

Usage
demea_change(
  myTB,
  timeName = "time",
  time_0 = NA,
  time_t = NA,
  sele_countries = NA,
  doplot = FALSE
)

Arguments
myTB a dataset time by countries
timeName name of the variable representing time
time_0 starting time
time_t ending time
sele_countries selection of countries to display; NA means all countries
doplot if a ggplot2 graphical object desired then TRUE, otherwise it is FALSE

Details
Let
\[ Y_{i,t,m} \]
be the indicator value i at time t for country m. Let
\[ D_{i,t,m} = Y_{i,t,m} - M_{i,t,m} \]
be the departure from the mean at time t. Let
\[ d_{i,t,m} = |D_{i,t,m}| - |D_{i,t,m}| \]
be the difference of absolute values within country m at time t. Then the overall negative and positive changes are
\[ C_n(i, t, m) = \sum_t d_{i,t,m} I_{d < 0}(d) \]
and
\[ C_p(i, t, m) = \sum_t d_{i,t,m} I_{d > 0}(d) \]
demea_change

Value

A list with intermediate and final statistics; list component res_graph is a ggplot2 object if the argument doplot = TRUE; to plot the object use function plot().

References


Examples

# Example 1
# A dataset in the format time by countries:
require(tibble)
testTB <- dplyr::tribble(~time, ~countryA, ~countryB, ~countryC, 
2000, 0.8, 2.7, 3.9,
2001, 1.2, 3.2, 4.2,
2002, 0.9, 2.9, 4.1,
2003, 1.3, 2.9, 4.0,
2004, 1.2, 3.1, 4.1,
2005, 1.2, 3.0, 4.0)
res <- demea_change(testTB, 
timeName="time",
time_0 = 2000,
time_t = 2005,
sele_countries= NA,
doplot=TRUE)
plot(res$res$res_graph)

# Example 2
# Deviations from the mean for the emp_20_64_MS Eurofound dataset
data(emp_20_64_MS)
# Calculate deviations from the mean from 2013 to 2016 for Italy, France and Germany
res1<-demea_change(emp_20_64_MS, 
timeName="time",
time_0 = 2013,
time_t = 2016,
sele_countries= c('IT','FR','DE'),
doplot=TRUE)
plot(res1$res$res_graph)
departure_best

Departures from the best country

Description

For each country the departure from the best performing Member State is calculated. Then, differences are cumulated over years.

Usage

departure_best(oriTB, timeName = "time", indiType = "highBest")

Arguments

oriTB original dataset (tibble) with time by country values.
timeName string with the name of the time variable in oriTB.
indiType the type of indicator 'highBest' (default) or 'lowBest'.

Value

a list with component res which contains the departures from the best performer (for each country and for each year) and the cumulated differences over years.

References


Examples

# Example 1
# Sorted dataframe in the format years by countries:
require(tibble)
testTB <- dplyr::tribble(
  ~time, ~countryA, ~countryB, ~countryC,
  2000, 0.8, 2.7, 3.9,
  2001, 1.2, 3.2, 4.2,
  2002, 0.9, 2.9, 0.1,
  2003, 1.3, 2.9, 1.0,
  2004, 1.2, 3.1, 4.1,
  2005, 1.2, 3.0, 4.0
)

# Departures from the best country according to the indicator higher is the best:
mySTB <- departure_best(testTB,timeName="time",indiType = "highBest")
# Differences from the best country for each year:
mySTB$res$raw_departures
# Sum of the cumulated differences for each country:
mySTB$res$cumulated_dif

# Departures from the best country according to the indicator lower is the best:
mySTB1 <- departure_best(testTB,timeName="time",indiType = "lowBest")

# Example 2
# Departures from the best country for the emp_20_64_MS Eurofound dataset:
mySTB2 <- departure_best(emp_20_64_MS,timeName="time",indiType = "highBest")
mySTB3 <- departure_best(emp_20_64_MS,timeName="time",indiType = "lowBest")

departure_best_plot  
Plot of deviations from the best performer

Description

Deviations from the best performer are added over years and plotted by country.

Usage

departure_best_plot(
  cumulaDifVector,
  mainCountry = NA,
  countries = c(NA, NA),
  displace = 0.25,
  axis_name_y = "Countries",
  val_alpha = 0.95,
  debug = FALSE
)

Arguments

cumulaDifVector  
a vector of cumulated differences, say from a call to departure_best()$res$cumulated_dif, 
with named elements.
mainCountry  
the main country of interest.
countries  
selection of countries to display; NA means all countries
displace  
graphical displacement
axis_name_y  
name of the axis
val_alpha  
transparency value in (0,1].
debug  
a flag to get debug information as msg component

Value

a list with ggplot2 graphical object within res component
# Example 1

# Sorted dataframe in the format years by countries:
require(tibble)
testTB <- dplyr::tribble(
  ~time, ~countryA, ~countryB, ~countryC,
  2000,  0.8,  2.7,  3.9,
  2001,  1.2,  3.2,  4.2,
  2002,  0.9,  2.9,  0.1,
  2003,  1.3,  2.9,  1.0,
  2004,  1.2,  3.1,  4.1,
  2005,  1.2,  3.0,  4.0
)

# Departures from the best country according to the "highBest" indicator:
mySTB <- departure_best(testTB,timeName="time",indiType = "highBest")

# Plot of deviations from the best performer:
departure_best_plot(cumulaDifVector = mySTB$res$cumulated_dif, mainCountry = "countryC",
countries = c("countryA","countryB"),displace = 0.25,
axis_name_y = "Countries",val_alpha = 0.95,debug=FALSE)

# Departures from the best country according to the "lowBest" indicator:
mySTB1 <- departure_best(testTB,timeName="time",indiType = "lowBest")
departure_best_plot(cumulaDifVector = mySTB1$res$cumulated_dif, mainCountry = "countryC",
countries = c("countryA","countryB"),displace = 0.25,
axis_name_y = "Countries",val_alpha = 0.95,debug=FALSE)

# Example 2

# Departures from the best country for the emp_20_64_MS Eurofound dataset:
mySTB2 <- departure_best(emp_20_64_MS,timeName="time",indiType = "highBest")

# Plot of deviations from the best performer with Italy as the country of interest:
departure_best_plot(mySTB2$res$cumulated_dif,
mainCountry = "IT",
countries=c("AT", "DE", "FR","SE","SK"),
displace = 0.25,
axis_name_y = "Countries",
val_alpha = 0.95,
dep=FALSE)

mySTB3 <- departure_best(emp_20_64_MS,timeName="time",indiType = "lowBest")
# Plot of deviations from the best performer with Germany as the country of interest:
departure_best_plot(mySTB3$res$cumulated_dif,
mainCountry = "DE",
countries=c("AT", "SE", "FR","IT","SK"),
displace = 0.25,
axis_name_y = "Countries",
val_alpha = 0.95,
dep=FALSE)
departure_mean

Description

For each country the departure from the average is calculated and a numerical label is created: -1 if smaller than one standard deviation from the mean, +1 if above one standard deviation from the mean, 0 otherwise.

Usage

departure_mean(oriTB, sigmaTB, timeName = "time")

Arguments

oriTB original dataset (tibble) with time by country values.
sigmaTB result from sigma_convergence called on oriTB.
timeName string with the name of the time variable in oriTB.

Value

list of tibbles containing labelled departures from the mean, square difference from the mean, and percentage of deviance.

