

# Package ‘UComp’

May 7, 2026

**Type** Package

**Title** Automatic Univariate Time Series Modelling of many Kinds

**Version** 5.3.1

**Date** 2026-05-06

**Author** Diego J. Pedregal [aut, cre] (ORCID:  
<<https://orcid.org/0000-0003-4958-0969>>)

**Maintainer** Diego J. Pedregal <diego.pedregal@uclm.es>

**Description** Comprehensive analysis and forecasting  
of univariate time series using automatic  
time series models of many kinds.  
Harvey AC (1989) <[doi:10.1017/CBO9781107049994](https://doi.org/10.1017/CBO9781107049994)>.  
Pedregal DJ and Young PC (2002) <[doi:10.1002/9780470996430](https://doi.org/10.1002/9780470996430)>.  
Durbin J and Koopman SJ (2012) <[doi:10.1093/acprof:oso/9780199641178.001.0001](https://doi.org/10.1093/acprof:oso/9780199641178.001.0001)>.  
Hyndman RJ, Koehler AB, Ord JK, and Snyder RD (2008) <[doi:10.1007/978-3-540-71918-2](https://doi.org/10.1007/978-3-540-71918-2)>.  
Gómez V, Maravall A (2000) <[doi:10.1002/9781118032978](https://doi.org/10.1002/9781118032978)>.  
Pedregal DJ, Trapero JR and Holgado E (2024) <[doi:10.1016/j.ijforecast.2023.09.004](https://doi.org/10.1016/j.ijforecast.2023.09.004)>.

**License** GPL-3

**Encoding** UTF-8

**Imports** ggplot2, gridExtra, tsibble, tsoutliers, stats, ggforce,  
utils, parallel

**LinkingTo** Rcpp, RcppArmadillo

**Depends** Rcpp (>= 1.0.3), R (>= 3.5.0)

**LazyData** true

**Suggests** knitr, rmarkdown

**RoxygenNote** 7.3.3

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2026-05-04 08:20:02 UTC

## Contents

Accuracy	3
acft	4
AIC.UComp	5
airpas	6
ARIMA	6
ARIMAestim	8
ARIMAforecast	9
ARIMAsetup	11
ARIMAvalidate	13
arma2tsi	13
armaFilter	14
auxInvBoxCox	15
BIC.UComp	15
box.cox	16
ch4	17
colMedians	17
conv	18
cusum	19
DAYforecast	20
dif	21
ETS	22
ETScomponents	24
ETSestim	25
ETSforecast	26
ETSsetup	28
ETSvalidate	30
extract	31
gaussTest	32
gdp	32
getp0	33
ident	34
inv.box.cox	35
invBoxCox	35
ipi	36
OECDgdp	37
plotAcfPacf	37
plotBar	38
plotSlide	39
plus_one	40
predict.UComp	40
removeNaNs	41
roots	42
rowMedians	42
sales	43
size	44
slide	44

slideAux . . . . .	46
sumStats . . . . .	47
tests . . . . .	48
TETS . . . . .	49
TETScomponents . . . . .	51
TETSestim . . . . .	52
TETSforecast . . . . .	53
TETSsetup . . . . .	55
TETSvalidate . . . . .	57
tsDisplay . . . . .	58
UC . . . . .	59
UCcommand . . . . .	62
UCcomponents . . . . .	63
UCdisturb . . . . .	64
UCestim . . . . .	65
UCfilter . . . . .	66
UCforecast . . . . .	67
UChp . . . . .	70
UComp . . . . .	71
UCsetup . . . . .	72
UCsmooth . . . . .	75
UCvalidate . . . . .	76
USgdp . . . . .	77
varTest . . . . .	77
WEEKforecast . . . . .	78
zplane . . . . .	79

**Index** **81**

---

Accuracy	<i>Accuracy</i>
----------	-----------------

---

**Description**

Accuracy for 1 time series  $y$  and several forecasting methods  $py$  and  $h$  steps ahead  $py$  is  $h \times nMethods \times nSeries$

**Usage**

Accuracy( $py$ ,  $y$ ,  $s = \text{frequency}(y)$ ,  $\text{collectFun} = \text{mean}$ )

**Arguments**

$py$	matrix of forecasts ( $h \times nMethods \times nForecasts$ )
$y$	a matrix of actual values ( $n \times nForecasts$ )
$s$	seasonal period, number of observations per year
$\text{collectFun}$	aggregation function (mean, median, etc.)

**Value**

Table of accuracy results

**Author(s)**

Diego J. Pedregal

**See Also**

[colMedians](#), [rowMedians](#), [tests](#), [sumStats](#), [gaussTest](#), [ident](#), [cusum](#), [varTest](#), [conv](#), [armaFilter](#), [dif](#), [roots](#), [zplane](#), [acft](#), [slide](#), [plotSlide](#), [tsDisplay](#), [size](#)

**Examples**

```
## Not run: Accuracy(py, y, 12)
```

---

acft

*acft*

---

**Description**

Theoretical autocorrelation functions of ARMA models

**Usage**

```
acft(MApolynomial = 1, ARpolynomial = 1, ncoef = 38, s = 1)
```

**Arguments**

MApolynomial	coefficients of numerator polynomial in descending order
ARpolynomial	coefficients of denominator polynomial in descending order
ncoef	number of coefficients
s	seasonal period, number of observations per year

**Value**

Theoretical autocorrelation functions

**Author(s)**

Diego J. Pedregal

**See Also**

[colMedians](#), [rowMedians](#), [tests](#), [sumStats](#), [gaussTest](#), [ident](#), [cusum](#), [varTest](#), [conv](#), [armaFilter](#), [dif](#), [roots](#), [zplane](#), [slide](#), [plotSlide](#), [Accuracy](#), [tsDisplay](#), [size](#)

**Examples**

```
acft(c(1, -0.8), c(1, 0.8))
```

---

AIC.UComp

*AIC.UComp*


---

**Description**

Extract AIC value of UComp object

**Usage**

```
## S3 method for class 'UComp'
AIC(object, ..., k = 2)
```

**Arguments**

object	Object of class “UComp”.
...	Additional inputs to function.
k	The penalty per parameter to be used.

**Details**

Selection criteria for models with different number of parameters, the smaller AIC the better. The formula used here is  $AIC = -2(\ln(L) - k)/n$ , where  $\ln(L)$  is the log-likelihood at the optimum,  $k$  is the number of parameters plus non-stationary states and  $n$  is the number of observations. Mind that this formulation differs from the usual definition that does not divide by  $n$ . This makes that  $AIC(m)$  and  $AIC(\logLik(m))$  give different results, being  $m$  an UComp object.

**Value**

AIC value of a UC model

**Author(s)**

Diego J. Pedregal

**See Also**

[UC](#), [UCforecast](#), [UCvalidate](#), [UCfilter](#), [UCsmooth](#), [UCdisturb](#), [UCcomponents](#)

**Examples**

```
y <- log(AirPassengers)
m1 <- UCforecast(y, model = "11t/equal/arma(0,0)")
AIC(m1)
```

---

airpas	<i>Airpassengers in Spain</i>
--------	-------------------------------

---

**Description**

Foreign arrivals by air in Spain in thousands of passengers (airpas).

**Usage**

```
airpas
```

**Format**

Time series objects.

Monthly data from 1969

<https://portal.mineco.gob.es/es-es/economiayempresa/EconomiaInformesMacro/Paginas/bdsice.aspx>

**Value**

No return value, called for side effects

**Examples**

```
airpas
```

---

ARIMA	<i>ARIMA</i>
-------	--------------

---

**Description**

Runs all relevant functions for ARIMA modelling

**Usage**

```
ARIMA(  
  y,  
  u = NULL,  
  model = NULL,  
  cnst = NULL,  
  s = frequency(y),  
  criterion = "bic",  
  h = 2 * s,  
  verbose = FALSE,  
  lambda = 1,  
  maxOrders = c(3, 2, 3, 2, 1, 2),
```

```

bootstrap = FALSE,
nSimul = 5000,
fast = FALSE
)

```

### Arguments

y	a time series to forecast (it may be either a numerical vector or a time series object). This is the only input required. If a vector, the additional input s should be supplied compulsorily (see below).
u	a matrix of input time series. If the output wanted to be forecast, matrix u should contain future values for inputs.
model	the model to estimate. A vector c(p,d,q,P,D,Q) containing the model orders of an ARIMA(p,d,q)x(P,D,Q)_s model. A constant may be estimated with the cnst input. Use a NULL to automatically identify the ARIMA model.
cnst	flag to include a constant in the model (TRUE/FALSE/NULL). Use NULL to estimate
s	seasonal period of time series (1 for annual, 4 for quarterly, ...)
criterion	information criterion for identification stage ("aic", "bic", "aicc")
h	forecast horizon. If the model includes inputs h is not used, the length of u is used instead.
verbose	intermediate estimation output (TRUE / FALSE)
lambda	Box-Cox lambda parameter (NULL: estimate)
maxOrders	a vector c(p,d,q,P,D,Q) containing the maximum orders of model orders to search for in the automatic identification
bootstrap	use bootstrap simulation for predictive distributions
nSimul	number of simulation runs for bootstrap simulation of predictive distributions
fast	fast identification (avoids post-identification checks)

### Details

See help of ARIMAforecast.

