

Package: RGAP (via r-universe)

September 10, 2024

Type Package

Title Production Function Output Gap Estimation

Version 0.1.1

Depends R (>= 3.1.0)

Description The output gap indicates the percentage difference between the actual output of an economy and its potential. Since potential output is a latent process, the estimation of the output gap poses a challenge and numerous filtering techniques have been proposed. 'RGAP' facilitates the estimation of a Cobb-Douglas production function type output gap, as suggested by the European Commission (Havik et al. 2014) [<https://ideas.repec.org/p/euf/ecopap/0535.html>](https://ideas.repec.org/p/euf/ecopap/0535.html). To that end, the non-accelerating wage rate of unemployment (NAWRU) and the trend of total factor productivity (TFP) can be estimated in two bivariate unobserved component models by means of Kalman filtering and smoothing. 'RGAP' features a flexible modeling framework for the appropriate state-space models and offers frequentist as well as Bayesian estimation techniques. Additional functionalities include direct access to the 'AMECO' [<https://economy-finance.ec.europa.eu/economic-research-and-databases/economic-databases/ameco-database_en>](https://economy-finance.ec.europa.eu/economic-research-and-databases/economic-databases/ameco-database_en) database and automated model selection procedures. See the paper by Streicher (2022) [<http://hdl.handle.net/20.500.11850/552089>](http://hdl.handle.net/20.500.11850/552089) for details.

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Imports stats, KFAS, zoo, dlm, openxlsx, ggplot2, gridExtra

Suggests testthat (>= 3.0.0), R.rsp

VignetteBuilder R.rsp

Config/testthat/edition 3

Repository <https://sinast3000.r-universe.dev>

RemoteUrl <https://github.com/sinast3000/rgap>

RemoteRef HEAD

RemoteSha 81b0c1bfe849f2d0b674fd69bee9f591fe0dc127

Contents

amecoData2input	3
autoGapProd	4
cubs	7
fetchAmecoData	8
fit	10
fit.KuttnerModel	11
fit.NAWRUmodel	12
fit.TFPmodel	16
gap	19
gapHP	20
gapProd	21
hpfilter	22
indicator	23
initializeExo	24
initializePrior	24
initializeRestr	25
is.gap	26
is.KuttnerModel	26
is.NAWRUmodel	27
is.TFPmodel	27
KuttnerModel	28
NAWRUmodel	30
plot.gap	32
plot.KuttnerFit	33
plot.NAWRUfit	35
plot.TFPfit	36
predict.fit	37
print.gap	38
print.KuttnerFit	38
print.KuttnerModel	39
print.NAWRUfit	39
print.NAWRUmodel	40
print.TFPfit	40
print.TFPmodel	41
TFPmodel	41
trendAnchor	43

Index

45

amecoData2input *Data for estimation*

Description

Computes the necessary input data for the EC output gap estimation on the basis of AMECO data.

Usage

```
amecoData2input(tslAmeco, alpha = 0.65)
```

Arguments

<code>tslAmeco</code>	A time series list or a multiple time series object containing AMECO data.
<code>alpha</code>	A number between 0 and 1 indicating the labor share. The default is <code>alpha = 0.65</code> .

Details

The list of time series `tslAmeco` needs to have the following components:

- popw** Population: 15 to 64 years (unit: 1000 persons, code: NPAN)
- ur** Unemployment rate, total; Member States: definition EUROSTAT (unit: Percentage of civilian labor force, code: ZUTN)
- etd** Employment, persons: total economy (National accounts) (unit: 1000 persons, code: NETN)
- et** Employment, persons: all domestic industries (National accounts) (unit: 1000 persons, code: NETD)
- eet** Employees, persons: all domestic industries (National accounts) (unit: 1000 persons, code: NWTD)
- pconsp** Price deflator private final consumption expenditure (unit: National currency reference year = 100, code: PCPH)
- ngdp** Gross domestic product at current prices (unit: bn National currency, code: UVGD)
- gdp** Gross domestic product at constant prices (unit: bn National currency, code: OVGd)
- l** Total annual hours worked: total economy (unit: millions, code: NLHT)
- wtotal** Compensation of employees: total economy (unit: bn National currency, code: UWCD)
- nulc** Nominal unit labour costs: total economy (Ratio of compensation per employee to real GDP per person employed.) (unit: National currency reference year = 100, code: PLCD)
- k** Net capital stock at constant prices: total economy (unit: bn National currency, code: OKND)

Value

A list of time series containing the same components as the input list `tslAmeco` and the following additional components:

<code>gdpdefl</code>	Gross domestic product deflator
<code>tfp</code>	Total factor productivity
<code>lfnd</code>	Labor force non-domestic (unit: 1000 persons)
<code>parts</code>	Participation rate
<code>ahours</code>	Average hours worked (unit: hours)
<code>prod</code>	Labor productivity (unit: real output in millions per person)
<code>tot</code>	Terms of trade ($pconcp / gdpdefl$)
<code>ws</code>	Wage share (unit: compensation per unit of nominal output)
<code>winfl</code>	Wage inflation
<code>rulc</code>	Real unit labor costs

Examples

```
# load data for Germany
data("gap")
country <- "Germany"
tsListRaw <- gap[[country]]
tsListInput <- amecoData2input(tslAmeco = tsListRaw)
```

`autoGapProd`

Fit best production function model

Description

Finds the most suitable model for the NAWRU and the TFP trend according to the BIC or the RMSE. The function computes the output gap based on the chosen models.

Usage

```
autoGapProd(
  tsl,
  type = "hp",
  q = 0.01,
  method = "MLE",
  criterion = "BIC",
  fast = TRUE,
  nModels = 5,
  nawruPoss = list(maxCycleLag = 2, trend = c("RW2", "DT"), cycle = c("AR1", "AR2"),
    errorARmax = 1, errorMAMax = 0, type = c("TKP", "NKP"), exoNames = c("ws", "prod",
    "tot"), signalToNoise = NULL),
  tfpPoss = list(maxCycleLag = 2, trend = c("RW2", "DT"), cycle = c("AR1", "AR2"),
```

```

    "RAR2"), cubsARmax = 0, errorARmax = 1, errorMAMax = 0, signalToNoise = NULL),
    auto = "gap"
)

```

Arguments

ts1	A list of time series objects, see details.
type	The variance restriction type. Possible options are "basic", "hp", see initializeRestr. The default is type = "hp".
q	Quantile for the Inverse Gamma distribution (only used if type = "hp"), see initializeRestr. The default is q = 0.01.
method	The estimation method. Options are maximum likelihood estimation "MLE" and bayesian estimation "bayesian". If method = c("MLE", "bayesian") the NAWRU is fitted by MLE and the TFP trend by Bayesian methods. The default is method = "MLE".
criterion	Model selection criterion. Options are the Bayesian information criterion "BIC" and the root mean squared error "RMSE", both computed for the second observation equation. The default is criterion = "BIC". For Bayesian estimation and criterion = "RMSE", the mean RMSE is used.
fast	Boolean, indicating whether a "fast" procedure should be used, see details.
nModels	Integer, the maximum number of models for each unobserved component model.
nawruPoss	List with possible model specifications for the NAWRU, see details.
tfpPoss	List with possible model specifications for the NAWRU, see details.
auto	If auto = "NAWRU" or auto = "TFP", the function only finds the most suitable NAWRU or TFP model, respectively. The default is auto = "gap".

Details

For fast = TRUE, the function pre-selects suitable models by applying the following procedure: A HP-filtered trend is computed based on which the best trend and cycle models are chosen according to the BIC. Also based on the HP trend, a variety of different specifications for the second observation equation are estimated in a univariate regression and the best models are selected via the BIC. The nModels best models are subsequently estimated in the usual bivariate unobserved component model. For fast = FALSE, a variety of models is estimated in the usual bivariate unobserved component framework.

The input component nawruPoss is a list containing a (sub-) set of the following components:

maxCycleLag Maximum cycle lag included in the second observation equation.

trend Trend model specification.

cycle Cycle model specification.

errorARmax Maximum autoregressive order of the error term in the second observation equation.

errorMAMax Maximum moving average order of the error term in the second observation equation.

type Type of Phillip's curve.

exoNames Names of the exogenous variables potentially included in the Phillip's curve (need to be included in the list of time series `ts1`).

signal-to-noise Signal-to-noise ratio.

The input component `tfpPoss` is a list containing a (sub-) set of the following components:

maxCycleLag Maximum cycle lag included in the second observation equation.

trend Trend model specification.

cycle Cycle model specification.

cubsARmax Maximum CUBS autoregressive order.

errorARmax Maximum autoregressive order of the error term in the second observation equation.

errorMAMax Maximum moving average order of the error term in the second observation equation.

signal-to-noise Signal-to-noise ratio.