References


Examples

# Example 1
# The original dataset in the format time by countries:
require(tibble)
testTB <- dplyr::tribble(~time, ~countryA, ~countryB, ~countryC,
2000, 0.8, 2.7, 3.9,
2001, 1.2, 3.2, 4.2,
2002, 0.9, 2.9, 4.1,
2003, 1.3, 2.9, 4.0,
2004, 1.2, 3.1, 4.1,
2005, 1.2, 3.0, 4.0


```r
dev_mean_plot

) # Calculate sigma_convergence on the original dataset:
mySTB <- sigma_conv(testTB)

# Calculate departures from the average for each country:
resDM <- departure_mean(oriTB=testTB, sigmaTB=mySTB$res)
names(resDM$res)

# Example 2: Departures from the average for the Eurofound dataset "emp_20_64_MS"
data(emp_20_64_MS)
# Sigma convergence on the original dataset:
mySC <- sigma_conv(emp_20_64_MS)

# Calculate departures from the mean for each country:
resDMeur <- departure_mean(oriTB = emp_20_64_MS, sigmaTB = mySC$res)

# Results for labelled departures from the mean:
resDMeur$res$departures

# Results for square difference from the mean:
resDMeur$res$squaredContrib

# Results for the percentage of deviance:
resDMeur$res$devianceContrib
```

---

### dev_mean_plot

Plot of deviations from the mean

---

**Description**

Negative deviations and positive deviations are added over years and plotted by country.

**Usage**

```r
dev_mean_plot(
  myTB,
  timeName = "time",
  time_0 = NA,
  time_t = NA,
  countries = c(NA, NA),
  indiType = "highBest",
  displace = 0.25,
  axis_name_y = "Countries",
  val_alpha = 0.95,
  debug = FALSE
)
```

---
Arguments

myTB a dataset time by countries
timeName name of the variable representing time
time_0 starting time
time_t ending time
countries selection of countries to display; NA means all countries
indiType the type of indicator "highBest" or "lowBest"
displace graphical displacement
axis_name_y name of the axis
val_alpha transparency value in (0,1]
debug a flag to get debug information as msg component

Value

a list with ggplot2 graphical object within res component

References


Examples

```r
## Not run:
# Example 1
# A dataset in the format time by countries:
require(tibble)
testTB <- dplyr::tribble(~time, ~countryA, ~countryB, ~countryC,
  2000, 0.8, 2.7, 3.9,
  2001, 1.2, 3.2, 4.2,
  2002, 0.9, 2.9, 4.1,
  2003, 1.3, 2.9, 4.0,
  2004, 1.2, 3.1, 4.1,
  2005, 1.2, 3.0, 4.0)
# Plot the deviations from the mean for all countries:
resDMP <- dev_mean_plot(testTB,
  timeName="time",
  displace = 0.25,
  axis_name_y = "Countries")
resDMP

# Plot by considering only some of the years:
resDMP1 <- dev_mean_plot(testTB,
  timeName="time",
  time_0 = 2002,
```
time_t = 2004,
displace = 0.25,
axis_name_y = "Countries")

resDMP1

# Example 2
# The Eurofound dataset "emp_20_64_MS":
myTB1 <- emp_20_64_MS

# Plot the deviations from the mean only for some of the member states:
resDMP2 <- dev_mean_plot(myTB1,
  timeName="time",
  time_0 = 2005,
  time_t = 2010,
  countries= c("AT","BE","IT"),
  displace = 0.25,
  axis_name_y = "Countries")

resDMP2

## End(Not run)

download_indicator_EUS

Download a dataset (tibble) from Eurostat.

Description

From the Eurostat web site, a dataset is created whose structure is years by countries, possibly conditioned to gender, age class and other variables.

Usage

download_indicator_EUS(
  indicator_code,
  fromTime,
  toTime,
  gender = c(NA, "T", "F", "M")[1],
  ageInterv = NA,
  rawDump = FALSE,
  uniqueIdentif = 1
)
**down_lo_EUS**

**Download a dataset (tibble) from Eurostat.**

**Description**

From the Eurostat web site, a dataset is created whose structure is time by countries, possibly conditioned to gender, age class and other variables. All indicators are supported and, after downloading, data are not filtered by country members (geo) and/or EU clusters.

**Usage**

```r
down_lo_EUS(
    indicator_code,
    fromTime,
    toTime,
    gender = c(NA, "T", "F", "M")[1],
    ageInterv = NA,
    rawDump = FALSE,
    uniqueIdentif = 1
)
```

**Arguments**

- `indicator_code`: the variable describing countries, chosen within the collection `convergEU_glb()$metaEUStat$selectorUser`.
- `fromTime`: first year to be considered.
- `toTime`: last year to be considered.
- `gender`: which gender, one of c("T","F","M") for Total, Females, Males.
- `ageInterv`: a string of character representing the age class to be considered as coded by Eurostat, for example 'Y15-74'.
- `countries`: a collection of strings representing countries in the standard two letters format; the most important sets are stored as a global function `convergEU_glb()$EU27`; if countries = NA, then all available countries are downloaded.
- `rawDump`: if TRUE raw downloaded data are returned, otherwise filtered values are provided.
- `uniqueIdentif`: identifiers of further conditional variables (1,2,...).

**Value**

A dataset (tibble) years by countries, possibly conditioned to gender, within the list as component named res. If rawDump is TRUE then bulk data are provided. The list component msg may contain auxiliary information on conditioning variables.

**References**

down_lo_EUS

Arguments

- `indicator_code`: defined by Eurostat as id.
- `fromTime`: first year to be considered.
- `toTime`: last year to be considered.
- `gender`: if available, the gender of interest c("T","F","M") for Total, Females, Males.
- `ageInterv`: if available, a string of character representing the age class to be considered as coded by Eurostat, for example ‘Y15-74’.
- `rawDump`: if TRUE raw downloaded data are returned, otherwise filtered values are provided.
- `uniqueIdentif`: identifiers of further conditional variables (1, 2,...).

Details

It is up to the user to proceed with further filtering/selection so that the desired collection of member states is obtained.

Value

a dataset (tibble) years by countries, possibly conditioned to gender, within the list as component named res. If rawDump is TRUE then bulk data are provided. The list component msg may contain auxiliary information on conditioning variables.

References


Examples

```r
## Not run:
myDF1 <- down_lo_EUS(indicator_code = "lfsa_ergaed",
  fromTime = 2005,
  toTime = 2015,
  gender = "F",
  ageInterv = NA,
  rawDump = FALSE,
  uniqueIdentif = 1)

myDF2 <- down_lo_EUS(indicator_code = "lfsa_ergaed",
  fromTime = 2005,
  toTime = 2015,
  gender = "M",
  ageInterv = "Y15-64",
  rawDump = FALSE,
  uniqueIdentif = 3)

## End(Not run)
```
dow_soc_scor_boa  Downloader of social scoreboard indicators

Description

This is an "envelope function" to automate the download from Eurostat of all the indicators involved in the social scoreboard.

Usage

dow_soc_scor_boa(fromTime = 1999, toTime = 2018, rm.EU = FALSE)

Arguments

fromTime  starting time

toTime  ending time

rm.EU  is TRUE remove all variables (columns) starting with "EU" and "EA", that is averages for different groups of countries.

Details

Note that the downloaded datasets may have auxiliary columns to be later removed and they may contain missing values, thus before further calculation taking place, imputation or truncation of missing values must be performed. Extra columns include EU12 and other similar weighted averages.

Value

a list with as many components as indicators.

References


emp_20_64_MS  Dataset emp_20_64_MS

Description

Source data provided by Eurofound, and reshaped so that the first column is time and all the other 28 columns are employment values of Member States. The first column refers to the time variable (e.g. years); the remaining 28 columns refer to the Member States; each Member State is identified through its corresponding country code accessible by invoking *convergEU_glb()*Eurozone$memberStates*.
Usage

    data(emp_20_64_MS)

Format

    A data frame with 17 rows and 29 variables

Examples

    data(emp_20_64_MS)
    head(emp_20_64_MS)
    names(emp_20_64_MS)

extract_indicator_ECG Create a dataset (tibble) for an ECG indicator.

Description

    From the ECG database, a dataset is created whose structure is years by countries.