### Value

An object of class ARIMA. It is a list with fields including all the inputs and the fields listed below as outputs. All the functions in this package fill in part of the fields of any ARIMA object as specified in what follows (function ARIMA fills in all of them at once):

After running ARIMAforecast or ARIMA:

p	Estimated parameters
yFor	Forecasted values of output
yForV	Variance of forecasted values of output
ySimul	Bootstrap simulations for forecasting distribution evaluation

After running ARIMAvalidate:

table	Estimation and validation table
-------	---------------------------------

**Author(s)**

Diego J. Pedregal

**See Also**

[ARIMAforecast](#), [ARIMAvalidate](#),

**Examples**

```
y <- log(AirPassengers)
m1 <- ARIMA(y)
m1 <- ARIMA(y, lambda = NULL)
```

---

ARIMAestim

*ARIMAestim*

---

**Description**

Estimates and forecasts ARIMA models

**Usage**

```
ARIMAestim(m)
```

**Arguments**

`m` an object of type ARIMA created with `ARIMAforecast`

**Details**

`ARIMAestim` estimates and forecasts a time series using an ARIMA model

**Value**

The same input object with the appropriate fields filled in, in particular:

<code>p</code>	Estimated parameters
<code>yFor</code>	Forecasted values of output
<code>yForV</code>	Variance of forecasted values of output
<code>ySimul</code>	Bootstrap simulations for forecasting distribution evaluation

ARIMAforecast

*ARIMAforecast***Description**

Estimates and forecasts ARIMA general univariate models

**Usage**

```
ARIMAforecast(
  y,
  u = NULL,
  model = NULL,
  cnst = NULL,
  s = frequency(y),
  criterion = "bic",
  h = 2 * s,
  verbose = FALSE,
  lambda = 1,
  maxOrders = c(3, 2, 3, 2, 1, 2),
  bootstrap = FALSE,
  nSimul = 5000,
  fast = FALSE
)
```

**Arguments**

y	a time series to forecast (it may be either a numerical vector or a time series object). This is the only input required. If a vector, the additional input s should be supplied compulsorily (see below).
u	a matrix of input time series. If the output wanted to be forecast, matrix u should contain future values for inputs.
model	the model to estimate. A vector c(p,d,q,P,D,Q) containing the model orders of an ARIMA(p,d,q)x(P,D,Q)_s model. A constant may be estimated with the cnst input. Use a NULL to automatically identify the ARIMA model.
cnst	flag to include a constant in the model (TRUE/FALSE/NULL). Use NULL to estimate
s	seasonal period of time series (1 for annual, 4 for quarterly, ...)
criterion	information criterion for identification stage ("aic", "bic", "aicc")
h	forecast horizon. If the model includes inputs h is not used, the length of u is used instead.
verbose	intermediate estimation output (TRUE / FALSE)
lambda	Box-Cox lambda parameter (NULL: estimate)
maxOrders	a vector c(p,d,q,P,D,Q) containing the maximum orders of model orders to search for in the automatic identification

<code>bootstrap</code>	use bootstrap simulation for predictive distributions
<code>nSimul</code>	number of simulation runs for bootstrap simulation of predictive distributions
<code>fast</code>	fast identification (avoids post-identification checks)

### Details

ARIMAforecast is a function for modelling and forecasting univariate time series with Autoregressive Integrated Moving Average (ARIMA) time series models. It sets up the model with a number of control variables that govern the way the rest of functions in the package will work. It also estimates the model parameters by Maximum Likelihood and forecasts the data.

### Value

An object of class ARIMA. It is a list with fields including all the inputs and the fields listed below as outputs. All the functions in this package fill in part of the fields of any ARIMA object as specified in what follows (function ARIMA fills in all of them at once):

After running ARIMAforecast or ARIMA:

<code>p</code>	Estimated parameters
<code>yFor</code>	Forecasted values of output
<code>yForV</code>	Variance of forecasted values of output
<code>ySimul</code>	Bootstrap simulations for forecasting distribution evaluation

After running ARIMAvalidate:

<code>table</code>	Estimation and validation table
--------------------	---------------------------------

### Author(s)

Diego J. Pedregal

### See Also

[ARIMA](#), [ARIMAvalidate](#),

### Examples

```
y <- log(AirPassengers)
m1 <- ARIMAforecast(y)
m1 <- ARIMAforecast(y, lambda = NULL)
```

ARIMAs<sub>etup</sub>*ARIMAs<sub>etup</sub>***Description**

Sets up ARIMA general models

**Usage**

```
ARIMAsetup(
  y,
  u = NULL,
  model = NULL,
  cnst = NULL,
  s = frequency(y),
  criterion = "bic",
  h = 2 * s,
  verbose = FALSE,
  lambda = 1,
  maxOrders = c(3, 2, 3, 2, 1, 2),
  bootstrap = FALSE,
  nSimul = 5000,
  fast = FALSE
)
```

**Arguments**

y	a time series to forecast (it may be either a numerical vector or a time series object). This is the only input required. If a vector, the additional input s should be supplied compulsorily (see below).
u	a matrix of input time series. If the output wanted to be forecast, matrix u should contain future values for inputs.
model	the model to estimate. A vector c(p,d,q,P,D,Q) containing the model orders of an ARIMA(p,d,q)x(P,D,Q) <sub>s</sub> model. A constant may be estimated with the cnst input. Use a NULL to automatically identify the ARIMA model.
cnst	flag to include a constant in the model (TRUE/FALSE/NULL). Use NULL to estimate
s	seasonal period of time series (1 for annual, 4 for quarterly, ...)
criterion	information criterion for identification stage ("aic", "bic", "aicc")
h	forecast horizon. If the model includes inputs h is not used, the length of u is used instead.
verbose	intermediate estimation output (TRUE / FALSE)
lambda	Box-Cox lambda parameter (NULL: estimate)
maxOrders	a vector c(p,d,q,P,D,Q) containing the maximum orders of model orders to search for in the automatic identification

bootstrap	use bootstrap simulation for predictive distributions
nSimul	number of simulation runs for bootstrap simulation of predictive distributions
fast	fast identification (avoids post-identification checks)

### Details

See help of ARIMAforecast.