The list of time series `ts1` needs to have the following components (plus those series included in the list component `exoNames` in `nawruPoss`):

ur Unemployment rate.

nulc Nominal Unit labor costs, if `type = "TKP"`.

rulc Real unit labor costs, if `type = "NKP"`.

tfp Total factor productivity.

cubs Capacity utilization economic sentiment indicator.

lfnd Labor force non-domestic (unit: 1000 persons).

parts Participation rate.

ahours Average hours worked (unit: hours).

gdp Gross domestic product at constant prices (unit: bn National currency, code: OVGD).

k Net capital stock at constant prices: total economy (unit: bn National currency, code: OKND).

popw Population: 15 to 64 years (unit: 1000 persons, code: NPAN).

The set of tested models is extensive but not exhaustive. The best model is solely based on convergence and the chosen criterion (RMSE or BIC). A manual check of the results is highly recommended.

In some cases, more than `nModels` are checked. For instance, if a re-parametrized and regular AR(2) process are options for the cycle.

Value

A list containing three components: `gap` (the best model of class "gap"), `tfp` (a nested list of TFP models, fitted objects and model fit criteria), `nawru` (a nested list of NAWRU models, fitted objects and model fit criteria). The lists `nawru` and `tfp` contain a list of models, a list of fitted objects and a dataframe `info`, which contains

<code>loglik</code>	log-likelihood function at optimum
<code>AIC</code>	Akaike information criterion

BIC	Bayesian information criterion
HQC	Hannan-Quinn information criterion
RMSE	Root mean squared error
R2	Coefficient of determination (R squared)
signal-to-noise	Signal-to-noise ratio
LjungBox	p-value of Ljung-Box test for autocorrelation (H0 = no autocorrelation)
convergence	0 indicates convergence of the optimization
rrange	relative range of trend series w.r.t original series
neg	1 indicates that negative values are present in the trend series
rev	relative excess volatility w.r.t. original series (stationary series)
rsd	relative standard deviation w.r.t. original series (stationary series)
cor	correlation between trend and original series (stationary series)
msdtg	mean standardized deviation (stationary trend)
magtg	mean absolute growth of trend (stationary trend)
drop	1 indicates the model should be dropped

cubs

CUBS indicator

Description

Computes the capacity utilization economic sentiment (CUBS) indicator.

Usage

```
cubs(tsCU, tsVA, frequency = 1, lambda = NULL)
```

Arguments

tsCU	A multiple time series containing three survey time series, the first element needs to be capacity utilization in industry, see details. Alternatively, a list of time series can be supplied.
tsVA	A multiple time series containing three value added series that correspond to tsCU. Alternatively, a list of time series can be supplied.
frequency	The frequency of the computed cubs indicator. Possible entries are frequency = 1 (annual), frequency = 4 (quarterly). The default is frequency = 1.
lambda	The smoothing parameter for the application of the HP filter (see details). If not supplied, lambda = 6.25 is used for yearly data and lambda = 1600 for quarterly data.

Details

The list `tsLCU` contains capacity utilization in industry, and the relevant survey outcomes of the construction and service sector. The first list object needs to contain capacity utilization in industry.

The list `tslVA` contains the real value added series for the industry, construction and service sector in the same order as `tsLCU`.

The computed CUBS indicator consists exclusively of capacity utilization in industry until both other series become available.

Value

A list containing the two time series capacity utilization in industry `cu` and the CUBS indicator `cubs`.

Examples

```
# load data for Germany
data("gap")
country <- "Germany"

# compute cubs indicator
namesCubs <- c("indu", "serv", "buil")
namesVACubs <- paste0("va", namesCubs)
tscubs <- cubs(
  tsCU = gap[[country]][, namesCubs],
  tsVA = gap[[country]][, namesVACubs]
)
```

fetchAmecoData

Current 'AMECO' data vintage

Description

Fetches the 'AMECO' data for the EC output gap estimation from the current vintage.

Usage

```
fetchAmecoData(country = NULL, cubs = TRUE)
```

Arguments

<code>country</code>	The country name. If left unspecified, data for all countries will be returned.
<code>cubs</code>	A logical indicating whether the CUBS indicator should be computed if possible (see details).

Details

For the computation of CUBS, the following three seasonally adjusted series are used: the utilization indicators in the service industry, the building and construction industry, and capacity utilization in manufacturing/industry.

The confidence indicator in the service industry is composed of question 1, 2, and 3 of the monthly service sector survey $((Q1 + Q2 + Q3)/3)$. The underlying survey questions are as follows:

- Q1 Business situation development over the past 3 months
- Q2 Evolution of the demand over the past 3 months
- Q3 Expectation of the demand over the next 3 months

The confidence indicator in the building and construction industry is composed of question 3 and 4 of the monthly building and construction sector survey $((Q3 \text{ and } Q4)/2)$. The underlying survey questions are as follows:

- Q3 Evolution of your current overall order books
- Q4 Employment expectations over the next 3 months

The indicator for capacity utilization in manufacturing/industry is based on question 13 of the quarterly industry sector survey. The underlying survey question is as follows:

- Q3 Current level of capacity utilization

Value

A list with multiple time series objects for each country. If country is specified, a multiple time series object is returned. For each country, the following series are included:

popw	Population: 15 to 64 years (unit: 1000 persons, code: NPAN)
ur	Unemployment rate, total; Member States: definition EUROSTAT (unit: Percentage of civilian labor force, code: ZUTN)
etd	Employment, persons: total economy (National accounts) (unit: 1000 persons, code: NETN)
et	Employment, persons: all domestic industries (National accounts) (unit: 1000 persons, code: NETD)
eet	Employees, persons: all domestic industries (National accounts) (unit: 1000 persons, code: NWTD)
vaind	Gross value added at constant prices: manufacturing industry (unit: bn National currency, code: OVG5)
vaserv	Gross value added at constant prices: services (unit: bn National currency, code: OVG5)
vabuil	Gross value added at constant prices: building and construction (unit: bn National currency, code: OVG4)
pconsp	Price deflator private final consumption expenditure (unit: National currency reference year = 100, code: PCPH)
cpih	Harmonised consumer price index (All-items, 2015 = 100, code: ZCPIH)

cpin	National consumer price index (All-items, 2015 = 100, code: ZCPIN)
ngdp	Gross domestic product at current prices (unit: bn National currency, code: UVGD)
gdp	Gross domestic product at constant prices (unit: bn National currency, code: OVGd)
gdpdefl	Price deflator gross domestic product (unit: National currency reference year = 100, code: PVGD)
ahours	Average annual hours worked per person employed (unit: Hours, code: NLHA)
l	Total annual hours worked: total economy (unit: millions, code: NLHT)
wtotal	Compensation of employees: total economy (unit: bn National currency, code: UWCD)
nulc	Nominal unit labour costs: total economy (Ratio of compensation per employee to real GDP per person employed.) (unit: National currency reference year = 100, code: PLCD)
k	Net capital stock at constant prices: total economy (unit: bn National currency, code: OKND)
serv	Confidence indicator in the service industry
buil	Confidence indicator in the bulding and construction industry
indu	Capacity utilization in manufacturing/industry

Additionally, if `cubs = TRUE`, the capacity utilization economic sentiment indicator `cubs` will be returned.

Source

https://economy-finance.ec.europa.eu/economic-research-and-databases/economic-databases/ameco-database/download-annual-data-set-macro-economic-database-ameco_en

https://economy-finance.ec.europa.eu/economic-forecast-and-surveys/business-and-consumer-surveys_en

<code>fit</code>	<i>Fit Method</i>
------------------	-------------------

Description

Defines the fit method.

Usage

```
fit(model, ...)
```

Arguments

<code>model</code>	Some model.
<code>...</code>	Some stuff passed on to methods.

Value

Depends on the model object, see documentation of specific methods.

See Also

Other fitting methods: [fit.KuttnerModel\(\)](#), [fit.NAWRUmodel\(\)](#), [fit.TFPmodel\(\)](#)

<code>fit.KuttnerModel</code>	<i>Maximum likelihood estimation of a KuttnerModel</i>
-------------------------------	--

Description

Estimates a two-dimensional state-space model and performs filtering and smoothing to obtain the output gap.