Usage

    extract_indicator_ECG(
        indicator_code,
        fromTime,
        toTime,
        countries = convergEU_glb()$EU27$memberStates$codeMS,
        type_flag = FALSE
    )

Arguments

    indicator_code  one of the following strings: "dataset2", "demo_magec_AND_demo_pjan"
    fromTime        first year to be considered
    toTime          last year to be considered
    countries       a collection of strings representing countries in the standard two letters format
    type_flag       if FALSE data are returned, otherwise the type of indicator is returned; if METADATA_ECG is selected, NA is returned

Details

    If the indicator_code is equal to "METADATA_ECG" then information on available indicators is provided as a dataframe (tibble): names of indicators are contained in the variable "Worksheet name".
extract_indicator_EIGE

Create a dataset (tibble) for an EIGE indicator.

Description

From the EIGE database, a dataset is created whose structure is years by countries.

Usage

```r
extract_indicator_EIGE(
  indicator_code,
  fromTime,
  toTime,
  countries = convergEU_glb()$EU27$memberStates$codeMS,
  type_flag = FALSE
)
```

Arguments

- `indicator_code`: one of the following strings: "INDEX", "WORK", "MONEY", "KNOWLEDGE", "TIME", "POWER", "HEALTH", "FTE", "DWL", "SEGREG", "INCOME", "TERTIARY", "CARE", "HOUSE", "MINISTER", "PARLIAMENT", "BOARD", "HLY", "METADATA"
- `fromTime`: first year to be considered
- `toTime`: last year to be considered

Examples

```r
# Extract metadata:
myTB1 <- extract_indicator_EIGE(
  indicator_code = "METADATA_ECG" #Code_in_database
)

# Extract indicator "dataset2" from 2010 to 2021:
myTB2 <- extract_indicator_EIGE(
  indicator_code = "dataset2", #Code_in_database
  fromTime=2010,
  toTime=2021)
```
extract_indicator_EUF

Create a dataset (tibble) for an indicator.

Description
From the Eurofound database, a dataset is created whose structure is years by countries, possibly conditioned to gender.

Usage

eextract_indicator_EUF(
    indicator_code,
    fromTime,
    toTime,
    gender = c("Total", "Females", "Males")[1],
    countries = convergEU_glb()$EU27$memberStates$codeMS
)
Arguments

- **indicator_code** the variable describing countries
- **fromTime** first year to be considered
- **toTime** last year to be considered
- **gender** which gender, one of c("Total","Females","Males")
- **countries** a collection of strings representing countries in the standard two letters format

Value

a dataset (tibble) years by countries, possibly conditioned to gender

References


Examples

```r
# Extract indicator labelled "lifesatisf" and accessible from "dbEUF2018meta" data:
print(dbEUF2018meta, n=20, width=100)
dbEUF2018meta$Code_in_database
myTB1 <- extract_indicator_EUF(
  indicator_code = "lifesatisf", #Code_in_database
  fromTime=2003,
  toTime=2015,
  gender= c("Total","Females","Males")[1])

# Extract indicator "exposdiscr_p" (Code_in_database) from 2003 to 2016:
myTB2 <- extract_indicator_EUF(
  indicator_code = "exposdiscr_p", #Code_in_database
  fromTime=2003,
  toTime=2016,
  gender= c("Total","Females","Males")[1])

# Extract indicator "lifesatisf" from 1998 to 2016 for females:
myTB3 <- extract_indicator_EUF(
  indicator_code = "lifesatisf", #Code_in_database
  fromTime = 1998,
  toTime = 2016,
  gender = c("Total","Females","Males")[2])

# Extract indicator "lifesatisf" from 1960 to 2016 for males of EU12:
myTB4 <- extract_indicator_EUF(
  indicator_code = "lifesatisf", #Code_in_database
  fromTime=1960,
  toTime=2016,
  gender= c("Total","Females","Males")[3],
  countries= convergEU_glb()$EU12$memberStates$codeMS)
```
Gamma convergence

Description
Given a dataframe (tibble) of times by countries indicator, the gamma convergence is calculated. A time index is required. Missing values are not allowed.

Usage

```r
gamma_conv(rawDat, ref = NA, last = NA, timeName = "time", printRanks = F)
```

Arguments

- `rawDat`: the tibble made by times and countries.
- `ref`: the reference time, typically zero.
- `last`: the last time to be considered.
- `timeName`: the name of the variable that contains time information.
- `printRanks`: logical flag for printing ranks based on data.

Value

gamma convergence (indicated as KIt in Eurofound 2018 paper).

References


Examples

```r
# Example 1
# Dataframe in the format time by countries:
require(tibble)
myTB <- tibble::tribble(
  ~years, ~UK, ~DE, ~IT,
  1990,  998, 1250, 332,
  1988,  1201, 868, 578,
  1989,  1150, 978, 682,
  1991,  1600, 1350, 892
)
# Gamma convergence, scrambled time and different time name:
resGamma <- gamma_conv(myTB, ref=1988, last=1991, timeName="years")
```
# Example 2

```r
myTB1 <- tibble::tribble(
  ~time, ~UK, ~DE, ~IT,
  1990, 998, 1250, 332,
  1988, 1201, 868, 578,
  1989, 1150, 978, 682,
  1991, 1600, 1350, 802
)
resGamma1 <- gamma_conv(myTB1, ref=1989, last=1990)
```

# Example 3

```r
# Gamma convergence for the emp_20_64_MS Eurofound dataset:
data("emp_20_64_MS")
resGamma2<-gamma_conv(emp_20_64_MS,ref=2002,last=2005)
resGamma3<-gamma_conv(emp_20_64_MS,ref=2002,last=2018)
# Print also ranks based on data:
resGamma4<-gamma_conv(emp_20_64_MS,ref=2002,last=2018,printRanks=TRUE)
```

gamma_conv_msteps  

### Description

Given a dataframe (tibble) of sorted times by countries indicator, the gamma convergence is calculated between pairs of subsequent years. A time index is required. Missing values are not allowed.

### Usage

```r
gamma_conv_msteps(rawDat, startTime, endTime, timeName = "time")
```

### Arguments

- **rawDat**: the tibble made by times and countries.
- **startTime**: the first year to consider, included.
- **endTime**: the last year to consider, included.
- **timeName**: the name of the variable that contains time information.

### Value

A dataset of gamma values (indicated as KIt in Eurofound 2018 paper).

### References

Examples

# Example 1
# Dataframe in the format time by countries:
require(tibble)
myTB <- tibble::tribble(~time, ~UK, ~DE, ~IT,
1988, 1201, 868, 578,
1989, 1150, 978, 682,
1990, 998, 1250, 332,
1991, 1600, 1350, 802)
resGammaST <- gamma_conv_msteps(myTB, startTime = 1988, endTime = 1991, timeName = "time"
)

# Example 2
# Gamma convergence iterated for several pairs of years for the emp_20_64_MS Eurofound dataset
data("emp_20_64_MS")
# check name of the time variable
names(emp_20_64_MS)
resGammaST2 <- gamma_conv_msteps(emp_20_64_MS, startTime = 2002, endTime = 2006, timeName = "time")
resGammaST3 <- gamma_conv_msteps(emp_20_64_MS, startTime = 2002, endTime = 2018, timeName = "time")
resGammaST4 <- gamma_conv_msteps(emp_20_64_MS, startTime = 2007, endTime = 2012, timeName = "time")

---

go_indica_fi

Create an indicator fiche for a given aggregation of countries.

Description

An auxiliary function to compile a rmarkdown file to produce the indicator fiche in html format within the output directory.