### Value

An object of class ARIMA. It is a list with fields including all the inputs and the fields listed below as outputs. All the functions in this package fill in part of the fields of any ARIMA object as specified in what follows (function ARIMA fills in all of them at once):

After running ARIMAforecast or ARIMA:

p	Estimated parameters
yFor	Forecasted values of output
yForV	Variance of forecasted values of output
ySimul	Bootstrap simulations for forecasting distribution evaluation

After running ARIMAvalidate:

table	Estimation and validation table
-------	---------------------------------

### Author(s)

Diego J. Pedregal

### See Also

[ARIMA](#), [ARIMAforecast](#), [ARIMAvalidate](#),

### Examples

```
y <- log(AirPassengers)
m1 <- ARIMAsetup(y)
m1 <- ARIMAsetup(y, lambda = NULL)
```

---

ARIMAvalidate	<i>ARIMAvalidate</i>
---------------	----------------------

---

**Description**

Shows a table of estimation and diagnostics results for ARIMA models

**Usage**

```
ARIMAvalidate(m)
```

**Arguments**

`m` an object of type ARIMA created with ARIMAforecast

**Value**

The same input object with the appropriate fields filled in, in particular:

`table` Estimation and validation table

**Author(s)**

Diego J. Pedregal

**See Also**

[ARIMA](#), [ARIMAforecast](#), [ARIMAvalidate](#),

**Examples**

```
m1 <- ARIMAforecast(log(gdp))
m1 <- ARIMAvalidate(m1)
```

---

arma2tsi	<i>arma2tsi</i>
----------	-----------------

---

**Description**

AR polynomial coefficients of ARMA model

**Usage**

```
arma2tsi(MApoly, ARpoly, n = 100)
```

**Arguments**

MApoly	coefficients of numerator polynomial in descending order
ARpoly	coefficients of denominator polynomial in descending order
n	number of coefficients

**Value**

Tsi (MA form) coefficients of equivalent ARMA model

**Author(s)**

Diego J. Pedregal

---

armaFilter

*armaFilter*

---

**Description**

Filter of time series

**Usage**

```
armaFilter(MA = 1, AR = 1, y)
```

**Arguments**

MA	numerator polynomial
AR	denominator polynomial
y	a vector, ts or tsibble object

**Value**

Filtered time series

**Author(s)**

Diego J. Pedregal

**See Also**

[colMedians](#), [rowMedians](#), [tests](#), [sumStats](#), [gaussTest](#), [ident](#), [cusum](#), [varTest](#), [conv](#), [dif](#), [roots](#), [zplane](#), [acft](#), [slide](#), [plotSlide](#), [Accuracy](#), [tsDisplay](#), [size](#)

**Examples**

```
y <- armaFilter(1, c(1, -0.8), rnorm(200))
```

---

auxInvBoxCox	<i>auxInvBoxCox</i>
--------------	---------------------

---

**Description**

Inverse of Box-Cox transformation

**Usage**

```
auxInvBoxCox(y, lambda)
```

**Arguments**

y	matrix, array or vector
lambda	lambda parameter of Box-Cox transformation

**Value**

Inverse of Box-Cox heteroskedasticity transformation

**Author(s)**

Diego J. Pedregal

---

BIC.UComp	<i>BIC.UComp</i>
-----------	------------------

---

**Description**

Extract BIC (or SBC) value of UComp object

**Usage**

```
## S3 method for class 'UComp'
BIC(object, ...)
```

**Arguments**

object	Object of class "UComp".
...	Additional inputs to function.

**Details**

Selection criteria for models with different number of parameters, the smaller BIC the better. The formula used here is  $BIC = (-2\ln(L) + k\ln(n))/n$ , where  $\ln(L)$  is the log-likelihood at the optimum,  $k$  is the number of parameters plus non-stationary states and  $n$  is the number of observations. Mind that this formulation differs from the usual definition that does not divide by  $n$ . This makes that  $BIC(m)$  and  $BIC(\logLik(m))$  give different results, being  $m$  an UComp object.

**Value**

BIC value of a UC model

**Author(s)**

Diego J. Pedregal

**See Also**

[UC](#), [UCforecast](#), [UCvalidate](#), [UCfilter](#), [UCsmooth](#), [UCdisturb](#), [UCcomponents](#)

**Examples**

```
y <- log(AirPassengers)
m1 <- UCforecast(y, model = "l1t/equal/arma(0,0)")
BIC(m1)
```

---

box.cox

*box.cox*

---

**Description**

Runs Box-Cox transform of a time series

**Usage**

```
box.cox(x, lambda)
```

**Arguments**

x	Time series object.
lambda	Lambda parameter for Box-Cox transform.

**Value**

Box-Cox transformed time series

**Author(s)**

Diego J. Pedregal

**See Also**

[inv.box.cox](#), [UC](#), [UCforecast](#), [UCvalidate](#), [UCfilter](#), [UCsmooth](#), [UCdisturb](#), [UCcomponents](#)

**Examples**

```
y <- box.cox(AirPassengers, 0.5)
plot(y)
```

---

ch4

*Methane concentration at Cape Grim in Australia*

---

**Description**

Methane concentration at Cape Grim in Australia (ch4).

**Usage**

ch4

**Format**

Time series objects.

Monthly data from January 1992 to December 2019

**Value**

No return value, called for side effects

**Source**

[CH4 data](#)

**Examples**

```
ch4
```

---

colMedians

*colMedians*

---

**Description**

Medians of matrix by columns

**Usage**

```
colMedians(x, na.rm = TRUE, ...)
```

**Arguments**

`x` a matrix  
`na.rm` boolean indicating whether to remove nans  
`...` rest of inputs

**Value**

A vector with all the medians in columns

**Author(s)**

Diego J. Pedregal

**See Also**

[rowMedians](#), [tests](#), [sumStats](#), [gaussTest](#), [ident](#), [cusum](#), [varTest](#), [conv](#), [armaFilter](#), [dif](#), [roots](#), [zplane](#), [acft](#), [slide](#), [plotSlide](#), [Accuracy](#), [tsDisplay](#), [size](#)

**Examples**

```
s <- colMedians(matrix(4, 3, 2))
```

---

conv

*conv*

---

**Description**

1D convolution: filtering or polynomial multiplication

**Usage**

```
conv(...)
```

**Arguments**

`...` list of vectors to convolute

**Value**

Convolution of all input vectors

**Author(s)**

Diego J. Pedregal

**See Also**

[colMedians](#), [rowMedians](#), [tests](#), [sumStats](#), [gaussTest](#), [ident](#), [cusum](#), [varTest](#), [armaFilter](#), [dif](#), [roots](#), [zplane](#), [acft](#), [slide](#), [plotSlide](#), [Accuracy](#), [tsDisplay](#), [size](#)

**Examples**

```
conv(c(1, -1), c(1, -2, 1))  
conv(c(1, -1), c(1, 0.8))
```

---

cusum

*cusum*

---

**Description**

Cusum and cusumsq tests

**Usage**

```
cusum(y, runFromTest = FALSE)
```

**Arguments**

`y` a vector, ts or tsibble object  
`runFromTest` internal check variable

**Value**

No return value, called for side effects

**Author(s)**

Diego J. Pedregal

**See Also**

[colMedians](#), [rowMedians](#), [tests](#), [sumStats](#), [gaussTest](#), [ident](#), [varTest](#), [conv](#), [armaFilter](#), [dif](#), [roots](#), [zplane](#), [acft](#), [slide](#), [plotSlide](#), [Accuracy](#), [tsDisplay](#), [size](#)

**Examples**

```
cusum(AirPassengers)
```

---

 DAYforecast

*DAYforecast*


---

### Description

Estimates and forecasts daily time series with weekly and monthly seasonal patterns

### Usage

```
DAYforecast(
  y,
  u = NULL,
  initial_date,
  h = 24,
  lambda = 1,
  criterion = "aic",
  p0 = -9999.9,
  verbose = FALSE
)
```

### Arguments

<code>y</code>	A numeric vector, ts object or list containing daily observations of the dependent variable.
<code>u</code>	Optional matrix or vector of exogenous regressors. If provided, it must cover both the estimation and forecast periods.
<code>initial_date</code>	A character or Date object indicating the calendar start date of the series.
<code>h</code>	Integer indicating the forecast horizon (number of days ahead).
<code>lambda</code>	Box–Cox transformation parameter (NULL for automatic identification).
<code>criterion</code>	Information criterion used for model selection (AIC, BIC, AICc).
<code>p0</code>	Initial parameter vector for Maximum Likelihood estimation.
<code>verbose</code>	if TRUE, prints estimation details.

### Details

DAYforecast estimates and forecasts daily univariate time series using Dynamic Harmonic Regression (DHR).

The function is tailored for daily data and explicitly accounts for:

- Monthly seasonality through sinusoidal terms based on calendar months
- Weekly seasonality through sinusoidal terms with periods of 7 days

Standard methods applicable to UComp objects are available.

**Value**

An object of class UComp. It is a list containing the model specification, estimated parameters, and forecasting results.