Usage

```
## S3 method for class 'KuttnerModel'
fit(
  model,
  parRestr = initializeRestr(model),
  signalToNoise = NULL,
  control = NULL,
  ...
)
```

Arguments

<code>model</code>	An object of class <code>KuttnerModel</code> .
<code>parRestr</code>	A list of matrices containing the parameter restrictions for the cycle, trend, and the inflation equation. Each matrix contains the lower and upper bound of the involved parameters. NA implies that no restriction is present. Autoregressive parameters are automatically restricted to the stationary region unless box constraints are specified. By default, <code>parRestr</code> is initialized by the function <code>initializeRestr(model)</code> .
<code>signalToNoise</code>	(Optional) signal to noise ratio.
<code>control</code>	(Optional) A list of control arguments to be passed on to <code>optim</code> .
<code>...</code>	additional arguments to be passed to the methods functions.

Value

An object of class `KuttnerFit` containing the following components:

<code>model</code>	The input object of class <code>KuttnerModel</code> .
<code>SSMfit</code>	The estimation output from the function <code>fitSSM</code> from KFAS.
<code>SSMout</code>	The filtering and smoothing output from the function <code>KFS</code> from KFAS.

parameters	A data frame containing the estimated parameters, including standard errors, t-statistics, and p-values.
fit	A list of model fit criteria (see below).
call	Original call to the function.

The list component `fit` contains the following model fit criteria:

loglik	Log-likelihood function value.
AIC	Akaike information criterion.
BIC	Bayesian information criterion.
AIC	Hannan-Quinn information criterion.
RMSE	Root mean squared error of the inflation equation.
R2	R squared of the inflation equation.
LjungBox	Ljung-Box test output of the inflation equation.

See Also

Other fitting methods: [fit.NAWRUmodel\(\)](#), [fit.TFPmodel\(\)](#), [fit\(\)](#)

Examples

```
# load data for the Netherlands
data("gap")
country <- "Netherlands"
tsList <- as.list(gap[[country]][, c("cpih", "gdp")])
tsList$infl <- diff(tsList$cpih)
model <- KuttnerModel(tsl = tsList, trend = "RW2", cycleLag = 1, cycle = "AR2", start = 1980)

# estimate Kutter's model
parRestr <- initializeRestr(model = model, type = "hp")

gapKuttner <- fit(model, parRestr, signalToNoise = 1 / 10)
```

fit.NAWRUmodel	<i>Estimation of a NAWRUmodel</i>
----------------	-----------------------------------

Description

Estimates a two-dimensional state-space model and performs filtering and smoothing to obtain the NAWRU using either maximum likelihood estimation or bayesian methods.

Usage

```
## S3 method for class 'NAWRUmodel'
fit(
  model,
  parRestr = initializeRestr(model = model),
  signalToNoise = NULL,
  method = "MLE",
  control = NULL,
  prior = initializePrior(model),
  R = 10000,
  burnin = ceiling(R/10),
  thin = 1,
  HPDIprob = 0.85,
  pointEstimate = "mean",
  MLEfit = NULL,
  ...
)
```

Arguments

model	An object of class NAWRUmodel.
parRestr	A list of matrices containing the parameter restrictions for the cycle, trend, and the Phillip's curve. Each matrix contains the lower and upper bound of the involved parameters. NA implies that no restriction is present. Autoregressive parameters are automatically restricted to the stationary region unless box constraints are specified. By default, parRestr is initialized by the function initializeRestr(model). Only used if method = "MLE".
signalToNoise	(Optional) signal to noise ratio. Only used if method = "MLE".
method	The estimation method. Options are maximum likelihood estimation "MLE" and bayesian estimation "bayesian". The default is method = "MLE".
control	(Optional) A list of control arguments to be passed on to optim.
prior	A list of matrices with parameters for the prior distribution and box constraints. By default, prior is initialized by initializePrior(model). See details. Only used if method = "bayesian".
R	An integer specifying the number of MCMC draws. The default is R = 10000. Only used if method = "bayesian".
burnin	An integer specifying the burn-in phase of the MCMC chain. The default is burnin = ceiling(R / 10). Only used if method = "bayesian".
thin	An integer specifying the thinning interval between consecutive draws. The default is thin = 1, implying that no draws are dopped. For thin = 2, every second draw is dropped and so on. Only used if method = "bayesian".
HPDIprob	A numeric in the interval (0, 1) specifying the target probability of the highest posterior density intervals. The default is HPDIprob = 0.9. Only used if method = "bayesian".

pointEstimate	Posterior distribution's statistic of central tendency. Possible options are "mean" and "median". The default is pointEstimate = "mean". Only used if method = "bayesian".
MLEfit	(Optional) An object of class NAWRUfit which is used for initialization. Only used if method = "bayesian".
...	additional arguments to be passed to the methods functions.

Details

The list object prior contains three list elements cycle, trend, and pcInd. Each list element is a $4 \times n$ matrix where n denotes the number of parameters involved in the respective equation. The upper two elements specify the distribution, the lower two parameters specify box constraints. NA denotes no constraints. Autoregressive parameters are automatically restricted to the stationary region unless box constraints are specified. For instance, prior\$cycle[, 1] contains the mean, standard deviation, lower and upper bound for the first variable, in that respective order.

The respective prior distributions are defined through their mean and standard deviation.

The Gibbs sampling procedure is as follows. For each $r = 1, \dots, R$

- The states are sampled by running the Kalman filter and smoother conditional on the parameters of the previous step, θ_{r-1}
- Trend equation parameters θ_{trend} : Conditional on the states α_r , a draw $\theta_{trend,k}$ is obtained either by a sequential Gibbs step, a Metropolis Hasting step, or by conjugacy, depending on the trend model specification.
- Cycle equation parameters θ_{cycle} : Conditional on the states α_r , a draw $\theta_{cycle,k}$ is obtained either by a sequential Gibbs step, a Metropolis Hasting step, or by conjugacy, depending on the cycle model specification.
- Phillip's curve equation parameters θ_{pcInd} : Conditional on the states α_r , a draw $\theta_{pcInd,k}$ is obtained either by a sequential Gibbs step, a Metropolis Hasting step, a combination thereof, or by conjugacy, depending on the Phillip's curve equation specification.

Value

For maximum likelihood estimation, an object of class NAWRUfit containing the following components:

model	The input object of class NAWRUmodel.
SSMfit	The estimation output from the function fitSSM from KFAS.
SSMout	The filtering and smoothing output from the function KFS from KFAS.
parameters	A data frame containing the estimated parameters, including standard errors, t-statistics, and p-values.
parRestr	A list of matrices containing the enforced parameter constraints.
fit	A list of model fit criteria (see below).
call	Original call to the function.

The list component fit contains the following model fit criteria:

loglik	Log-likelihood function values.
AIC	Akaike information criterion.
BIC	Bayesian information criterion.
AIC	Hannan-Quinn information criterion.
RMSE	root mean squared error of the Phillip's curve equation.
R2	R squared of the Phillip's curve equation.
LjungBox	Ljung-Box test output of the Phillip's curve equation.
signal-to-noise	Signal-to-noise ratio.

For bayesian estimation, an object of class `NAWRUfit` containing the following components:

model	The input object of class <code>NAWRUmodel</code> .
ts1	A list of time series containing the estimated states.
parameters	A data frame containing the estimated parameters, including standard errors, highest posterior density credible sets.
prior	A list of matrices containing the used prior distributions.
fit	A list of model fit criteria (see below).
call	Original call to the function.

The list component `fit` contains the following model fit criteria:

R2	R squared of the phillips curve equation,
signal-to-noise	Signal-to-noise ratio.

See Also

Other fitting methods: `fit.KuttnerModel()`, `fit.TFPmodel()`, `fit()`

Examples

```
# define nawru model for France
data("gap")
country <- "France"
tsList <- amecoData2input(gap[[country]])
model <- NAWRUmodel(ts1 = tsList)
# estimate nawru model via MLE
parRestr <- initializeRestr(model = model, type = "hp")

f <- fit(model = model, parRestr = parRestr)

# initialize priors and estimate model via Bayesian methods
prior <- initializePrior(model = model)

f <- fit(model = model, method = "bayesian", prior = prior, R = 5000, thin = 2)
```

fit.TFPmodel

*Estimation of a TFPmodel***Description**

Estimates a two-dimensional state-space model and performs filtering and smoothing to obtain the TFP trend using either maximum likelihood estimation or bayesian methods.