Usage

go_indica_fi(time_0 = NA, time_t = NA, timeName = NA, workDF = NA, indicaT = NA, indiType = c("highBest", "lowBest")[1], seleMeasure = "all", seleAggre = "EU27", x_angle = 45, data_res_download = FALSE, auth = "A.Student", dataNow = Sys.time(), outFile = NA,
Arguments

- **time_0**: starting time.
- **time_t**: ending time.
- **timeName**: name of the variable containing times (years).
- **workDF**: name (string) of the dataset in the global environment containing all countries contributing to average.
- **indicaT**: name of the considered indicator.
- **indiType**: type of indicator "lowBest" or "highBest" (default).
- **seleMeasure**: set of measures of convergence; this is a subset of the following collection of strings: "beta", "delta", "gamma", "sigma"; "all" is a shortcut for the whole set.
- **seleAggre**: selection of member states, default 'EU27' ('custom' if not pre-coded).
- **x_angle**: axis orientation for time labels, default 45.
- **data_res_download**: should data and results be downloaded, default FALSE.
- **auth**: author of this report, default 'A.Student'.
- **dataNow**: date of production of this country fiche, default is current time.
- **outFile**: name of the output file (without path), without extension.
- **outDir**: output directory, eventually not existing (only one level allowed).
- **pdf_out**: should the output be saved as PDF file? The default is FALSE.
- **workTB**: a tibble containing data.
- **selfContained**: TRUE if just one file is desired
- **eige_layout**: TRUE if the EIGE layout is desired
- **memStates**: Member States visualizations, default is set as quintiles.

Details

Note that most of function arguments are passed as strings of characters instead of object names. For example, if the object of a dataset in the workspace is myTB, the parameter is set like workDF='myTB' instead of workDF=myTB as one may expect. Furthermore, the dataset must be complete, that is without missing values. Note also that Internet connection should be available when invoking the function to properly rendering the results in the html file. The fiches have been tested with the browsers Mozilla Firefox and Google Chrome.

References

Create a country fiche for an indicator

Description
An auxiliary function to compile a rmarkdown file to produce a country fiche in html format within the output directory.

Usage
```r
go_ms_fi(
  workDF = NA,
  countryRef = NA,
  otherCountries = c(NA, NA),
  time_0 = NA,
  time_t = NA,
  tName = NA,
  indiType = NA,
  aggregation = NA,
  x_angle = NA,
  dataNow = NA,
  author = NA,
  outFile = NA,
  outDir = NA,
  indiName = NA,
  workTB = NULL
)
```

Arguments

- **workDF**: name (string) of the dataset with all countries contributing to average
- **countryRef**: country of main interest
- **otherCountries**: other countries for comparison
- **time_0**: starting time
- **time_t**: ending time
- **tName**: name of the variable containing times (years)
- **indiType**: type of indicator "lowBest" or "highBest"
- **aggregation**: label indicator the reference group of countries (’custom’ if not pre-coded)
- **x_angle**: axis orientation for time labels
- **dataNow**: date of production of this country fiche
- **author**: author of this report
- **outFile**: name of the output file (without path)
- **outDir**: output directory, eventually not existing (only one level allowed)
- **indiName**: name of the considered indicator
- **workTB**: tibble containing data, optional, as alternative to a global object.
Details

Note that most of function arguments are passed as strings of characters instead of object names. For example, if the object of a dataset in the workspace is myTB, the parameter is set like workDF='myTB' instead of workDF=myTB as one may expect. Furthermore, the dataset must be complete, that is without missing values. Note also that connection to Internet should be available when invoking the function to properly rendering the results in the html file. A tibble object containing data can be passed with the argument workTB instead of a string.

References


---

```r
graph_departure
```

*Graphical representation based on sigma convergence*

---

Description

A ggplot object countries by time where coloured rectangles show if in that time unit the indicator is below one standard deviation (-1) from the mean, above one standard deviation (-1) from the mean or within 2 standard deviations around the mean.

Usage

```r
graph_departure(
  myTB,
  timeName = "time",
  indiType = "highBest",
  displace = 0.25,
  displaceh = 0.45,
  dimeFontNum = 6,
  myfont_scale = 1.35,
  x_angle = 45,
  color_rect = c("-1" = "red1", "0" = "gray80", "1" = "lightskyblue1"),
  axis_name_y = "Countries",
  axis_name_x = "Time",
  alpha_color = 0.9
)
```

Arguments

- `myTB`: the component $res$departure of an object created by `departure_mean()`
- `timeName`: name of the time variable
- `indiType`: indicator type, one among "highBest" and "lowBest"
- `displace`: rectangle half height
- `displaceh`: rectangle half base
graph_department

dimeFontNum  size of font
myfont_scale axes magnification
x_angle angle of x axis labels
color_rect colors within rectangles; the default for a "highBest" indicator type is red for "-1", grey for "0" and light sky blue for "1"; the default for a "lowBest" indicator type is light sky blue for "-1", grey for "0" and red for "1"
axis_name_y name of y axis
axis_name_x name of x axis
alpha_color transparency

Details

Note that calculation of departure must be already performed by invoking departure_mean.

Value

a list with component $res made by a ggplot object to be displayed or saved using ggsave function.

References


Examples

# Example 1: "lowBest" indicator type:
# Dataframe in the format time by countries:
require(tibble)
testTB <- dplyr::tribble(  
  ~time, ~countryA, ~countryB, ~countryC,  
  2000, 0.8, 2.7, 3.9,  
  2001, 1.2, 3.2, 4.2,  
  2002, 0.9, 2.9, 4.1,  
  2003, 1.3, 2.9, 4.0,  
  2004, 1.2, 3.1, 4.1,  
  2005, 1.2, 3.0, 4.0  
)
mySTB <- sigma_conv(testTB)
resDM <- departure_mean(oriTB=testTB, sigmaTB=mySTB$res)
myG <- NULL
myG <- graph_department(resDM$res$departures,
  timeName = "time",
  indiType = "lowBest",
  displace = 0.25,
  displaceh = 0.45,
  dimeFontNum = 6,
  myfont_scale = 1.35,
  x_angle = 45,
  )
axis_name_y = "Countries",
axis_name_x = "Time",
alpha_color = 0.9)

# Change the colour of rectangles:
myGG <- graph_departure(resDM$res$departures,
timeName = "time",
indiType = "lowBest",
displace = 0.25,
displaceh = 0.45,
dimeFontNum = 6,
myfont_scale = 1.35,
x_angle = 45,
color_rect = c("-1"="green4", "0"="yellow","1"="red"),
axis_name_y = "Countries",
axis_name_x = "Time",
alpha_color = 0.9)

# Example 2: "highBest" type of indicator:
# Graphical plot of sigma convergence for the emp_20_64_MS Eurofound dataset:
data(emp_20_64_MS)
mySC <- sigma_conv(emp_20_64_MS)
resDMeur <- departure_mean(oriTB = emp_20_64_MS, sigmaTB = mySC$res)
myG1 <- NULL
myG1 <- graph_departure(resDMeur$res$departures,
timeName = "time",
indiType = "highBest",
displace = 0.25,
displaceh = 0.45,
dimeFontNum = 6,
myfont_scale = 1.35,
x_angle = 45,
axis_name_y = "Countries",
axis_name_x = "Time",
alpha_color = 0.9)

# Plot mean departures for selected countries only and change the colour of rectangles:
myG2 <- NULL
myG2 <- graph_departure(resDMeur$res$departures[,1:8],
timeName = "time",
indiType = "highBest",
displace = 0.25,
displaceh = 0.45,
dimeFontNum = 6,
myfont_scale = 1.35,
x_angle = 45,
color_rect = c("-1"="red", "0"="yellow","1"="green4"),
axis_name_y = "Countries",
axis_name_x = "Time",
alpha_color = 0.9)
### Description

Gradients values and Delta2 are mapped to one pattern (string and number). See Eurofound 2018 report. In the mapping table within this function +1 means greater than zero, 0 means equal to zero, -1 means smaller than 0. For column EU_vs_MS, if graEU > graMS then EU_vs_MS = +1; if graEU < graMS then EU_vs_MS = -1; if graEU == graMS then EU_vs_MS = 0. Code NA is left to indicate not relevant features. Further codes are added here from 13 to 18 for parallelism; codes 19 and 20 are for crossed lines joining the EU pair and the MS pair. Code 21 stands for "to be visually inspected".