**Author(s)**

Diego J. Pedregal

**See Also**

[UC](#), [UCforecast](#), [UCfilter](#), [UCsmooth](#), [UCdisturb](#), [UCcomponents](#)

**Examples**

```
y <- rnorm(400)
m <- DAYforecast(y, initial_date = "2015-01-01", h = 30)
```

---

dif

*dif*

---

**Description**

Discrete differencing of time series

**Usage**

```
dif(y, difs = 1, seas = 1)
```

**Arguments**

y	a vector, ts or tibble object
difs	vector with differencing orders
seas	vector of seasonal periods

**Value**

Differenced time series

**Author(s)**

Diego J. Pedregal

**See Also**

[colMedians](#), [rowMedians](#), [tests](#), [sumStats](#), [gaussTest](#), [ident](#), [cusum](#), [varTest](#), [conv](#), [armaFilter](#), [roots](#), [zplane](#), [acft](#), [slide](#), [plotSlide](#), [Accuracy](#), [tsDisplay](#), [size](#)

**Examples**

```
dif(AirPassengers)
dif(AirPassengers, 2)
dif(AirPassengers, c(1, 1), c(1, 12))
```

---

ETS

*ETS*


---

**Description**

Runs all relevant functions for ETS modelling

**Usage**

```
ETS(
  y,
  u = NULL,
  model = "???",
  s = frequency(y),
  h = 2 * s,
  criterion = "aicc",
  lambda = 1,
  armaIdent = FALSE,
  identAll = FALSE,
  forIntervals = FALSE,
  bootstrap = FALSE,
  nSimul = 5000,
  verbose = FALSE,
  alphaL = c(1e-08, 1 - 1e-08),
  betaL = alphaL,
  gammaL = alphaL,
  phiL = c(0.8, 0.98),
  p0 = -99999
)
```

**Arguments**

- |                    |  |
|--------------------|--|
| <code>y</code>     | a time series to forecast (it may be either a numerical vector or a time series object). This is the only input required. If a vector, the additional input <code>s</code> should be supplied compulsorily (see below).              |
| <code>u</code>     | a matrix of input time series. If the output wanted to be forecast, matrix <code>u</code> should contain future values for inputs.   |
| <code>model</code> | the model to estimate. It is a single string indicating the type of model for each component with one or two letters: <ul style="list-style-type: none"> <li>• Error: ? / A / M</li> <li>• Trend: ? / N / A / Ad / M / Md</li> </ul> |

	<ul style="list-style-type: none"> <li>Seasonal: ? / N / A / M</li> </ul>
s	seasonal period of time series (1 for annual, 4 for quarterly, ...)
h	forecast horizon. If the model includes inputs h is not used, the length of u is used instead.
criterion	information criterion for identification ("aic", "bic" or "aicc").
lambda	Box-Cox lambda parameter (NULL: estimate)
armaIdent	check for arma models for error component (TRUE / FALSE).
identAll	run all models to identify the best one (TRUE / FALSE)
forIntervals	estimate forecasting intervals (TRUE / FALSE)
bootstrap	use bootstrap simulation for predictive distributions
nSimul	number of simulation runs for bootstrap simulation of predictive distributions
verbose	intermediate estimation output (TRUE / FALSE)
alphaL	constraints limits for alpha parameter
betaL	constraints limits for beta parameter
gammaL	constraints limits for gamma parameter
phiL	constraints limits for phi parameter
p0	initial values for parameter search (alpha, beta, phi, gamma) with constraints: <ul style="list-style-type: none"> <li><math>0 &lt; \alpha &lt; 1</math></li> <li><math>0 &lt; \beta &lt; \alpha</math></li> <li><math>0 &lt; \phi &lt; 1</math></li> <li><math>0 &lt; \gamma &lt; 1 - \alpha</math></li> </ul>

### Details

See help of ETSforecast.

### Value

An object of class ETS. It is a list with fields including all the inputs and the fields listed below as outputs. All the functions in this package fill in part of the fields of any ETS object as specified in what follows (function ETS fills in all of them at once):

After running ETSforecast:

p	Estimated parameters
criteria	Values for estimation criteria (LogLik, AIC, BIC, AICc)
yFor	Forecasted values of output
yForV	Variance of forecasted values of output
ySimul	Bootstrap simulations for forecasting distribution evaluation

After running ETSvalidate:

table	Estimation and validation table
comp	Estimated components in matrix form

After running ETScomponents:

comp                    Estimated components in matrix form

An object of class ETS. See ETSforecast.

**Author(s)**

Diego J. Pedregal

**See Also**

[ETSforecast](#), [ETSvalidate](#), [ETScomponents](#)

**Examples**

```
y <- log(AirPassengers)
m1 <- ETS(y)
m1 <- ETS(y, model = "MAM")
```

---

ETScomponents

*ETScomponents*

---

**Description**

Estimates components of ETS models

**Usage**

```
ETScomponents(m)
```

**Arguments**

m                    an object of type ETS created with ETSforecast

**Value**

The same input object with the appropriate fields filled in, in particular:

comp                    Estimated components in matrix form

**Author(s)**

Diego J. Pedregal

**See Also**

[ETS](#), [ETSforecast](#), [ETSvalidate](#)

**Examples**

```
m1 <- ETS(log(gdp))
m1 <- ETScomponents(m1)
```

---

**ETSestim***ETSestim*

---

**Description**

Estimates and forecasts ETS models

**Usage**

```
ETSestim(m)
```

**Arguments**

**m** an object of type ETS created with ETSforecast

**Details**

ETSestim estimates and forecasts a time series using an an ETS model

**Value**

The same input object with the appropriate fields filled in, in particular:

<b>p</b>	Estimated parameters
<b>yFor</b>	Forecasted values of output
<b>yForV</b>	Variance of forecasted values of output
<b>ySimul</b>	Bootstrap simulations for forecasting distribution evaluation

**Author(s)**

Diego J. Pedregal

**See Also**

[ETS](#), [ETSforecast](#), [ETSvalidate](#), [ETScomponents](#)

**Examples**

```
m1 <- ETSsetup(log(gdp))
m1 <- ETSestim(m1)
```

ETSforecast

*ETSforecast***Description**

Estimates and forecasts ETS general univariate models

**Usage**

```
ETSforecast(
  y,
  u = NULL,
  model = "???",
  s = frequency(y),
  h = max(2 * s, 6),
  criterion = "aicc",
  lambda = 1,
  armaIdent = FALSE,
  identAll = FALSE,
  forIntervals = FALSE,
  bootstrap = FALSE,
  nSimul = 5000,
  verbose = FALSE,
  alphaL = c(1e-08, 1 - 1e-08),
  betaL = alphaL,
  gammaL = alphaL,
  phiL = c(0.8, 0.98),
  p0 = -99999
)
```

**Arguments**

y	a time series to forecast (it may be either a numerical vector or a time series object). This is the only input required. If a vector, the additional input s should be supplied compulsorily (see below).
u	a matrix of input time series. If the output wanted to be forecast, matrix u should contain future values for inputs.
model	the model to estimate. It is a single string indicating the type of model for each component with one or two letters: <ul style="list-style-type: none"> <li>• Error: ? / A / M</li> <li>• Trend: ? / N / A / Ad / M / Md</li> <li>• Seasonal: ? / N / A / M</li> </ul>
s	seasonal period of time series (1 for annual, 4 for quarterly, ...)
h	forecast horizon. If the model includes inputs h is not used, the length of u is used instead.

crit <code>erion</code>	information criterion for identification ("aic", "bic" or "aicc").
lamb <code>da</code>	Box-Cox lambda parameter (NULL: estimate)
arma <code>Ident</code>	check for arma models for error component (TRUE / FALSE).
ident <code>All</code>	run all models to identify the best one (TRUE / FALSE)
for <code>Intervals</code>	estimate forecasting intervals (TRUE / FALSE)
boot <code>strap</code>	use bootstrap simulation for predictive distributions
n <code>Simul</code>	number of simulation runs for bootstrap simulation of predictive distributions
ver <code>bose</code>	intermediate estimation output (TRUE / FALSE)
alph <code>aL</code>	constraints limits for alpha parameter
bet <code>aL</code>	constraints limits for beta parameter
gam <code>maL</code>	constraints limits for gamma parameter
phi <code>L</code>	constraints limits for phi parameter
p <code>0</code>	initial values for parameter search (alpha, beta, phi, gamma) with constraints: <ul style="list-style-type: none"> <li>• <math>0 &lt; \alpha &lt; 1</math></li> <li>• <math>0 &lt; \beta &lt; \alpha</math></li> <li>• <math>0 &lt; \phi &lt; 1</math></li> <li>• <math>0 &lt; \gamma &lt; 1 - \alpha</math></li> </ul>

## Details

ETSforecast is a function for modelling and forecasting univariate time series with Exponential Smoothing (ETS) time series models. It sets up the model with a number of control variables that govern the way the rest of functions in the package will work. It also estimates the model parameters by Maximum Likelihood and forecasts the data.