Usage

```
## S3 method for class 'TFPmodel'
fit(
  model,
  parRestr = initializeRestr(model = model),
  signalToNoise = NULL,
  method = "MLE",
  control = NULL,
  prior = initializePrior(model),
  R = 10000,
  burnin = ceiling(R/10),
  thin = 1,
  HPDIprob = 0.85,
  pointEstimate = "mean",
  MLEfit = NULL,
  ...
)
```

Arguments

model	An object of class TFPmodel.
parRestr	A list of matrices containing the parameter restrictions for the cycle, trend, and the CUBS equation. Each matrix contains the lower and upper bound of the involved parameters. NA implies that no restriction is present. Autoregressive parameters are automatically restricted to the stationary region unless box constraints are specified. By default, parRestr is initialized by the function initializeRestr(model). Only used if method = "MLE".
signalToNoise	(Optional) signal to noise ratio. Only used if method = "MLE".
method	The estimation method. Options are maximum likelihood estimation "MLE" and bayesian estimation "bayesian". The default is method = "MLE".
control	(Optional) A list of control arguments to be passed on to optim.
prior	A list of matrices with parameters for the prior distribution and box constraints. By default, prior is initialized by initializePrior(model). See details. Only used if method = "bayesian".
R	An integer specifying the number of MCMC draws. The default is R = 10000. Only used if method = "bayesian".

burnin	An integer specifying the burn-in phase of the MCMC chain. The default is $\text{burnin} = \text{ceiling}(R / 10)$. Only used if <code>method = "bayesian"</code> .
thin	An integer specifying the thinning interval between consecutive draws. The default is <code>thin = 1</code> , implying that no draws are dopped. For <code>thin = 2</code> , every second draw is dropped and so on. Only used if <code>method = "bayesian"</code> .
HPDIprob	A numeric in the interval $(0, 1)$ specifying the target probability of the highest posterior density intervals. The default is <code>HPDIprob = 0.9</code> . Only used if <code>method = "bayesian"</code> .
pointEstimate	Posterior distribution's statistic of central tendency. Possible options are "mean" and "median". The default is <code>pointEstimate = "mean"</code> . Only used if <code>method = "bayesian"</code> .
MLEfit	(Optional) An object of class <code>TFPfit</code> which is used for initialization. Only used if <code>method = "bayesian"</code> .
...	additional arguments to be passed to the methods functions.

Details

The list object `prior` contains three list elements `cycle`, `trend`, and `cubs`. Each list element is a $4 \times n$ matrix where n denotes the number of parameters involved in the respective equation. The upper two elements specify the distribution, the lower two parameters specify box constraints. NA denotes no constraints. Autoregressive parameters are automatically restricted to the stationary region unless box constraints are specified. For instance, `prior$cycle[, 1]` contains the mean, standard deviation, lower and upper bound for the first variable, in that respective order.

The respective prior distributions are defined through their mean and standard deviation.

The Gibbs sampling procedure is as follows. For each $r = 1, \dots, R$

- The states are sampled by running the Kalman filter and smoother conditional on the parameters of the previous step, θ_{r-1}
- Trend equation parameters θ_{trend} : Conditional on the states α_r , a draw $\theta_{trend,k}$ is obtained either by a sequential Gibbs step, a Metropolis Hasting step, or by conjugacy, depending on the trend model specification.
- Cycle equation parameters θ_{cycle} : Conditional on the states α_r , a draw $\theta_{cycle,k}$ is obtained either by a sequential Gibbs step, a Metropolis Hasting step, or by conjugacy, depending on the cycle model specification.
- CUBS equation parameters θ_{cubs} : Conditional on the states α_r , a draw $\theta_{cubs,k}$ is obtained either by a sequential Gibbs step, a Metropolis Hasting step, a combination thereof, or by conjugacy, depending on the CUBS equation specification.

Value

For maximum likelihood estimation, an object of class `TFPfit` containing the following components:

<code>model</code>	The input object of class <code>TFPmodel</code> .
<code>SSMfit</code>	The estimation output from the function <code>fitSSM</code> from <code>KFAS</code> .
<code>SSMout</code>	The filtering and smoothing output from the function <code>KFS</code> from <code>KFAS</code> .

parameters	A data frame containing the estimated parameters, including standard errors, t-statistic, and p-values.
parRestr	A list of matrices containing the enforced parameter constraints.
fit	A list of model fit criteria (see below).
call	Original call to the function.

The list component `fit` contains the following model fit criteria:

loglik	Log-likelihood function values.
AIC	Akaike information criterion.
BIC	Bayesian information criterion.
AIC	Hannan-Quinn information criterion.
RMSE	root mean squared error of the CUBS equation.
R2	R squared of the CUBS equation.
LjungBox	Ljung-Box test output of the CUBS equation.
signal-to-noise	Signal-to-noise ratio.

For bayesian estimation, an object of class `TFPfit` containing the following components:

model	The input object of class <code>TFPmodel</code> .
ts1	A list of time series containing the estimated states.
parameters	A data frame containing the estimated parameters, including standard errors, highest posterior density credible sets.
prior	A list of matrices containing the used prior distributions.
fit	A list of model fit criteria (see below).
call	Original call to the function.

The list component `fit` contains the following model fit criteria:

R2	R squared of the CUBS equation.
signal-to-noise	Signal-to-noise ratio.

See Also

Other fitting methods: `fit.KuttnerModel()`, `fit.NAWRUmodel()`, `fit()`

Examples

```
# load data for Italy
data("gap")
country <- "Italy"
tsList <- amecoData2input(gap[[country]])
# define tfp model
model <- TFPmodel(ts1 = tsList, cycle = "RAR2")
# initialize parameter restrictions and estimate model
```

```

parRestr <- initializeRestr(model = model, type = "hp")

f <- fit(model = model, parRestr = parRestr)

# Bayesian estimation
prior <- initializePrior(model = model)

f <- fit(model = model, method = "bayesian", prior = prior, R = 5000, thin = 2)

```

gap

gap data set

Description

A dataset containing economic data on various countries from the 'AMECO' 2018 autumn vintage.

Usage

gap

Format

A list object with 53 country time series objects. Each time series object contains 14 variables:

- popw** Population: 15 to 64 years (unit: 1000 persons, code: NPAN)
- ur** Unemployment rate, total; Member States: definition EUROSTAT (unit: Percentage of civilian labor force, code: ZUTN)
- etd** Employment, persons: total economy (National accounts) (unit: 1000 persons, code: NETN)
- et** Employment, persons: all domestic industries (National accounts) (unit: 1000 persons, code: NETD)
- eet** Employees, persons: all domestic industries (National accounts) (unit: 1000 persons, code: NWTD)
- vaind** Gross value added at 2010 prices: manufacturing industry (unit: bn National currency, code: OVG5)
- vaserv** Gross value added at 2010 prices: services (unit: bn National currency, code: OVG4)
- vabuil** Gross value added at 2010 prices: building and construction (unit: bn National currency, code: OVG4)
- pconsp** Price deflator private final consumption expenditure (unit: National currency 2010 = 100, code: PCPH)
- cpih** Harmonised consumer price index (All-items, 2015 = 100, code: ZCPIH)
- cpin** National consumer price index (All-items, 2015 = 100, code: ZCPIN)
- ngdp** Gross domestic product at current prices (unit: bn National currency, code: UVGD)
- gdp** Gross domestic product at 2010 reference levels (unit: bn National currency, code: OVG5)

- gdpdefl** Price deflator gross domestic product (unit: National currency 2010 = 100, code: PVGD)
- ahours** Average annual hours worked per person employed (unit: Hours, code: NLHA)
- l** Total annual hours worked: total economy (unit: millions, code: NLHT)
- wtotal** Compensation of employees: total economy (unit: bn National currency, code: UWCD)
- nulc** Nominal unit labour costs: total economy (Ratio of compensation per employee to real GDP per person employed.) (unit: National currency 2010 = 100, code: PLCD)
- k** Net capital stock at 2010 prices: total economy (unit: bn National currency, code: OKND)
- serv** Confidence indicator in the service industry
- buil** Confidence indicator in the bulding and construction industry
- indu** Capacity utilization in manufacturing/industry

Source

https://economy-finance.ec.europa.eu/economic-research-and-databases/economic-databases/ameco-database_en

gapHP

HP-filter output gap

Description

Computes a HP filtered output gap.

Usage

```
gapHP(x, lambda = NULL, end = NULL, start = NULL)
```

Arguments

- | | |
|---------------------|--|
| <code>x</code> | A time series object containing gdp. |
| <code>lambda</code> | The smoothing parameter for the application of the HP filter. If not supplied, <code>lambda = 6.25</code> for yearly data, <code>lambda = 1600</code> for quarterly data, and <code>lambda = 129600</code> for monthly data. |
| <code>end</code> | (optional) A two element vector containing a year and a period specifying the end point for the filter application. |
| <code>start</code> | (optional) A two element vector containing a year and a period specifying the start point for the filter application. |

Value

Object of class `gap`, which is a list containing the two elements `potential` and `gap` and additionally the original time series.

gapProd	<i>Production function output gap</i>
---------	---------------------------------------

Description

Computes potential output and the output gap based on a production function methodology.