### Usage

```r
gra_de2_patt(vaEU, vaMS, vaTime)
```

### Arguments

- `vaEU`: EU values sorted in ascending order by time.
- `vaMS`: member state values sorted in ascending order by time.
- `vaTime`: sorted pair of times.

### Value

A number referring to pattern whose label depends on the indicator type.

### References


### Examples

```r
# Example 1
vaEU <- c(5,7)
vaMS <- c(6,8)
vaTime <- c(1999,2000)
resG1 <- gra_de2_patt(vaEU, vaMS, vaTime)

# Example 2:
vaEU <- c(7,2)
vaMS <- c(9,4)
vaTime <- c(2009,2010)
resG2 <- gra_de2_patt(vaEU, vaMS, vaTime)

# Example 3:
```
```r
vaTime <- c(2009,2010)
vaEU <- c(100, 120)
vaMS <- c( 50, 90)
resG3 <- gra_de2_patt(vaEU,vaMS,vaTime)
```

---

**impute_dataset**

*Imputation to make a dataset complete*

**Description**

For initial and final missing values there are two options: they could be completely cancelled or, otherwise propagated. For all other missing values within the dataset, deterministic linear imputation is applied in order to obtain complete data.

**Usage**

```r
impute_dataset(
  myTB,
  countries,
  timeName = "time",
  tailMiss = c("cut", "constant")[2],
  headMiss = c("cut", "constant")[1]
)
```

**Arguments**

- `myTB` a dataset (tibble) time by countries for a given indicator, sorted by time. Note that times corresponding to missing data must be contained in the dataset.
- `countries` the collection of labels representing countries to process.
- `timeName` the string that represent the name of the time variable.
- `tailMiss` what should be done with subsequent missing values starting at the oldest year: cut those years, or input constant values equal to the first observed year.
- `headMiss` what should be done with subsequent missing values ending at the last year: cut those years, or input constant values equal to the first observed year.

**Value**

a list with three components: "res": the dataset (tibble) without missing values; "msg" and "err"

**References**

Examples

# Example 1
# Dataset in the format time by countries with missing values:
myTB2 <- tibble::tribble(
  ~time, ~UK, ~DE, ~IT,
  1988, 998, 1250, 332,
  1989, NA, 868, NA,
  1990, 1150, 978, NA,
  1991, 1600, NA, 802
)
toBeProcessed <- c("UK","DE","IT")
# Simplest Imputation using option "cut":
resImpu <- impute_dataset(myTB2, countries=toBeProcessed,
  timeName = "time",
  tailMiss = c("cut", "constant")[[1]],
  headMiss = c("cut", "constant")[[1]])

# Imputation using option "constant":
resImpu1 <- impute_dataset(myTB2, countries=toBeProcessed,
  timeName = "time",
  tailMiss = c("cut", "constant")[[2]],
  headMiss = c("cut", "constant")[[2]])

# Imputation using both options "cut" and "constant":
resImput <- impute_dataset(myTB2, countries=toBeProcessed,
  timeName = "time",
  tailMiss = c("cut", "constant")[[2]],
  headMiss = c("cut", "constant")[[1]])

# Example 2
# Dataset time by countries for the indicator "JQIintensity_i":
myTB <- extract_indicator_EUF(
  indicator_code = "JQIintensity_i",  #Code_in_database
  fromTime= 1965,
  toTime=2016,
  gender= c("Total","Females","Males")[[1]],
  countries= convergEU_glb()$EU27$memberStates$codeMS)

# Imputation of missing values, option "cut":
myTBinp <- impute_dataset(myTB$res, timeName = "time",
  countries=convergEU_glb()$EU27$memberStates$codeMS,
  tailMiss = c("cut", "constant")[[1]],
  headMiss = c("cut", "constant")[[1]])

# Imputation of missing values, option "constant":
myTBinp1 <- impute_dataset(myTB$res, timeName = "time",
  countries=convergEU_glb()$EU27$memberStates$codeMS,
  tailMiss = c("cut", "constant")[[2]],
  headMiss = c("cut", "constant")[[2]])
**impu_det_lin**  
*Imputation of missing values*

**Description**
Imputation is deterministic and based on a straight line between two points.

**Usage**
```r
impu_det_lin(timeIni, timeEnd, timeDelta, indicIni, indicFin)
```

**Arguments**
- `timeIni`: starting time
- `timeEnd`: ending time
- `timeDelta`: collection of times where missing values are located
- `indicIni`: observed value at timeIni
- `indicFin`: observed value at timeEnd

**Value**
imputed tibble with an indicator of missingness (wasMissing).

**References**

**Examples**
```r
# Example 1  
# Simplest Imputation of one missing value between two observed values:
res1 <- impu_det_lin(timeIni= 88,  
                     timeEnd = 90,  
                     timeDelta = 89,  
                     indicIni = 120,  
                     indicFin = 100)

# Example 2  
# Multiple Imputation of missing values:
res2 <- impu_det_lin(timeIni= 90,  
                     timeEnd = 93,  
                     timeDelta=c(91,92),  
                     indicIni = 100,  
                     indicFin = 108)
```
# Multiple Imputation of missing values with delta > 1:
res3 <- impu_det_lin(timeIni= 2000,
timeEnd = 2015,
timeDelta=seq(2005,2010,5),
indicIni = 100,
indicFin = 108)

## Description

Gradients values and Delta2 are mapped to one pattern (string and number). See Eurofound 2018 report.

In the mapping table within this function +1 means greater than zero, 0 means equal to zero, -1 means smaller than 0. For column EU_vs_MS, if graEU > graMS then EU_vs_MS = +1; if graEU < graMS then EU_vs_MS = -1; if graEU == graMS then EU_vs_MS = 0. Code NA is left to indicate not relevant features. Further codes are added here from 13 to 18 for parallelism; codes 19 and 20 are for crossed lines joining the EU pair and the MS pair. Code 21 stands for "to be visually inspected".

## Usage

map_2_patt_39(vaMS, vaEU, vaT, remap = FALSE)

## Arguments

- **vaMS** member state values sorted in ascending order by time.
- **vaEU** EU values sorted in ascending order by time.
- **vaT** sorted pair of times.
- **remap** is FALSE for the original numerical labelling of patterns otherwise TRUE to map to old numerical correspondence.

## Value

a number referring to pattern whose label depends on the indicator type as originally produced in the technical report.
**map_2_patt_59**  
*Values to patterns*

**Description**

Gradients values and Delta2 are mapped to one pattern (string and number). See Eurofound 2018 report.

In the mapping table within this function a pattern number 1-59 is assigned for each combination of EU average vs MS average and their differences called delta_endpoints. Code NA is left to indicate not relevant features. In version 0.6.4 the patterns have been updated from the original 12 to 59 to cover all possible combinations. Code 60 stands for "to be visually inspected". See vignette for further info.

**Usage**

```r
map_2_patt_59(vaMS, vaEU, vaT, remap = FALSE)
```

**Arguments**

- `vaMS`: member state values sorted in ascending order by time.
- `vaEU`: EU values sorted in ascending order by time.
- `vaT`: sorted pair of times.
- `remap`: is FALSE for the original numerical labelling of patterns otherwise TRUE to map to old numerical correspondence.

**Value**

a number referring to pattern whose label depends on the indicator type as originally produced in the technical report.

---

**ma_dataset**  
*Smother based on moving average*

**Description**

The smoother change each value into the average of values around it spanning a window of size kappa. Missing values are not allowed.