## Value

An object of class ETS. It is a list with fields including all the inputs and the fields listed below as outputs. All the functions in this package fill in part of the fields of any ETS object as specified in what follows (function ETS fills in all of them at once):

After running ETSforecast:

p	Estimated parameters
crit <code>eria</code>	Values for estimation criteria (LogLik, AIC, BIC, AICc)
y <code>For</code>	Forecasted values of output
y <code>ForV</code>	Variance of forecasted values of output
y <code>Simul</code>	Bootstrap simulations for forecasting distribution evaluation

After running ETSvalidate:

tab <code>le</code>	Estimation and validation table
comp	Estimated components in matrix form

After running ETScomponents:

comp	Estimated components in matrix form
------	-------------------------------------

**Author(s)**

Diego J. Pedregal

**See Also**

[ETS](#), [ETSvalidate](#), [ETScomponents](#)

**Examples**

```
y <- log(AirPassengers)
m1 <- ETSforecast(y)
m1 <- ETSforecast(y, model = "A?A")
```

---

ETSsetup

*ETSsetup*

---

**Description**

Sets up ETS general univariate models

**Usage**

```
ETSsetup(
  y,
  u = NULL,
  model = "???",
  s = frequency(y),
  h = 2 * s,
  criterion = "aicc",
  lambda = 1,
  armaIdent = FALSE,
  identAll = FALSE,
  forIntervals = FALSE,
  bootstrap = FALSE,
  nSimul = 5000,
  verbose = FALSE,
  alphaL = c(1e-08, 1 - 1e-08),
  betaL = alphaL,
  gammaL = alphaL,
  phiL = c(0.8, 0.98),
  p0 = -99999
)
```

**Arguments**

y	a time series to forecast (it may be either a numerical vector or a time series object). This is the only input required. If a vector, the additional input s should be supplied compulsorily (see below).
u	a matrix of input time series. If the output wanted to be forecast, matrix u should contain future values for inputs.
model	the model to estimate. It is a single string indicating the type of model for each component with one or two letters: <ul style="list-style-type: none"> <li>• Error: ? / A / M</li> <li>• Trend: ? / N / A / Ad / M / Md</li> <li>• Seasonal: ? / N / A / M</li> </ul>
s	seasonal period of time series (1 for annual, 4 for quarterly, ...)
h	forecast horizon. If the model includes inputs h is not used, the length of u is used instead.
criterion	information criterion for identification ("aic", "bic" or "aicc").
lambda	Box-Cox lambda parameter (NULL: estimate)
armaIdent	check for arma models for error component (TRUE / FALSE).
identAll	run all models to identify the best one (TRUE / FALSE)
forIntervals	estimate forecasting intervals (TRUE / FALSE)
bootstrap	use bootstrap simulation for predictive distributions
nSimul	number of simulation runs for bootstrap simulation of predictive distributions
verbose	intermediate estimation output (TRUE / FALSE)
alphaL	constraints limits for alpha parameter
betaL	constraints limits for beta parameter
gammaL	constraints limits for gamma parameter
phiL	constraints limits for phi parameter
$\rho\emptyset$	initial values for parameter search (alpha, beta, phi, gamma) with constraints: <ul style="list-style-type: none"> <li>• <math>0 &lt; \alpha &lt; 1</math></li> <li>• <math>0 &lt; \beta &lt; \alpha</math></li> <li>• <math>0 &lt; \phi &lt; 1</math></li> <li>• <math>0 &lt; \gamma &lt; 1 - \alpha</math></li> </ul>

**Details**

See help of ETSforecast.

**Value**

An object of class ETS. It is a list with fields including all the inputs and the fields listed below as outputs. All the functions in this package fill in part of the fields of any ETS object as specified in what follows (function ETS fills in all of them at once):

After running ETSforecast:

p	Estimated parameters
criteria	Values for estimation criteria (LogLik, AIC, BIC, AICc)
yFor	Forecasted values of output
yForV	Variance of forecasted values of output
ySimul	Bootstrap simulations for forecasting distribution evaluation

After running ETSvalidate:

table	Estimation and validation table
comp	Estimated components in matrix form

After running ETScomponents:

comp	Estimated components in matrix form
------	-------------------------------------

An object of class ETS. See ETSforecast.

### Author(s)

Diego J. Pedregal

### See Also

[ETS](#), [ETSforecast](#), [ETSvalidate](#), [ETScomponents](#)

### Examples

```
y <- log(AirPassengers)
m1 <- ETSsetup(y)
m1 <- ETSsetup(y, model = "???" )
m1 <- ETSsetup(y, model = "?AA" )
```

---

ETSvalidate

*ETSvalidate*

---

### Description

Shows a table of estimation and diagnostics results for ETS models

### Usage

```
ETSvalidate(m)
```

### Arguments

m an object of type ETS created with ETSforecast

**Value**

The same input object with the appropriate fields filled in, in particular:

table            Estimation and validation table

**Author(s)**

Diego J. Pedregal

**See Also**

[ETS](#), [ETSforecast](#), [ETSvalidate](#), [ETScomponents](#)

**Examples**

```
m1 <- ETSforecast(log(gdp))
m1 <- ETSvalidate(m1)
```

---

extract

*extract*

---

**Description**

Reorder data frame returning column col reordered according to the values in column accordingTo

**Usage**

```
extract(x, col, accordingTo = 1)
```

**Arguments**

x                    a data frame  
col                  column to be ordered  
accordingTo        column to take as the pattern

**Value**

Data frame reordered according to a given column data

**Author(s)**

Diego J. Pedregal

---

`gaussTest`*gaussTest*

---

**Description**

Gaussianity tests

**Usage**

```
gaussTest(y, runFromTests = FALSE)
```

**Arguments**

`y` a vector, ts or tsibble object  
`runFromTests` internal check

**Value**

No return value, called for side effects

**Author(s)**

Diego J. Pedregal

**See Also**

[colMedians](#), [rowMedians](#), [tests](#), [sumStats](#), [ident](#), [cusum](#), [varTest](#), [conv](#), [armaFilter](#), [dif](#), [roots](#), [zplane](#), [acft](#), [slide](#), [plotSlide](#), [Accuracy](#), [tsDisplay](#), [size](#)

**Examples**

```
gaussTest(AirPassengers)
```

---

`gdp`*Spanish GDP*

---

**Description**

Quarterly real Spanish Gross Domestic Product (gdp)

**Usage**

```
gdp
```

**Format**

Time series objects.

Quarterly since 1995

<https://portal.mineco.gob.es/es-es/economiayempresa/EconomiaInformesMacro/Paginas/bdsice.aspx>

**Value**

No return value, called for side effects

**Examples**

gdp

---

getp0

*getp0*

---

**Description**

Get initial conditions for parameters of UComp object

**Usage**

```
getp0(y, model = "l1t/equal/arma(0,0)", periods = NA)
```

**Arguments**

y	a time series to forecast.
model	any valid UComp model without any ?.
periods	vector of fundamental period and harmonics required.

**Details**

Provides initial parameters of a given model for the time series. They may be changed arbitrarily by the user to include as an input  $p_0$  to UC or UCforecast functions (see example below). There is no guarantee that the model will converge and selecting initial conditions should be used with care.

**Value**

A set of parameters  $p_0$  of an object of class UComp to use as input to UC, UCforecast.

**Author(s)**

Diego J. Pedregal

**See Also**

[UC](#), [UCvalidate](#), [UCfilter](#), [UCsmooth](#), [UCdisturb](#), [UCcomponents](#), [UChp](#)

**Examples**

```
## Not run:
p0 <- getp0(log(AirPassengers), model = "l1t/equal/arma(0,0)")
p0[1] <- 0 # p0[1] <- NA
m <- UCforecast(log(AirPassengers), model = "l1t/equal/arma(0,0)", p0 = p0)

## End(Not run)
```

---

 ident

*ident*


---

**Description**

Autocorrelation functions of a time series

**Usage**

```
ident(y, nCoef = min(37, floor(length(y)/4)), nPar = 0, runFromTests = FALSE)
```

**Arguments**

<code>y</code>	a vector, ts or tibble object
<code>nCoef</code>	number of autocorrelation coefficients to estimate
<code>nPar</code>	number of parameters in a model if y is a residual
<code>runFromTests</code>	internal check

**Value**

A vector with output table including ACF, etc.