Usage

```
gapProd(
  tsl,
  NAWRUfit,
  TFPfit,
  alpha = 0.65,
  start = NULL,
  end = NULL,
  lambda = NULL
)
```

Arguments

<code>tsl</code>	A list of time series objects, see details.
<code>NAWRUfit</code>	An object of class <code>NAWRUfit</code> obtained via the function <code>fit</code> .
<code>TFPfit</code>	An object of class <code>TFPfit</code> obtained via the function <code>fit</code> .
<code>alpha</code>	A scalar between zero and one depicting the labor share. The default is <code>alpha = 0.65</code> .
<code>start</code>	(optional) A two element vector containing a year and a period specifying the start point for the estimation.
<code>end</code>	(optional) A two element vector containing a year and a period specifying the end point for the estimation.
<code>lambda</code>	The smoothing parameter for the application of the HP filter (see details). If not supplied, <code>lambda = 6.25</code> for yearly data, <code>lambda = 1600</code> for quarterly data, and <code>lambda = 129600</code> for monthly data.

Details

The list of time series `tsl` needs to have the following components:

lfnd Labor force non-domestic (unit: 1000 persons). (Set to zero if left unspecified).

parts Participation rate.

ahours Average hours worked (unit: hours).

gdp Gross domestic product at constant prices (unit: bn National currency, code: OVGD).

k Net capital stock at constant prices: total economy (unit: bn National currency, code: OKND).

popw Population: 15 to 64 years (unit: 1000 persons, code: NPAN).

The trend of the list components `parts`, `ahours` and `lfnd` (if available) is computed using the Hodrick-Prescott filter with the smoothing constant `lambda`, unless the supplied time series list `tsl` contains their trend (for instance, denoted by `partsTrend`).

Value

Object of class `gap`, which is a list with the following components:

<code>tsl</code>	List of time series including potential output <code>potential</code> , the output gap <code>gap</code> , all HP-filtered trend series, and all original series.
<code>NAWRUfit</code>	Provided <code>NAWRUfit</code> object.
<code>TFPfit</code>	Provided <code>TFPfit</code> object.
<code>call</code>	Original call to the function.

Examples

```
# compute the output gap given the previously obtained nawru and trend tfp
data("gap")
country <- "Belgium"
tsList <- amecoData2input(gap[[country]])
modelNAWRU <- NAWRUmodel(tsl = tsList)
modelTFP <- TFPmodel(tsl = tsList, cycle = "RAR2")

fittedNAWRU <- fit(model = modelNAWRU)
fittedTFP <- fit(model = modelTFP)

gapProd(tsl = tsList, NAWRUfit = fittedNAWRU, TFPfit = fittedTFP)
```

hpfilter

HP filter

Description

Applies the Hodrick Prescott Filter.

Usage

```
hpfilter(x, lambda)
```

Arguments

<code>x</code>	A univariate time series object.
<code>lambda</code>	The smoothing parameter.

Value

A univariate time series object containing the trend of the original time series.

Examples

```
# get data for France
data("gap")
country <- "France"
tsList <- amecoData2input(gap[[country]], alpha = 0.65)
hp <- hpfilter(x = tsList$gdp, lambda = 6.25)
```

indicator

Indicators fo CUBS

Description

A dataset containing the service sector confidence indicator, the construction sector confidence indicator and the capacity utilization in manufacturing/industry.

Usage

```
indicator
```

Format

A list with 53 nested country lists with time series objects. Each country list contains 3 time series variables:

serv Confidence indicator in the service industry.

buil Confidence indicator in the bulding and construction industry.

indu Capacity utilization in manufacturing/industry.

Details

A dataset containing the seasonally adjusted utilization indicators in the service industry, the building and construction industry, and capacity utilization in manufacturing/industry for all EU countries and some neighboring countries at different frequencies.

The confidence indicator in the service industry is composed of question 1, 2, and 3 of the monthly service sector survey $((Q1 + Q2 + Q3)/3)$. The underlying survey questions are as follows:

- Q1 Business situation development over the past 3 months
- Q2 Evolution of the demand over the past 3 months
- Q3 Expectation of the demand over the next 3 months

The confidence indicator in the building and construction industry is composed of question 3 and 4 of the monthly building and construction sector survey $((Q3 \text{ and } Q4)/2)$. The underlying survey questions are as follows:

- Q3 Evolution of your current overall order books
- Q4 Employment expectations over the next 3 months

The indicator for capacity utilization in manufacturing/industry is based on question 13 of the quarterly industry sector survey. The underlying survey question is as follows:

- Q3 Current level of capacity utilization

Source

[https://economy-finance.ec.europa.eu/economic-forecast-and-surveys_business-and-consumer-surveys_en](https://economy-finance.ec.europa.eu/economic-forecast-and-surveys/business-and-consumer-surveys_en)

initializeExo *Initialization of exogenous variables*

Description

Initializes the transformations applied to exogenous variables.

Usage

```
initializeExo(varNames, D = NULL, L = NULL)
```

Arguments

varNames A (k x 1) character vector containing the names of the exogenous variables.
D A (n x k) matrix containing the difference transformations, see details.
L A (n x k) matrix containing the lag transformations, see details.

Details

For the matrices D and L, the rows denote different transformations to each of the variables in the columns. NA indicates no transformation.

Value

An array of size (n, k, 2). The [, , 1] specifies the difference order and [, , 2] the lag order.

initializePrior *Initialization of prior distributions*

Description

Initializes the prior distributions for a model of class TFPmodel or NAWRModel.

Usage

```
initializePrior(model, MLE = !is.null(MLEfit), MLEfit = NULL)
```

Arguments

model	An object of class TFPmodel or NAWRModel.
MLE	(Optional) A logical indicating whether the MLE estimates should be used for the initialization. The default is MLE = FALSE if MLEfit is not provided and vice versa.
MLEfit	(Optional) An object of class TFPfit or NAWRfit which is used if MLE = TRUE.

Value

A list of three matrices with parameters for the prior distribution and box constraints. Each list item refers to an equation, namely the cycle, trend, and second observation equation. Each list element is a 4 x n matrix where n denotes the number of parameters involved in the respective equation. The upper two elements specify the distribution, the lower two parameters specify box constraints. NA denotes no constraints. Autoregressive parameters are automatically restricted to the stationary region unless box constraints are specified. The respective prior distributions are defined through their mean and standard deviation. For instance, prior\$cycle[, 1] contains the mean, standard deviation, lower and upper bound for the first variable, in that respective order.

initializeRestr	<i>Initialization of parameter restrictions</i>
-----------------	---

Description

Initializes parameter restrictions for objects of class NAWRModel, TFPmodel, or KuttnerModel.

Usage

```
initializeRestr(model, type = "basic", lambda = NULL, q = 0.01)
```

Arguments

model	An object of class NAWRModel, TFPmodel, or KuttnerModel.
type	The variance restriction type. Possible options are "basic", "hp", see details. The default is type = "basic".
lambda	The smoothing constant for the HP-filter if type = "hp".
q	Quantile for the Inverse Gamma distribution (only used if type = "hp"). The default is q = 0.01.

Details

For type = "hp", the HP filter is applied to the appropriately differences first observation series to obtain its trend and cycle. Subsequently, the specified trend and cycle models are fitted to obtain its innovation variance. Moreover, the second observation series (according to its specification) is fitted to obtain its innovation variance. Lastly, the obtained innovations variances are used to get lower and upper bounds. To that end, the q and 1-q quantiles of the inverse gamma distribution are used, with mean and standard deviation set to the estimated variances.

Value

A list of three matrices containing the parameter restrictions for the cycle, trend, and the second observation equation. Each matrix contains the lower and upper bound of the involved parameters. NA implies that no restriction is present.

is.gap	gap object check
--------	------------------

Description

Tests whether the input object is a valid object of class gap.

Usage

```
is.gap(object, return.logical = FALSE)
```

Arguments

object An object to be tested.
return.logical If return.logical = FALSE (default), an error message is printed if the object is not of class gap. If return.logical = TRUE, a logical value is returned.

Value

A logical value or nothing, depending on the value of return.logical.

is.KuttnerModel	KuttnerModel object check
-----------------	---------------------------

Description

Tests whether the input object is a valid object of class KuttnerModel.

Usage

```
is.KuttnerModel(object, return.logical = FALSE)
```

Arguments

object An object to be tested.
return.logical If return.logical = FALSE (default), an error message is printed if the object is not of class KuttnerModel. If return.logical = TRUE, a logical value is returned.

Value

A logical value or nothing, depending on the value of return.logical.

is.NAWRUmodel	NAWRUodel <i>object check</i>
---------------	-------------------------------

Description

Tests whether the input object is a valid object of class NAWRUmodel.