**Usage**

```r
ma_dataset(myTB, kappa = 2, timeName = "time")
```
Arguments

myTB          a complete dataset (tibble) time by countries, with just time column and country columns.
kappa        integer greater than 1 as smoothed value, to set the time window of the moving average.
timeName      name of the time variable.

Value

a dataset of smoothed values.

References


Examples

# Example 1
# Smoother based on moving average with k=1.5:
require(tibble)

# Dataset in the format time by countries
myTB <- tibble::tibble(
  time = 2010:2001,
  IT = c(10,14,13,12,9,11,13,17,15,25),
  DE = c(10,11,12,9,14,17,23,29,26,23)
)
resMA1 <- ma_dataset(myTB, kappa=1.5)

# Smoother based on moving average with k=3:
resMA2<-ma_dataset(myTB, kappa=3)

# Example 2
# Smoother based on moving average for the emp_20_64_MS Eurofound dataset:

myTB1 <- emp_20_64_MS[,c("time","IT","DE", "FR")]
# Smoother based on moving average with k=2:
resMAeu<-ma_dataset(myTB1, kappa=2, timeName= "time")

# Smoother based on moving average with k=3:
resMAeu1<-ma_dataset(myTB1, kappa=3, timeName= "time")
**ms_dynam**  
*Member state dynamics*

**Description**

A ggplot object time by countries where coloured rectangles show the departure from the mean after partitioning into intervals (-Inf, m-1 s, m-0.5 s, m+0.5 s, m+1 s, Inf). Note that the following convention is adopted where the colour of labels changes depending on the type of indicator, i.e. "lowBest" or "highBest":

**Usage**

```r
ms_dynam(  
  myTB,  
  timeName = "time",  
  displace = 0.25,  
  displaceh = 0.45,  
  dimeFontNum = 5,  
  myfont_scale = 1.35,  
  x_angle = 45,  
  axis_name_y = "Countries",  
  axis_name_x = "Time",  
  alpha_color = 0.9,  
  indiType = "highBest"  
)
```

**Arguments**

- **myTB**: dataset time by countries.
- **timeName**: a string, name of the time variable.
- **displace**: rectangle half height.
- **displaceh**: rectangle half base.
- **dimeFontNum**: size of font.
- **myfont_scale**: axes magnification.
- **x_angle**: angle of x axis labels.
- **axis_name_y**: name of y axis.
- **axis_name_x**: name of x axis.
- **alpha_color**: transparency.
- **indiType**: is a string: "highBest" or "lowBest" to define the type of indicator.
Details

* \((-\infty, \text{-1 s}]\) is labelled as -1; it is coloured in dark green for "lowBest" type of indicator and in red for "highBest" type of indicator.
* \((\text{-1 s, -0.5 s}]\) is labelled as -0.5; it is coloured in pale green for "lowBest" type of indicator and in yellow (oca) for "highBest" type of indicator.
* \((\text{-0.5 s, 0 s}]\) is labelled as 0; it is coloured in pale yellow for both "lowBest" and "highBest" types of indicators.
* \((0 s, \text{1 s}]\) is labelled as 0.5; it is coloured in yellow (oca) for "lowBest" type of indicator and in pale green for "highBest" type of indicator.
* \((\text{1 s,} \infty]\) is labelled as 1; it is coloured in red for "lowBest" type of indicator and in dark green for "highBest" type of indicator.

Value

a ggplot object to be displayed or saved using ggsave.

References


Examples

# Example 1: "lowBest" type of indicator:
# Dataset in the format time by countries:
require(tibble)
testTB <- dplyr::tribble(
  ~time, ~countryA, ~countryB, ~countryC,
  2000, 0.8, 2.7, 3.9,
  2001, 1.2, 3.2, 4.2,
  2002, 0.9, 2.9, 4.1,
  2003, 1.3, 2.9, 4.0,
  2004, 1.2, 3.1, 4.1,
  2005, 1.2, 3.0, 4.0
)

# Calculate scoreboards for countries:
res<-scoreb_yrs(testTB, timeName = "time")

# Extract the component "sco_level_num" from "res"
resTB<-$res$sco_level_num

# Plot the departures from the mean for each country:
ms_dynam ( resTB,
  timeName = "time",
  displace = 0.25,
  displaceh = 0.45,
  dimeFontNum = 5,
  myfont_scale = 1.35,
  x_angle = 45,
  axis_name_y = "Countries",
)
# Plot the departures from the mean for some years only:
# Extract results from sco_level_num for some years only:
estrattore <- resTB["time"] >= 2001 & resTB["time"] <= 2004
csob_lvl <- dplyr::filter(resTB, estrattore)

# Plot the countries dynamics
ms_dynam (csob_lvl,
  timeName = "time",
  displace = 0.25,
  displaceh = 0.45,
  dimeFontNum = 5,
  myfont_scale = 1.35,
  x_angle = 45,
  axis_name_y = "Countries",
  axis_name_x = "Time",
  alpha_color = 0.9,
  indiType = "lowBest"
)

# Example 2: "highBest" type of indicator:
# Scoreboards of Member States for the emp_20_64_MS Eurofound dataset:
data(emp_20_64_MS)

# Extract the component "sco_level_num
scob_lvl <- scoreb_yrs(emp_20_64_MS, timeName = "time")$res$sco_level_num

# Extract the results from 2009 to 2016
estrattore1 <- scob_lvl["time"] >= 2009 & scob_lvl["time"] <= 2016
scob_lvl1 <- dplyr::filter(scob_lvl, estrattore1)

# Plot the departures from the mean for the EU Member States:
ms_dynam (scob_lvl1,
  timeName = "time",
  displace = 0.25,
  displaceh = 0.45,
  dimeFontNum = 3,
  myfont_scale = 1.35,
  x_angle = 45,
  axis_name_y = "Countries",
  axis_name_x = "Time",
  alpha_color = 0.9,
  indiType = "highBest"
)

# Extract the results for Member States from 2007 to 2012:
estrattore2 <- scob_lvl["time"] >= 2007 & scob_lvl["time"] <= 2012
scob_lvl2 <- dplyr::filter(scob_lvl, estrattore2)

# Plot the departures from the mean:
ms_dynam (scob_lvl2,
  timeName = "time",
  displace = 0.25,
  displaceh = 0.45,
  dimeFontNum = 3,
  myfont_scale = 1.35,
  x_angle = 45,
  axis_name_y = "Countries",
  axis_name_x = "Time",
  alpha_color = 0.9,
  indiType = "highBest"
)
ms_pattern_39

Find patterns for all countries

Description

The input is a time by countries dataset where all countries contributing to the average must be present. The expanded set of qualitative equivalence classes is made by 39 patterns.

Usage

ms_pattern_39(myTB, timeName = "time")

Arguments

myTB  a dataset (tibble) for an indicator, time by countries. The first and last time are respectively the first and last rows of the dataset, which must be time sorted.
timeName  a string with name of the time variable

Details

This is an expanded implementation recently submitted for publications.

Value

the type of pattern
**ms_pattern_59**  
*Find patterns for all countries*

**Description**

The input is a time by countries dataset where all countries contributing to the average must be present. The expanded set of qualitative equivalence classes is made by 39 patterns.

**Usage**

```r
ms_pattern_59(myTB, timeName = "time")
```

**Arguments**

- `myTB`: a dataset (tibble) for an indicator, time by countries. The first and last time are respectively the first and last rows of the dataset, which must be time sorted.
- `timeName`: a string with name of the time variable

**Details**

This is an expanded implementation recently submitted for publications.

**Value**

the type of pattern

---

**ms_pattern_ori**  
*Find patterns for all countries*

**Description**

The input is a time by countries dataset where all countries contributing to the average must be present. Indicators of type 'low is better' are transformed (highestRef - Y), thus the distance from the maximum value for each original observation is calculated.