**Author(s)**

Diego J. Pedregal

**See Also**

[colMedians](#), [rowMedians](#), [tests](#), [sumStats](#), [gaussTest](#), [cusum](#), [varTest](#), [conv](#), [armaFilter](#), [dif](#), [roots](#), [zplane](#), [acft](#), [slide](#), [plotSlide](#), [Accuracy](#), [tsDisplay](#), [size](#)

**Examples**

```
ident(AirPassengers)
```

---

`inv.box.cox``inv.box.cox`

---

**Description**

Runs inverse of Box-Cox transform of a time series

**Usage**

```
inv.box.cox(x, lambda)
```

**Arguments**

`x` Transformed time series object.  
`lambda` Lambda parameter used for Box-Cox transform.

**Value**

Inverse Box-Cox transformed time series

**Author(s)**

Diego J. Pedregal

**See Also**

[box.cox](#), [UC](#), [UCforecast](#), [UCvalidate](#), [UCfilter](#), [UCsmooth](#), [UCdisturb](#), [UCcomponents](#)

**Examples**

```
y <- inv.box.cox(box.cox(AirPassengers, 0.5), 0.5)
plot(y)
```

---

`invBoxCox``invBoxCox`

---

**Description**

Calculates inverse of Box-Cox transformation with confidence bands, calculated as const time the standard error

**Usage**

```
invBoxCox(y, yVar, lambda, const = 2)
```

**Arguments**

y	matrix, array or vector
yVar	matrix, array or vector of variances of y
lambda	lambda parameter of Box-Cox transformation
const	number of standard error for confidence band

**Value**

Inverse of Box-Cox heteroskedasticity transformation with confidence bands

**Author(s)**

Diego J. Pedregal

---

ipi	<i>Spanish Industrial Production Index</i>
-----	--

---

**Description**

Spanish Industrial Production Index (ipi).

**Usage**

ipi

**Format**

Objeto time series.

Monthly since 1975

<https://portal.mineco.gob.es/es-es/economiayempresa/EconomiaInformesMacro/Paginas/bdsice.aspx>

**Value**

No return value, called for side effects

**Examples**

ipi

---

OECDgdp	<i>OECD GDP</i>
---------	-----------------

---

**Description**

Seasonally adjusted quarterly OECD real gross domestic product (OECDgdp).

**Usage**

OECDgdp

**Format**

Time series objects.

Quarterly data from 1962 to 2019

<https://portal.mineco.gob.es/es-es/economiayempresa/EconomiaInformesMacro/Paginas/bdsice.aspx>

**Value**

No return value, called for side effects

**Examples**

OECDgdp

---

plotAcfPacf	<i>plotAcfPacf</i>
-------------	--------------------

---

**Description**

Plot of ACF and PACF

**Usage**

plotAcfPacf(ACF, PACF, s = 1, n = NA, runFromTest = FALSE)

**Arguments**

ACF	variable to plot
PACF	second variable to plot
s	seasonal period
n	number of coefficients
runFromTest	internal check variable

**Value**

No return value, called for side effects

**Author(s)**

Diego J. Pedregal

---

<code>plotBar</code>	<i>plotBar</i>
----------------------	----------------

---

**Description**

Plot variable in bars

**Usage**

```
plotBar(ACF, s = 1, n = NA, label = "ACF")
```

**Arguments**

ACF	variable to plot
s	seasonal period
n	number of coefficients
label	label for plot

**Value**

Handle of plot

**Author(s)**

Diego J. Pedregal

---

plotSlide	<i>plotSlide</i>
-----------	------------------

---

**Description**

Plot summarised results from slide

**Usage**

```
plotSlide(py1, y, orig, step = 1, errorFun, collectFun = mean)
```

**Arguments**

py1	output from slide function
y	a vector, matrix or list of time series (the same used in slide call)
orig	starting forecasting origin (the same used in slide call)
step	observations ahead to move the forecasting origin (the same used in slide call)
errorFun	user function to calculate error measures
collectFun	aggregation function (mean, median, etc.)

**Value**

An array of forecasting errors of dimensions (horizon x nOrigs x nModels x nSeries)

**Author(s)**

Diego J. Pedregal

**See Also**

[colMedians](#), [rowMedians](#), [tests](#), [sumStats](#), [gaussTest](#), [ident](#), [cusum](#), [varTest](#), [conv](#), [armaFilter](#), [dif](#), [roots](#), [zplane](#), [acft](#), [slide](#), [Accuracy](#), [tsDisplay](#), [size](#)

**Examples**

```
## Not run: plotSlide(py1, AirPassengers, 100, 1, errorFun)
```

---

plus_one	<i>plus_one</i>
----------	-----------------

---

**Description**

Returns date of next to end time series y

**Usage**

```
plus_one(y)
```

**Arguments**

y	a ts object
---	-------------

**Value**

Next time stamp

**Author(s)**

Diego J. Pedregal

---

predict.UComp	<i>predict.UComp</i>
---------------	----------------------

---

**Description**

Forecasting using structural Unobserved Components models with prediction intervals

**Usage**

```
## S3 method for class 'UComp'
predict(object, newdata = NULL, n.ahead = NULL, level = 0.95, ...)
```

**Arguments**

object	Object of class “UComp”.
newdata	New output data to apply “UComp” object to.
n.ahead	Number of steps ahead to forecast or new inputs variables including their predictions.
level	Confidence level for prediction intervals.
...	Ignored.

**Details**

See help of UC.

**Value**

Forecasts of a UC model

A matrix with the mean forecasts and lower and upper prediction intervals

**Author(s)**

Diego J. Pedregal

**See Also**

[UC](#), [UCforecast](#), [UCvalidate](#), [UCfilter](#), [UCsmooth](#), [UCdisturb](#), [UCcomponents](#)

**Examples**

```
y <- log(AirPassengers)
m1 <- UCforecast(y, model = "l1t/eq/arma(0,0)")
f1 <- predict(m1)
```

---

removeNaNs

*removeNaNs*

---

**Description**

Remove nans at beginning or end of vector

**Usage**

```
removeNaNs(x)
```

**Arguments**

x                    a vector or a ts object

**Value**

vector with nans removed (only those at beginning or end)

**Author(s)**

Diego J. Pedregal

roots

*roots*

---

**Description**

Roots of polynomial

**Usage**

```
roots(x)
```

**Arguments**

x                    coefficients of polynomial in descending order

**Value**

Roots of polynomial

**Author(s)**

Diego J. Pedregal

**See Also**

[colMedians](#), [rowMedians](#), [tests](#), [sumStats](#), [gaussTest](#), [ident](#), [cusum](#), [varTest](#), [conv](#), [armaFilter](#), [dif](#), [zplane](#), [acft](#), [slide](#), [plotSlide](#), [Accuracy](#), [tsDisplay](#), [size](#)

**Examples**

```
roots(c(1, -2, 1))
roots(conv(c(1, -1), c(1, 0.8)))
```

---

rowMedians*rowMedians*

---

**Description**

Medians of matrix by rows

**Usage**

```
rowMedians(x, na.rm = TRUE, ...)
```

**Arguments**

x                    a matrix  
na.rm                boolean indicating whether to remove nans  
...                   rest of inputs

**Value**

A vector with all the medians in rows

**Author(s)**

Diego J. Pedregal

**See Also**

[colMedians](#), [tests](#), [sumStats](#), [gaussTest](#), [ident](#), [cusum](#), [varTest](#), [conv](#), [armaFilter](#), [dif](#),  
[roots](#), [zplane](#), [acft](#), [slide](#), [plotSlide](#), [Accuracy](#), [tsDisplay](#), [size](#)

**Examples**

```
s <- rowMedians(matrix(4, 3, 2))
```

---

sales	<i>Sales index for large retailers in Spain</i>
-------	---

---

**Description**

Sales index for food of large retailers in Spain

**Usage**

```
sales
```

**Format**

Time series objects.