Usage

```
is.NAWRUmodel(object, return.logical = FALSE)
```

Arguments

object	An object to be tested.
return.logical	If return.logical = FALSE (default), an error message is printed if the object is not of class NAWRUmodel. If return.logical = TRUE, a logical value is returned.

Value

A logical value or nothing, depending on the value of return.logical.

Examples

```
# load data for France
data("gap")
tsList <- amecoData2input(gap$France, alpha = 0.65)

# Traditional phillips curve
model <- NAWRUmodel(tsl = tsList, trend = "RW2", cycle = "AR2", type = "NKP", cycleLag = 0:1)
is.NAWRUmodel(model, return.logical = TRUE)
attr(model, "phillips curve")$cycleLag <- 0
is.NAWRUmodel(model, return.logical = TRUE)
```

is.TFPmodel	TFPmodel <i>object check</i>
-------------	------------------------------

Description

Tests whether the input object is a valid object of class TFPmodel.

Usage

```
is.TFPmodel(object, return.logical = FALSE)
```

Arguments

`object` An object to be tested.

`return.logical` If `return.logical = FALSE` (default), an error message is printed if the object is not of class `TFPmodel`. If `return.logical = TRUE`, a logical value is returned.

Value

A logical value or nothing, depending on the value of `return.logical`.

Examples

```
# load data for Germany
data("gap")
data("indicator")
country <- "Germany"
tsList <- amecoData2input(gap[[country]], alpha = 0.65)

# compute cubs indicator
namesCubs <- c("indu", "serv", "buil")
namesVACubs <- paste0("va", namesCubs)
tscubs <- cubs(
  tsCU = gap[[country]][, namesCubs],
  tsVA = gap[[country]][, namesVACubs]
)
tsList <- c(tsList, tscubs)

# define tfp model
model <- TFPmodel(
  tsl = tsList, trend = "DT", cycle = "RAR2",
  cycleLag = 2, cubsErrorARMA = c(1, 0)
)
is.TFPmodel(model, return.logical = TRUE)
attr(model, "cubs")$cycleLag <- 1
is.TFPmodel(model, return.logical = TRUE)
```

KuttnerModel

Kuttner model

Description

Creates a state space object of class `KuttnerModel` which can be fitted using `fit`.

Usage

```
KuttnerModel(
  tsl,
  cycle = "AR2",
  cycleLag = 1,
  trend = "RW1",
```

```

inflErrorARMA = c(0, 3),
start = NULL,
end = NULL,
anchor = NULL,
anchor.h = NULL
)

```

Arguments

<code>ts1</code>	A list of time series objects, see details.
<code>cycle</code>	A character string specifying the cycle model. <code>cycle = "AR1"</code> denotes an AR(1) process, <code>cycle = "AR2"</code> an AR(2) process. The default is <code>cycle = "AR2"</code> .
<code>cycleLag</code>	A non-negative integer specifying the maximum cycle lag that is included in the inflation equation. The default is <code>cycleLag = 0</code> , see details.
<code>trend</code>	A character string specifying the trend model. <code>trend = "RW1"</code> denotes a first order random walk, <code>trend = "RW2"</code> a second order random walk (local linear trend) and <code>trend = "DT"</code> a damped trend model. The default is <code>trend = "RW1"</code> .
<code>inflErrorARMA</code>	A 2 x 1 vector with non-negative integers specifying the AR and MA degree of the error term in the inflation equation. The default is <code>inflErrorARMA = c(0, 3)</code> , see details.
<code>start</code>	(Optional) Start vector for the estimation, e.g. <code>c(1980, 1)</code> .
<code>end</code>	(Optional) End vector for the estimation, e.g. <code>c(2020, 1)</code> .
<code>anchor</code>	(Optional) Anchor value for the logarithm of trend <code>gdp</code> .
<code>anchor.h</code>	(Optional) Anchor horizon in the frequency of the given time series.

Details

The list of time series `ts1` needs to have the following components:

gdp Real gross domestic product.

infl Inflation.

A `cycleLag` equal to 0 implies that only the contemporaneous cycle is included in the inflation equation. A `cycleLag` equal to 0:1 implies that the contemporaneous as well as the lagged cycle are included.

A `inflErrorARMA` equal to `c(0, 0)` implies that the error term in the inflation equation is white noise. `inflErrorARMA = c(1, 0)` implies that the error is an AR(1) process and for `inflErrorARMA = c(1, 2)` the error follows an ARMA(1, 2) process.

Value

Object of class `KuttnerModel`, which is a list with the following components:

<code>ts1</code>	A list of used time series.
<code>SSModel</code>	An object of class <code>SSModel</code> specifying the state-space model.
<code>loc</code>	A data frame containing information on each involved parameter, for instance its corresponding system matrix, variable names, and parameter restrictions.

call Original call to the function.

In addition, the object contains the following attributes:

cycle Cycle specification.

trend Trend specification.

inflation equation

A list containing the components cycleLag, errorARMA, exoVariables.

anchor A list containing the components value, horizon.

period A list containing the components start, end, frequency.

Examples

```
# load data for the Netherlands
data("gap")
country <- "Netherlands"
tsList <- as.list(gap[[country]][, c("cpih", "gdp")])
tsList$infl <- diff(tsList$cpih)
model <- KuttnerModel(tsl = tsList, trend = "RW2", start = 1980)
```

NAWRUmodel

NAWRU model

Description

Creates a state space object of class NAWRUmodel which can be fitted using `fit`.

Usage

```
NAWRUmodel(
  tsl,
  trend = "RW2",
  cycle = "AR2",
  type = "TKP",
  cycleLag = 0,
  pcErrorARMA = c(0, 0),
  exoType = NULL,
  start = NULL,
  end = NULL,
  anchor = NULL,
  anchor.h = NULL
)
```

Arguments

<code>ts1</code>	A list of time series objects, see details.
<code>trend</code>	A character string specifying the trend model. <code>trend = "RW1"</code> denotes a first order random walk, <code>trend = "RW2"</code> a second order random walk (local linear trend) and <code>trend = "DT"</code> a damped trend model. The default is <code>trend = "RW2"</code> .
<code>cycle</code>	A character string specifying the cycle model. <code>cycle = "AR1"</code> denotes an AR(1) process, <code>cycle = "AR2"</code> an AR(2) process. The default is <code>cycle = "AR2"</code> .
<code>type</code>	A character string specifying the type of the Phillip's curve. <code>type = "TKP"</code> denotes the traditional Keynesian Phillip's curve and <code>type = "NKP"</code> the New Keynesian Phillip's curve, see details. The default is <code>type = "TKP"</code> .
<code>cycleLag</code>	A vector specifying the cycle lags that are included in the Phillip's curve. The default is <code>cycleLag = 0</code> , see details.
<code>pcErrorARMA</code>	A 2×1 vector with non-negative integers specifying the AR and MA degree of the error term in the Phillip's curve equation. The default is <code>pcErrorARMA = c(0, 0)</code> , see details.
<code>exoType</code>	An optional $n \times m \times 2$ array specifying the possible difference and lag transformation for the variables. <code>exoType</code> can be initialized using the function <code>initializeExo</code> . The column names give the variable names. <code>exoType[, , 1]</code> contains the difference transformations and <code>exoType[, , 2]</code> the subsequent lag transformations, see details.
<code>start</code>	(Optional) Start vector for the estimation, e.g. <code>c(1980, 1)</code> .
<code>end</code>	(Optional) End vector for the estimation, e.g. <code>c(2020, 1)</code> .
<code>anchor</code>	(Optional) Anchor value for the unemployment rate.
<code>anchor.h</code>	(Optional) Anchor horizon in the frequency of the given time series.

Details

The list of time series `ts1` needs to have the following components:

ur Unemployment rate.

nulc Nominal Unit labor costs, if `type = "TKP"`.

rulc Real unit labor costs, if `type = "NKP"`.

and optionally other variables included in `exoType`.

A `cycleLag` equal to 0 implies that only the contemporaneous cycle is included in the Phillip's curve. A `cycleLag` equal to 0:1 implies that the contemporaneous as well as the lagged cycle are included.

A `pcErrorARMA` equal to `c(0, 0)` implies that the error term in the Phillip's curve is white noise. `pcErrorARMA = c(1, 0)` implies that the error is an AR(1) process and for `pcErrorARMA = c(1, 2)` the error follows an ARMA(1, 2) process.