**Usage**

```r
ms_pattern_ori(myTB, timeName = "time", typeIn = c("highBest", "lowBest"))[1])
```

**Arguments**

- `myTB`: a dataset (tibble) for an indicator, time by countries. The first and last time are respectively the first and last rows of the dataset, which must be time sorted.
- `timeName`: a string with name of the time variable
- `typeIn`: the type of indicator considered 'highBest' (default) or 'lowBest'
**not_in**

**Details**
This is the reference implementation as described by the Eurofound report "Monitoring convergence in the European Union Upward convergence in the EU: Concepts, measurements and indicators", 2018.

**Value**
the type of pattern

**References**

---

**not_in**

**Auxiliary function for membership**

**Description**
A fast check if one or more values are outside a set.

**Usage**
not_in(values, set_collection)

**Arguments**
values one or more values
set_collection a collection of values

**Value**
TRUE if not within or FALSE otherwise

**References**

**Examples**

val<-c(1,2,3,5)
mycol<-c(7,8)
not_in(val,mycol)

val1<-c(1,2,3,5)
mycol1<-c(3,5)
patt_legend

Graphical legend about time patterns

Description

A 5 by 4 plot showing patterns of change along time is made and returned a ggplot object.

Usage

patt_legend(indiType = "highBest")

Arguments

indiType a string equal to "highBest" or "lowBest" to select a type of indicator.

Value

a ggplot object to be plotted using grid.arrange() function.

References


Examples

require(gridExtra)
refGGpat2 <- patt_legend(indiType="lowBest")
refGGpat3 <- patt_legend(indiType="highBest")


**patt_legend_39**

*Graphical legend about time patterns*

**Description**

A plot showing patterns of change along time is made and returned as a list of ggplot objects.

**Usage**

```
patt_legend_39()
```

**Value**

a list of 2 ggplot objects 20 plus 19 patterns.

**References**


**Examples**

```
require(ggplot2)
refGGpat <- patt_legend_39()
```

---

**points2par**

*From points to parameters of a straight line*

**Description**

Given two points on a plane, parameters of a straight line are calculated.

**Usage**

```
points2par(point1, point2)
```

**Arguments**

- `point1`: collection abscissa,ordinate.
- `point2`: collection abscissa,ordinate.

**Value**

collection made by (intercept, slope)
Examples

# Example 1
require(tibble)
myTB <- tribble(
  ~time , ~indic,
  1 , 25,
  10 , 5,
  1 , 10,
  10,  3
)
resparamIT1 <- points2par(as.numeric(myTB[1,]),as.numeric(myTB[2,]))

# Example 2
myTB1 <- tribble(
  ~time , ~indic,
  2 , 25,
  16 , 5,
  1 , 9,
  10,  3,
  34,  4
)
resparamIT2 <- points2par(as.numeric(myTB1[1,]),as.numeric(myTB1[2,]))

# Example 3
myTB2 <- tribble(
  ~time , ~indic,
  5 , 2,
  1 , 15,
  11, 19,
  20, 33,
  25, 14
)
resparamIT3 <- points2par(as.numeric(myTB2[1,]),as.numeric(myTB2[2,]))

---

pop_var (Population variance and standard deviation)

Description

The denominator in n instead of n-1, like in the R base function. Note that missing values are deleted by default.

Usage

pop_var(veval)
scoreb_yrs

Arguments

veval vector of data.

Details

Note that the second argument, if assigned, causes only one summary of object returned.

Value

the variance and standard deviation

References


Examples

myvec<-c(5,2,3,NA,4)
pop_var(myvec)

vec1<-c(10, 20, 15,60,32)
pop_var(vec1)

vec2<-c(NA,NA, 13, 19, 20)
pop_var(vec2)

vec4<-c(seq(from = 5, to = 100, by = 5))
pop_var(vec4)

scoreb_yrs Scoreboard of countries

Description

A scoreboard of countries shows the departure of an indicator level from the average, for each year in the dataset. It also considers one-year changes and the inherent average (and departure) for each year.

Usage

scoreb_yrs(myTB, timeName = "time")
Arguments

myTB original complete dataset (tibble) time by country, ordered by time; only time and countries variables must be present, no average or auxiliary variables at all. Only years of interest must be present and only countries contributing to the average of each year.

timeName string with the name of the time variable in myTB

Value

list of tibbles containing departures and integer labels. Integer values in the result refers to the partition (-Inf, m-1 s, m-0.5 s, m+0.5 s, m+1 s, Inf) where m is the average and s the standard deviation at a given time t; in particular the ordinal is 1 if the interval (-Inf, m -1 s) contains the indicator, it is 2 if the interval ( m-1 s, m-0.5 s) contains the indicator, and so on up to the value 5 that means an indicator value above m + 1 s.

References


Examples

# Example 1
# Dataset in the format years by countries:
require(tibble)
testTB <- dplyr::tribble(
  ~time, ~countryA, ~countryB, ~countryC,
  2000, 0.8, 2.7, 3.9,
  2001, 1.2, 3.2, 4.2,
  2002, 0.9, 2.9, 4.1,
  2003, 1.3, 2.9, 4.0,
  2004, 1.2, 3.1, 4.1,
  2005, 1.2, 3.0, 4.0
)
resTB1 <- scoreb_yrs(testTB, timeName = "time")

# Example 2
# Scoreboard of countries for the emp_20_64_MS Eurofound dataset:
data("emp_20_64_MS")
resTB2 <- scoreb_yrs(emp_20_64_MS, timeName = "time")

---

sigma_conv Sigma-convergence statistic
Description

Given a dataframe of quantitative indicators along time, the sigma convergence is a statistic capturing some convergence features. A time variable must be present whether sorted or not. Missing values are not allowed. Here it is calculated at each observed time. All countries belonging to the reference mean must be included into the dataset.

Usage

sigma_conv(tavDes, timeName = "time", time_0 = NA, time_t = NA)

Arguments

tavDes the dataframe time by countries.
timeName the name of the variable that contains time information.
time_0 starting time to consider; if NA all times considered.
time_t last time to consider; if NA all times considered.

Value

a tibble with the value of sigma convergence (called stdDev or CV) along time, where the original *timeName* is preserved.

References


Examples

# Example 1
# Dataframe in the format time by countries:
require(tibble)
myTB <- tibble::tribble(~years, ~UK, ~DE, ~IT,
  1990, 998, 1250, 332,
  1988, 1201, 868, 578,
  1989, 1150, 978, 682)
reSigConv <- sigma_conv(myTB, timeName = "years")

# Results for the sigma convergence:
reSigConv$res

# Example 2
# Sigma convergence, scrambled time, different name, subset of times:
myTB1 <- tibble::tribble(~years, ~UK, ~DE, ~IT,
  1990, 998, 1250, 332,
  1988, 1201, 868, 578,
sigma_conv_graph

Graphical representation based on sigma convergence

Description

A ggplot of the standard deviation and the coefficient of variation based on the results obtained for sigma-convergence

Usage

sigma_conv_graph(
  sigmaconvOut,
  time_0 = NA,
  time_t = NA,
  aggregation = NA,
  x_angle = 45
)

Arguments

  sigmaconvOut  the output obtained from sigma_conv function.
  time_0        starting time.
  time_t        ending time.
  aggregation   the name of the set of member states for which the sigma-convergence is calculated.
  x_angle       axis orientation for time labels, default 45.

Value

  a ggplot object to be displayed or saved using ggsave.