Monthly data from January 1995 to December 2019

<https://portal.mineco.gob.es/es-es/economiayempresa/EconomiaInformesMacro/Paginas/bdsice.aspx>

**Value**

No return value, called for side effects

**Examples**

```
sales
```

size

*size*

---

**Description**

Size of vector, matrix or array

**Usage**

```
size(y)
```

**Arguments**

*y* a vector, matrix or array

**Value**

A vector with all the dimensions

**Author(s)**

Diego J. Pedregal

**See Also**

[colMedians](#), [rowMedians](#), [tests](#), [sumStats](#), [gaussTest](#), [ident](#), [cusum](#), [varTest](#), [conv](#), [armaFilter](#), [dif](#), [roots](#), [zplane](#), [acft](#), [slide](#), [plotSlide](#), [Accuracy](#), [tsDisplay](#)

**Examples**

```
s <- size(matrix(4, 3, 2))
s <- size(rep(4, 3))
s <- size(array(4, c(3, 2, 2)))
```

---

slide*slide*

---

**Description**

Rolling forecasting of a matrix of time series

**Usage**

```
slide(  
  y,  
  orig,  
  forecFun,  
  ...,  
  h = 12,  
  step = 1,  
  output = TRUE,  
  window = NA,  
  parallel = FALSE  
)
```

**Arguments**

y	a vector, a matrix or a list of time series
orig	starting forecasting origin
forecFun	user function that implements forecasting methods
...	rest of inputs to forecFun function
h	forecasting horizon
step	observations ahead to move the forecasting origin
output	output TRUE/FALSE
window	fixed window width in number of observations (NA for non fixed)
parallel	run forecasts in parallel

**Details**

Takes time series and run forecasting methods implemented in function forecFun h steps ahead along the time series y, starting at forecasting origin orig, and moving step observations ahead. Forecasts may be run in parallel by setting parallel to TRUE. A fixed window width may be specified with input window. The output is of dimensions (h, nOrigs, nModels, nSeries)

**Value**

An array of forecasts of dimensions (horizon x nOrigs x nModels x nSeries)

**Author(s)**

Diego J. Pedregal

**See Also**

[colMedians](#), [rowMedians](#), [tests](#), [sumStats](#), [gaussTest](#), [ident](#), [cusum](#), [varTest](#), [conv](#), [armaFilter](#), [dif](#), [roots](#), [zplane](#), [acft](#), [plotSlide](#), [Accuracy](#), [tsDisplay](#), [size](#)

**Examples**

```
## Not run: slide(AirPassengers, 100, forecFun)
```

---

slideAux	<i>slideAux</i>
----------	-----------------

---

**Description**

Auxiliary function run from slide

**Usage**

```
slideAux(  
  y,  
  orig,  
  forecFun,  
  h = 12,  
  step = 1,  
  output = TRUE,  
  graph = TRUE,  
  window = NA,  
  parallel = FALSE,  
  isList = FALSE,  
  ...  
)
```

**Arguments**

y	a vector or matrix of time series
orig	starting forecasting origin
forecFun	user function that implements forecasting methods
h	forecasting horizon
step	observations ahead to move the forecasting origin
output	output TRUE/FALSE
graph	graphical output TRUE/FALSE
window	fixed window width in number of observations (NA for non fixed)
parallel	run forecasts in parallel
isList	whether the input data y is a list or a matrix
...	rest of inputs to forecFun function

**Value**

Auxiliary output of slide function for just one time series

**Author(s)**

Diego J. Pedregal

---

sumStats	<i>sumStats</i>
----------	-----------------

---

**Description**

Summary statistics of a matrix of variables

**Usage**

```
sumStats(y, decimals = 5)
```

**Arguments**

y	a vector, matrix of time series
decimals	number of decimals for table

**Details**

Position, dispersion, skewness, kurtosis, etc.

**Value**

Table of values in string matrix

**Author(s)**

Diego J. Pedregal

**See Also**

[colMedians](#), [rowMedians](#), [tests](#), [gaussTest](#), [ident](#), [cusum](#), [varTest](#), [conv](#), [armaFilter](#), [dif](#), [roots](#), [zplane](#), [acft](#), [slide](#), [plotSlide](#), [Accuracy](#), [tsDisplay](#), [size](#)

**Examples**

```
s <- sumStats(AirPassengers)
```

---

 tests

*tests*


---

**Description**

Tests on a time series

**Usage**

```
tests(
  y,
  parts = 1/3,
  nCoef = min(25, length(x)/4),
  nPar = 0,
  s = frequency(y),
  avoid = 16
)
```

**Arguments**

<code>y</code>	a vector, ts or tsibble object
<code>parts</code>	proportion of sample to include in ratio of variances test
<code>nCoef</code>	number of autocorrelation coefficients to estimate
<code>nPar</code>	number of parameters in a model if <code>y</code> is a residual
<code>s</code>	seasonal period, number of observations per year
<code>avoid</code>	number of observations to avoid at beginning of sample to eliminate initial effects

**Details**

Multiple tests on a time series, including summary statistics, autocorrelation, Gaussianity and heteroskedasticity,

**Value**

Table with all test results

**Author(s)**

Diego J. Pedregal

**See Also**

[colMedians](#), [rowMedians](#), [sumStats](#), [gaussTest](#), [ident](#), [cusum](#), [varTest](#), [conv](#), [armaFilter](#), [dif](#), [roots](#), [zplane](#), [acft](#), [slide](#), [plotSlide](#), [Accuracy](#), [tsDisplay](#), [size](#)

**Examples**

```
tests(AirPassengers)
```

---

TETS

*TETS*


---

**Description**

Runs all relevant functions for TETS modelling

**Usage**

```
TETS(
  y,
  u = NULL,
  model = "???",
  s = frequency(y),
  h = 2 * s,
  criterion = "aicc",
  forIntervals = FALSE,
  bootstrap = FALSE,
  nSimul = 5000,
  verbose = FALSE,
  alphaL = c(0, 1),
  betaL = alphaL,
  gammaL = alphaL,
  phiL = c(0.8, 0.98),
  p0 = -99999,
  Ymin = -Inf,
  Ymax = Inf
)
```

**Arguments**

- |       |  |
|-------|--|
| y     | a time series to forecast (it may be either a numerical vector or a time series object). This is the only input required. If a vector, the additional input s should be supplied compulsorily (see below).   |
| u     | a matrix of input time series. If the output wanted to be forecast, matrix u should contain future values for inputs.  |
| model | the model to estimate. It is a single string indicating the type of model for each component with one or two letters: <ul style="list-style-type: none"> <li>• Error: ? / A</li> <li>• Trend: ? / N / A / Ad</li> <li>• Seasonal: ? / N / A</li> </ul> |
| s     | seasonal period of time series (1 for annual, 4 for quarterly, ...)  |

<code>h</code>	forecast horizon. If the model includes inputs <code>h</code> is not used, the length of <code>u</code> is used instead.
<code>criterion</code>	information criterion for identification ("aic", "bic" or "aicc").
<code>forIntervals</code>	estimate forecasting intervals (TRUE / FALSE)
<code>bootstrap</code>	use bootstrap simulation for predictive distributions
<code>nSimul</code>	number of simulation runs for bootstrap simulation of predictive distributions
<code>verbose</code>	intermediate estimation output (TRUE / FALSE)
<code>alphaL</code>	constraints limits for alpha parameter
<code>betaL</code>	constraints limits for beta parameter
<code>gammaL</code>	constraints limits for gamma parameter
<code>phiL</code>	constraints limits for phi parameter
<code>p0</code>	initial values for parameter search (alpha, beta, phi, gamma, sigma2) with constraints:
<code>Ymin</code>	scalar or vector of time varying censoring values from below
<code>Ymax</code>	scalar or vector of time varying censoring values from above <ul style="list-style-type: none"> <li>• <math>0 &lt; \alpha &lt; 1</math></li> <li>• <math>0 &lt; \beta &lt; \alpha</math></li> <li>• <math>0 &lt; \phi &lt; 1</math></li> <li>• <math>0 &lt; \gamma &lt; 1 - \alpha</math></li> <li>• <math>\sigma^2 &gt; 0</math></li> </ul>

### Details

See help of `TETSforecast`.