For the New Keynesian Phillip's curve, the `cycleLag` cannot be chosen. `cycleLag` will be set to 0 if `cycle = "AR1"` and to 1 if `cycle = "AR2"`. In the latter case, the forward solution of the Phillip's curve implies parameter restrictions for the lagged cycle on the Phillip's curve. Moreover, exogenous variables will be ignored in the case of the New Keynesian Phillip's curve.

The array `exoType` consists of non-negative integers or NAs. `exoType[, , 1] = c(NA, 1)` and `exoType[, , 2] = c(NA, 2)` implies that the first variable is not included in the Phillip's curve whereas the second lag of the first difference of the second variable is included.

Value

Object of class `NAWRUmodel`, which is a list with the following components:

<code>ts1</code>	A list of used time series.
<code>SSModel</code>	An object of class <code>SSModel</code> specifying the state-space model.
<code>loc</code>	A data frame containing information on each involved parameter, for instance its corresponding system matrix, variable names, and parameter restrictions.
<code>call</code>	Original call to the function.

In addition, the object contains the following attributes:

<code>cycle</code>	Cycle specification.
<code>trend</code>	Trend specification.
<code>phillipsCurve</code>	A list containing the components <code>type</code> , <code>cycleLag</code> , <code>errorARMA</code> , <code>exoVariables</code> .
<code>anchor</code>	A list containing the components <code>value</code> , <code>horizon</code> .
<code>period</code>	A list containing the components <code>start</code> , <code>end</code> , <code>frequency</code> .

Examples

```
# load data for France
data("gap")
tsList <- amecoData2input(gap$France, alpha = 0.65)
# Traditional phillips curve
model <- NAWRUmodel(ts1 = tsList, trend = "RW2", cycle = "AR2", type = "TKP", cycleLag = 0)

# New-Keynesian Phillips curve
model <- NAWRUmodel(ts1 = tsList, trend = "RW2", cycle = "AR2", type = "NKP", cycleLag = 0:1)

# Traditional Phillips curve with 6 exogenous variables
# specify exogenous variable transformations
D <- matrix(c(2, 2, 2, 1, 1, 1), 2, 3, byrow = TRUE)
L <- matrix(c(0, 0, 0, 1, 1, 1), 2, 3, byrow = TRUE)
exoType <- initializeExo(varNames = c("tot", "prod", "ws"), D = D, L = L)
model <- NAWRUmodel(ts1 = tsList, cycleLag = 0:1, exoType = exoType)
```

plot.gap

Plots for a gap object

Description

Plots potential output growth and the output gap based on an objects of class `gap`.

Usage

```
## S3 method for class 'gap'
plot(
  x,
  contribution = FALSE,
  path = NULL,
  combine = TRUE,
  prefix = NULL,
  device = "png",
  width = 10,
  height = 3,
  ...
)
```

Arguments

x	An object of class gap.
contribution	A boolean indicating whether the contributions to potential output growth and the output gap should be plotted (only applicable for production function type output gaps).
path	An optional file path. If specified, the plots will be saved using the format in device under the given path.
combine	A logical indicating whether the plots should be combined or not, the default is TRUE.
prefix	An optional character string to be added to the names of the plots in case path is specified.
device	Device passed on to ggplot for plot saving. Options are "eps", "ps", "tex" (pictex), "pdf", "jpeg", "tiff", "png", "bmp", "svg" or "wmf".
width	The plot width in case of printing.
height	The plot height in case of printing.
...	Ignored.

Value

No return value, plots are printed.

plot.KuttnerFit *Plots for a KuttnerFit object*

Description

Plots potential growth and the output gap and gives diagnostic plots based on standardized residuals for objects of class KuttnerFit.

Usage

```
## S3 method for class 'KuttnerFit'
plot(
  x,
  alpha = 0.05,
  bounds = TRUE,
  path = NULL,
  combine = TRUE,
  prefix = NULL,
  device = "png",
  width = 10,
  height = 3,
  ...
)
```

Arguments

x	An object of class KuttnerFit.
alpha	The significance level for the trend (alpha in $[0, 1]$). Only used if bounds = TRUE.
bounds	A logical indicating whether significance intervals should be plotted around gdp. The default is bounds = TRUE.
path	An optional file path. If specified, the plots will be saved using the format in device under the given path.
combine	A logical indicating whether the diagnostic plots should be combined or not, the default is TRUE.
prefix	An optional character string to be added to the names of the plots in case path is specified.
device	Device passed on to ggplot for plot saving. Options are "eps", "ps", "tex" (pictex), "pdf", "jpeg", "tiff", "png", "bmp", "svg" or "wmf".
width	The plot width in case of printing.
height	The plot height in case of printing.
...	Ignored.

Value

No return value, plots are printed.

plot.NAWRUfit *Plots for a NAWRUfit object*

Description

Plots the NAWRU and the Phillip's curve and gives diagnostic plots based on standardized residuals for objects of class NAWRUfit.

Usage

```
## S3 method for class 'NAWRUfit'
plot(
  x,
  alpha = 0.05,
  bounds = TRUE,
  path = NULL,
  combine = TRUE,
  prefix = NULL,
  posterior = FALSE,
  device = "png",
  width = 10,
  height = 3,
  ...
)
```

Arguments

x	An object of class NAWRUfit.
alpha	The significance level for the NAWRU (alpha in $[\theta, 1]$). Only used if bounds = TRUE.
bounds	A logical indicating whether significance intervals should be plotted around the nawru. The default is bounds = TRUE.
path	An optional file path. If specified, the plots will be saved using the format in device under the given path.
combine	A logical indicating whether the diagnostic plots should be combined or not, the default is TRUE.
prefix	An optional character string to be added to the names of the plots in case path is specified.
posterior	A logical indicating whether posterior diagnostics should be plotted. The default is FALSE. Only applied in the case of bayesian estimation.
device	Device passed on to ggplot for plot saving. Options are "eps", "ps", "tex" (pictex), "pdf", "jpeg", "tiff", "png", "bmp", "svg" or "wmf".
width	The plot width in case of printing.
height	The plot height in case of printing.
...	Ignored.

Value

No return value, plots are printed.

plot.TFPfit	<i>Plots for a TFPfit object</i>
-------------	----------------------------------

Description

Plots the TFP trend and the CUBS equation and gives diagnostic plots based on standardized residuals for objects of class TFPfit.

Usage

```
## S3 method for class 'TFPfit'
plot(
  x,
  alpha = 0.05,
  bounds = TRUE,
  path = NULL,
  combine = TRUE,
  prefix = NULL,
  posterior = FALSE,
  device = "png",
  width = 10,
  height = 3,
  ...
)
```

Arguments

x	An object of class TFPfit.
alpha	The significance level for the TFP trend (alpha in $[\theta, 1]$). Only used if bounds = TRUE.
bounds	A logical indicating whether significance intervals should be plotted around tfp growth. The default is bounds = TRUE.
path	An optional file path. If specified, the plots will be saved using the format in device under the given path.
combine	A logical indicating whether the diagnostic plots should be combined or not, the default is TRUE.
prefix	An optional character string to be added to the names of the plots in case path is specified.
posterior	A logical indicating whether posterior diagnostics should be plotted. The default is FALSE. Only applied in the case of bayesian estimation.
device	Device passed on to ggplot for plot saving. Options are "eps", "ps", "tex" (pictex), "pdf", "jpeg", "tiff", "png", "bmp", "svg" or "wmf".

width	The plot width in case of printing.
height	The plot height in case of printing.
...	Ignored.

Value

No return value, plots are printed.

predict.fit	<i>Predictions</i>
-------------	--------------------

Description

Computes predictions for an object of class NAWRUfit, TFPfit, or KuttnerFit estimated via MLE or Bayesian methods (objects of class fit).

Usage

```
## S3 method for class 'fit'
predict(object, n.ahead = 10, exogenous = "mean", returnFit = TRUE, ...)
```

Arguments

object	An object of class NAWRUfit, TFPfit, or KuttnerFit (objects of class fit).
n.ahead	An integer specifying the prediction horizon.
exogenous	A character string specifying the computation of exogenous variables included in the model (if applicable). Valid options are exogenous = "mean" and exogenous = "last".
returnFit	A logical. If TRUE, an object of the same class as fit where the list entry tsl is replaced. If FALSE, only the new time series list is returned.
...	Ignored.

Value

The fitted object with an updated time series list tsl. If returnFit = FALSE, only the updated time series list is returned.

<code>print.gap</code>	<i>Print gap object</i>
------------------------	-------------------------

Description

Prints the model specifications of an object of class `gap`.

Usage

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

Arguments

<code>x</code>	An object of class <code>gap</code> .
<code>...</code>	Ignored.

Value

No return value, results are printed.

<code>print.KuttnerFit</code>	<i>Print KuttnerFit object</i>
-------------------------------	--------------------------------

Description

Prints the model specifications and the estimation results of an object of class `KuttnerFit`.