References


1989, 1150, 978, 682,
1991, 232, 225, 227,
1987, 122, 212, 154
)

reSigConv1 <- sigma_conv(myTB1,timeName="years", time_0 = 1988,time_t = 1990)

# Example 3
# Sigma convergence for the emp_20_64_MS Eurofound dataset:
data("emp_20_64_MS")
reSigConv2 <- sigma_conv(emp_20_64_MS)
reSigConv3 <- sigma_conv(emp_20_64_MS, timeName = "time", time_0 = 2002,time_t = 2004)
reSigConv4 <- sigma_conv(emp_20_64_MS, timeName = "time", time_0 = 2002,time_t = 2016)
**Examples**

```r
# Example 1
# Sigma convergence for the emp_20_64_MS Eurofound dataset in the period 2002-2006:
data(emp_20_64_MS)
reSigConv <- sigma_conv(emp_20_64_MS, timeName = "time", time_0 = 2002, time_t = 2006)

# Graphical plot based on the results for sigma-convergence
reSiggraph <- sigma_conv_graph(reSigConv, 2002, 2006, aggregation = 'EU27')

# Example 2
# Sigma-convergence for the emp_20_64_MS Eurofound dataset in the period 2008-2016:
reSigConv1 <- sigma_conv(emp_20_64_MS, timeName = "time", time_0 = 2008, time_t = 2016)

# Graphical plot based on the results for sigma-convergence
reSiggraph1 <- sigma_conv_graph(reSigConv1, 2008, 2016, aggregation = 'EU27')

# Select different time windows, e.g. 2012-2016 and change x_angle:
reSiggraph2 <- sigma_conv_graph(reSigConv1, 2012, 2016, aggregation = 'EU27', x_angle=90)
```

---

**smoo_dataset**

*Smooستر based on weighting*

**Description**

The smoother substitutes an original raw value $Y(m,i,t)$ of country $m$ indicator $i$ at time $t$ with the weighted average $y(m,i,t) = y(m,i,t-1) (1-w)/2 + w y(m,i,t) + y(m,i,t+1) (1-w)/2$ where $0 < w < 1$. The special case $w=1$ corresponds to no smoothing. In case of missing values an NA is returned. If the weight is outside the interval $(0,1]$ then a NA is returned. The first and last values are smoothed using weights $w$ and $1-w$.

**Usage**

```r
smoo_dataset(myTB, leadW = 1, timeTB = NULL)
```

**Arguments**

- **myTB**: a complete dataset time by countries, with just country columns.
- **leadW**: leading positive weight less or equal to 1.
- **timeTB**: a dataset with the time variable, if a dataset is desired as output

**Value**

a matrix of dataset of smoothed values
Examples

# Example 1
# Dataset in the format time by countries:
myTB <- tibble::tibble(
  time = 2001:2010,
  IT = c(10,14,13,12,9,11,13,17,15,25),
  DE = c(10,11,12,9,14,17,23,29,26,23)
)

# Remove the time variable in order to obtain just country columns and compute smoothed values:
reSMO <- smoo_dataset(myTB[-1], leadW=1)
reSMO1 <- smoo_dataset(myTB[-1], leadW=0.5)

# Add the time variable for tibble in output:
reSMO2 <- smoo_dataset(myTB[-1], leadW=0.5,timeTB= dplyr::select(myTB,time))

# Example 2
# Smoother based on weighting for the emp_20_64_MS Eurofound dataset:
data(emp_20_64_MS)
# Select countries:
myTB <- dplyr::select(emp_20_64_MS, time, IT,DE,FR)
# Compute smoothed values by also adding the time variable to the output:
resSM <- smoo_dataset(dplyr::select(myTB,-time), leadW = 0.2, timeTB= dplyr::select(myTB,time))

---

ts_parlin

Time-indicator serie to straight lines parameters

Description

Given a dataset with first column times and second column the indicator values parameters of time-spliced straight lines are calculated. No checking is performed in input. Time values must differ by a positive constant.

Usage

ts_parlin(dataMat)

Arguments

dataMat two columns (times, indicator) dataset
Value

dataset(tibble) where each row is (times, intercept, slope)

References


Examples

```r
require(tibble)
testTB <- dplyr::tribble(~time, ~countryA, ~countryB, ~countryC,
                        2000, 0.8, 2.7, 3.9,
                        2001, 1.2, 3.2, 4.2,
                        2002, 0.9, 2.9, 4.1,
                        2003, 1.3, 2.9, 4.0,
                        2004, 1.2, 3.1, 4.1,
                        2005, 1.2, 3.0, 4.0)

curcountry <- 2
resPAR <- ts_parlin(testTB[,c(1,curcountry)])

curcountry <- 4
resPAR1 <- ts_parlin(testTB[,c(1,curcountry)])
```

---

**Description**

Convergence and divergence may be strict or weak, upward or downward. The interpretation depends on the type of indicator, that is "highBest" or "lowBest".

**Usage**

```r
upDo_CoDi(
  myTB,
  timeName = "time",
  indiType = "highBest",
  time_0 = NA,
  time_t = NA,
  heter_fun = "pop_var"
)
```
Arguments

- **myTB**: time by member states dataset. No other variables can be in the dataset.
- **timeName**: name of the variable that contains time.
- **indiType**: a string, "lowBest" or "highBest".
- **time_0**: reference time.
- **time_t**: target time strictly larger than time_0.
- **heter_fun**: function to summarize dispersion, like var(), sd(); user-developed function are allowed; `pop_var` is the variance with denominator n.

Details

Note that if the argument heter_fun is set to sd or var, then those statistics use a denominator which is n-1, i.e. the number of observations decreased by 1. This is not typically what one wants here, thus the function `pop_var` may be used instead, because it adopts n as denominator. It is also possible to map a summary of dispersion with a monotonic function, like `sqrt` (see examples).

All the Member states contributing to the mean must be columns of the dataset given as input.

Value

- list of declarations.

References


Examples

```r
# using the standard deviation
upDo_CoDi(emp_20_64_MS,
  timeName = "time",
  indiType = "highBest",
  time_0 = 2010,
  time_t = 2015,
  heter_fun = "var" # watchout the denominator here is n-1
)

# using the standard pop_var function
upDo_CoDi(emp_20_64_MS,
  timeName = "time",
  indiType = "highBest",
  time_0 = 2010,
  time_t = 2015,
  heter_fun = "pop_var" # the denominator here is n
)
```
# using personalized summary of dispersion
diffQQmu <- function(vettore){
  (quantile(vettore,0.75)-quantile(vettore,0.25))/mean(vettore)
}

upDo_CoDi(emp_20_64_MS,
  timeName = "time",
  indiType = "highBest",
  time_0 = 2010,
  time_t = 2015,
  heter_fun = "diffQQmu"
)
Index

* datasets
  dbEUF2018meta, 15
  dbEurofound, 16
  dbMetaEUStat, 18
  emp_20_64_MS, 32

  abso_change, 3
  average_clust, 4

  beta_conv, 6
  beta_conv_graph, 8

  check_country, 9
  check_data, 10
  coeu_grad, 11
  coeu_gradV, 12
  compo_cond_EUS, 12
  convergEU_glb, 13
  country_ranking, 14

  dbEUF2018meta, 15
  dbEurofound, 16
  dbMetaEUStat, 18
  delta_conv, 19
  demea_change, 20
  departure_best, 23
  departure_best_plot, 24
  departure_mean, 26, 43
  dev_mean_plot, 27
  dow_soc scor boa, 32
  down lo EUS, 30
  download_indicator_EUS, 29

  emp_20_64_MS, 32
  extract_indicator_ECG, 33
  extract_indicator_EIGE, 34
  extract_indicator_EUF, 35

  gamma_conv, 37
  gamma_conv_msteps, 38
  go_indica f1, 39

  go ms fi, 41
  gra de2 patt, 45
  graph_departure, 42
  impu det lin, 48
  impute_dataset, 46

  ma dataset, 50
  map 2 patt 39, 49
  map 2 patt 59, 50
  ms dynam, 52
  ms pattern 39, 55
  ms pattern 59, 56
  ms pattern ori, 56

  not in, 57

  patt legend, 58
  patt legend 39, 59
  points2par, 59
  pop var, 60

  scoreb yrs, 61
  sigma conv, 62
  sigma conv graph, 64
  smoo dataset, 65

  ts parlin, 66

  upDo CoDi, 67