Usage

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

Arguments

<code>x</code>	An object of class <code>KuttnerFit</code> .
<code>...</code>	Ignored.

Value

No return value, results are printed.

`print.KuttnerModel` *Print KuttnerModel object*

Description

Prints the model specifications of an object of class `KuttnerModel`.

Usage

```
## S3 method for class 'KuttnerModel'  
print(x, call = TRUE, check = TRUE, ...)
```

Arguments

<code>x</code>	An object of class <code>KuttnerModel</code> .
<code>call</code>	A logical. If <code>TRUE</code> , the call will be printed.
<code>check</code>	A logical. If <code>TRUE</code> , the model class will be checked.
<code>...</code>	Ignored.

Value

No return value, model information is printed.

`print.NAWRUFit` *Print NAWRUFit object*

Description

Prints the model specifications and the estimation results of an object of class `NAWRUFit`.

Usage

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

Arguments

<code>x</code>	An object of class <code>NAWRUFit</code> .
<code>...</code>	Ignored.

Value

No return value, results are printed.

print.NAWRUmodel *Print NAWRUmodel object*

Description

Prints the model specifications of an object of class NAWRUmodel.

Usage

```
## S3 method for class 'NAWRUmodel'  
print(x, call = TRUE, check = TRUE, ...)
```

Arguments

x	An object of class NAWRUmodel.
call	A logical. If TRUE, the call will be printed.
check	A logical. If TRUE, the model class will be checked.
...	Ignored.

Value

No return value, model information is printed.

print.TFPfit *Print TFPfit object*

Description

Prints the model specifications and the estimation results of an object of class TFPfit.

Usage

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

Arguments

x	An object of class TFPfit.
...	Ignored.

Value

No return value, results are printed.

print.TFPmodel	<i>Print TFPmodel object</i>
----------------	------------------------------

Description

Prints the model specifications of an object of class TFPmodel.

Usage

```
## S3 method for class 'TFPmodel'
print(x, call = TRUE, check = TRUE, ...)
```

Arguments

x	An object of class TFPmodel.
call	A logical. If TRUE, the call will be printed.
check	A logical. If TRUE, the model class will be checked.
...	Ignored.

Value

No return value, model information is printed.

TFPmodel	<i>TFP trend model</i>
----------	------------------------

Description

Creates a state space object of class TFPmodel which can be fitted using fit.

Usage

```
TFPmodel(
  tsl,
  trend = "DT",
  cycle = "AR2",
  cycleLag = 0,
  cubsAR = 0,
  cubsErrorARMA = c(0, 0),
  start = NULL,
  end = NULL,
  anchor = NULL,
  anchor.h = NULL
)
```

Arguments

<code>ts1</code>	A list of time series objects, see details.
<code>trend</code>	A character string specifying the trend model. <code>trend = "RW1"</code> denotes a first order random walk, <code>trend = "RW2"</code> a second order random walk (local linear trend) and <code>trend = "DT"</code> a damped trend model. The default is <code>trend = "DT"</code> .
<code>cycle</code>	A character string specifying the cycle model. <code>cycle = "AR1"</code> denotes an AR(1) process, <code>cycle = "AR2"</code> an AR(2) process, <code>cycle = "RAR2"</code> a reparametrized AR(2) process. The default is <code>cycle = "AR2"</code> .
<code>cycleLag</code>	A non-negative integer specifying the maximum cycle lag that is included in the CUBD equation. The default is <code>cycleLag = 0</code> , see details.
<code>cubsAR</code>	A non-negative integer specifying the maximum CUBS lag that is included in the CUBS equation. The default is <code>cubsAR = 0</code> , see details.
<code>cubsErrorARMA</code>	A vector with non-negative integers specifying the AR and MA degree of the error term in the CUBS equation. The default is <code>cubsErrorARMA = c(0, 0)</code> , see details.
<code>start</code>	(Optional) Start vector for the estimation, e.g. <code>c(1980, 1)</code> .
<code>end</code>	(Optional) End vector for the estimation, e.g. <code>c(2020, 1)</code> .
<code>anchor</code>	(Optional) Sanchor value for the log of the TFP trend.
<code>anchor.h</code>	(Optional) Anchor horizon in the frequency of the given time series.

Details

The list of time series `ts1` needs to have the following components:

tfp Total factor productivity.

cubs Capacity utilization economic sentiment indicator.

A `cycleLag` equal to 0 implies that only the contemporaneous cycle is included in the CUBS equation. A `cycleLag` equal to 0:1 implies that the contemporaneous as well as the lagged cycle are included.

A `cubsAR` equal to 0 implies that no autoregressive term is included in the CUBS equation. `cubsAR = 1` implies that a lagged term is included, `cubsAR = 2` implies that a two lags are included, and so on.

A `cubsErrorARMA` equal to `c(0, 0)` implies that the error term in the CUBS equation is white noise. `cubsErrorARMA = c(1, 0)` implies that the error is an AR(1) process and for `cubsErrorARMA = c(1, 2)` the error follows an ARMA(1, 2) process.

Value

Object of class `TFPmodel`, which is a list with the following components:

<code>ts1</code>	A list of used time series.
<code>SSModel</code>	An object of class <code>SSModel</code> specifying the state-space model.
<code>loc</code>	A data frame containing information on each involved parameter, for instance its corresponding system matrix, variable names, and parameter restrictions.

call	Original call to the function.
In addition, the object contains the following attributes:	
cycle	Cycle specification.
trend	Trend specification.
cubs	A list containing the components cycleLag, cubsAR, errorARMA, exoVariables.
anchor	A list containing the components value, horizon.
period	A list containing the components start, end, frequency.

Examples

```
# load data for Germany
data("gap")
data("indicator")
country <- "Germany"
tsList <- amecoData2input(gap[[country]], alpha = 0.65)

# compute cubs indicator
namesCubs <- c("indu", "serv", "buil")
namesVACubs <- paste0("va", namesCubs)
tscubs <- cubs(
  tsCU = gap[[country]][, namesCubs],
  tsVA = gap[[country]][, namesVACubs]
)
tsList <- c(tsList, tscubs)

# define tfp model
model <- TFPmodel(tsl = tsList, cycle = "RAR2", cubsErrorARMA = c(1,0))
```

trendAnchor	<i>Trend anchor</i>
-------------	---------------------

Description

Computes the anchored trend given a fitted object of class NAWRUfit, TFPfit, or KuttnerFit.

Usage

```
trendAnchor(fit, anchor = NULL, h = NULL, returnFit = FALSE)
```

Arguments

fit	An object of class NAWRUfit, TFPfit, or KuttnerFit.
anchor	A numeric specifying the anchor value. If unspecified, anchor is taken from the object fit (if specified).
h	An integer specifying the anchor horizon in the frequency of the underlying model. If unspecified, h is taken from the object fit (if specified).
returnFit	A logical. If TRUE, an object of the same class as fit including the anchored trend is returned. If FALSE, only the anchored trend time series is returned.

Value

A fitted object if `returnFit = TRUE` or a time series with the anchored trend.

Examples

```
# define nawru model for France
data("gap")
tsList <- amecoData2input(gap$France)
model <- NAWRUmodel(ts1 = tsList)

# estimate nawru model

f <- fit(model = model)

# compute anchored nawru
anchoredNawru <- trendAnchor(fit = f, anchor = 6.5, h = 10)
```

Index

* datasets

gap, 19
indicator, 23

* fitting methods

fit, 10
fit.KuttnerModel, 11
fit.NAWRUmodel, 12
fit.TFPmodel, 16

amecoData2input, 3
autoGapProd, 4

cubs, 7

fetchAmecoData, 8
fit, 10, 12, 15, 18
fit.KuttnerModel, 11, 11, 15, 18
fit.NAWRUmodel, 11, 12, 12, 18
fit.TFPmodel, 11, 12, 15, 16

gap, 19
gapHP, 20
gapProd, 21

hpfilter, 22

indicator, 23
initializeExo, 24
initializePrior, 24
initializeRestr, 25
is.gap, 26
is.KuttnerModel, 26
is.NAWRUmodel, 27
is.TFPmodel, 27

KuttnerModel, 28

NAWRUmodel, 30

plot.gap, 32
plot.KuttnerFit, 33

plot.NAWRUfit, 35
plot.TFPfit, 36
predict.fit, 37
print.gap, 38
print.KuttnerFit, 38
print.KuttnerModel, 39
print.NAWRUfit, 39
print.NAWRUmodel, 40
print.TFPfit, 40
print.TFPmodel, 41

TFPmodel, 41
trendAnchor, 43