### Value

An object of class `TETS`. It is a list with fields including all the inputs and the fields listed below as outputs. All the functions in this package fill in part of the fields of any `TETS` object as specified in what follows (function `TETS` fills in all of them at once):

After running `TETSforecast`:

<code>p</code>	Estimated parameters
<code>criteria</code>	Values for estimation criteria (LogLik, AIC, BIC, AICc)
<code>yFor</code>	Forecasted values of output
<code>yForV</code>	Variance of forecasted values of output
<code>ySimul</code>	Bootstrap simulations for forecasting distribution evaluation

After running `TETSvalidate`:

<code>table</code>	Estimation and validation table
<code>comp</code>	Estimated components in matrix form

After running `TETScomponents`:

<code>comp</code>	Estimated components in matrix form
-------------------	-------------------------------------

**Author(s)**

Diego J. Pedregal

**See Also**

[TETSforecast](#), [TETSvalidate](#), [TETScomponents](#)

**Examples**

```
y <- log(AirPassengers)
m1 <- TETS(y)
m1 <- TETS(y, model = "MAM")
```

---

TETScomponents

*TETScomponents*

---

**Description**

Estimates components of TOBIT TETS models

**Usage**

```
TETScomponents(m)
```

**Arguments**

`m` an object of type TETS created with TETSforecast

**Value**

The same input object with the appropriate fields filled in, in particular:

`comp` Estimated components in matrix form

**Author(s)**

Diego J. Pedregal

**See Also**

[TETS](#), [TETSforecast](#), [TETSvalidate](#)

**Examples**

```
m1 <- TETS(log(gdp))
m1 <- TETScomponents(m1)
```

---

TETSestim

*TETSestim*

---

### Description

Estimates and forecasts TOBIT TETS models

### Usage

```
TETSestim(m)
```

### Arguments

`m` an object of type TETS created with TETSforecast

### Details

TETSestim estimates and forecasts a time series using an a TOBIT TETS model

### Value

The same input object with the appropriate fields filled in, in particular:

<code>p</code>	Estimated parameters
<code>yFor</code>	Forecasted values of output
<code>yForV</code>	Variance of forecasted values of output
<code>ySimul</code>	Bootstrap simulations for forecasting distribution evaluation

### Author(s)

Diego J. Pedregal

### See Also

[TETS](#), [TETSforecast](#), [TETSvalidate](#), [TETScomponents](#)

### Examples

```
m1 <- TETSsetup(log(gdp))
m1 <- TETSestim(m1)
```

TETSforecast

*TETSforecast***Description**

Estimates and forecasts TOBIT TETS general univariate models

**Usage**

```
TETSforecast(
  y,
  u = NULL,
  model = "???",
  s = frequency(y),
  h = max(2 * s, 6),
  criterion = "aicc",
  forIntervals = FALSE,
  bootstrap = FALSE,
  nSimul = 5000,
  verbose = FALSE,
  alphaL = c(0, 1),
  betaL = alphaL,
  gammaL = alphaL,
  phiL = c(0.8, 0.98),
  p0 = -99999,
  Ymin = -Inf,
  Ymax = Inf
)
```

**Arguments**

y	a time series to forecast (it may be either a numerical vector or a time series object). This is the only input required. If a vector, the additional input s should be supplied compulsorily (see below).
u	a matrix of input time series. If the output wanted to be forecast, matrix u should contain future values for inputs.
model	the model to estimate. It is a single string indicating the type of model for each component with one or two letters: <ul style="list-style-type: none"> <li>• Error: ? / A</li> <li>• Trend: ? / N / A / Ad</li> <li>• Seasonal: ? / N / A</li> </ul>
s	seasonal period of time series (1 for annual, 4 for quarterly, ...)
h	forecast horizon. If the model includes inputs h is not used, the length of u is used instead.
criterion	information criterion for identification ("aic", "bic" or "aicc").

forIntervals	estimate forecasting intervals (TRUE / FALSE)
bootstrap	use bootstrap simulation for predictive distributions
nSimul	number of simulation runs for bootstrap simulation of predictive distributions
verbose	intermediate estimation output (TRUE / FALSE)
alphaL	constraints limits for alpha parameter
betaL	constraints limits for beta parameter
gammaL	constraints limits for gamma parameter
phiL	constraints limits for phi parameter
p0	initial values for parameter search (alpha, beta, phi, gamma, sigma2) with constraints:
Ymin	scalar or vector of time varying censoring values from below
Ymax	scalar or vector of time varying censoring values from above

- $0 < \alpha < 1$
- $0 < \beta < \alpha$
- $0 < \phi < 1$
- $0 < \gamma < 1 - \alpha$
- $\sigma^2 > 0$

## Details

TETSforecast is a function for modelling and forecasting univariate time series with TOBIT Exponential Smoothing (TETS) time series models. It sets up the model with a number of control variables that govern the way the rest of functions in the package will work. It also estimates the model parameters by Maximum Likelihood and forecasts the data.

## Value

An object of class TETS. It is a list with fields including all the inputs and the fields listed below as outputs. All the functions in this package fill in part of the fields of any TETS object as specified in what follows (function TETS fills in all of them at once):

After running TETSforecast:

p	Estimated parameters
criteria	Values for estimation criteria (LogLik, AIC, BIC, AICc)
yFor	Forecasted values of output
yForV	Variance of forecasted values of output
ySimul	Bootstrap simulations for forecasting distribution evaluation

After running TETSvalidate:

table	Estimation and validation table
comp	Estimated components in matrix form

After running TETScomponents:

comp	Estimated components in matrix form
------	-------------------------------------

**Author(s)**

Diego J. Pedregal

**See Also**

[TETS](#), [TETSvalidate](#), [TETScomponents](#)

**Examples**

```
y <- log(AirPassengers)
m1 <- TETSforecast(y)
m1 <- TETSforecast(y, model = "A?A")
```

---

TETSsetup

*TETSsetup*

---

**Description**

Sets up TOBIT TETS general univariate models

**Usage**

```
TETSsetup(
  y,
  u = NULL,
  model = "???",
  s = frequency(y),
  h = 2 * s,
  criterion = "aicc",
  forIntervals = FALSE,
  bootstrap = FALSE,
  nSimul = 5000,
  verbose = FALSE,
  alphaL = c(0, 1),
  betaL = alphaL,
  gammaL = alphaL,
  phiL = c(0.8, 0.98),
  p0 = -99999,
  Ymin = -Inf,
  Ymax = Inf
)
```

**Arguments**

y	a time series to forecast (it may be either a numerical vector or a time series object). This is the only input required. If a vector, the additional input s should be supplied compulsorily (see below).
u	a matrix of input time series. If the output wanted to be forecast, matrix u should contain future values for inputs.
model	the model to estimate. It is a single string indicating the type of model for each component with one or two letters: <ul style="list-style-type: none"> <li>• Error: ? / A</li> <li>• Trend: ? / N / A / Ad</li> <li>• Seasonal: ? / N / A</li> </ul>
s	seasonal period of time series (1 for annual, 4 for quarterly, ...)
h	forecast horizon. If the model includes inputs h is not used, the length of u is used instead.
criterion	information criterion for identification ("aic", "bic" or "aicc").
forIntervals	estimate forecasting intervals (TRUE / FALSE)
bootstrap	use bootstrap simulation for predictive distributions
nSimul	number of simulation runs for bootstrap simulation of predictive distributions
verbose	intermediate estimation output (TRUE / FALSE)
alphaL	constraints limits for alpha parameter
betaL	constraints limits for beta parameter
gammaL	constraints limits for gamma parameter
phiL	constraints limits for phi parameter
p0	initial values for parameter search (alpha, beta, phi, gamma, sigma2) with constraints:
Ymin	scalar or vector of time varying censoring values from below
Ymax	scalar or vector of time varying censoring values from above <ul style="list-style-type: none"> <li>• <math>0 &lt; \alpha &lt; 1</math></li> <li>• <math>0 &lt; \beta &lt; \alpha</math></li> <li>• <math>0 &lt; \phi &lt; 1</math></li> <li>• <math>0 &lt; \gamma &lt; 1 - \alpha</math></li> <li>• <math>\sigma_2 &gt; 0</math></li> </ul>

**Details**

See help of TETSforecast.

**Value**

An object of class TETS. It is a list with fields including all the inputs and the fields listed below as outputs. All the functions in this package fill in part of the fields of any TETS object as specified in what follows (function TETS fills in all of them at once):

After running TETSforecast:

p	Estimated parameters
criteria	Values for estimation criteria (LogLik, AIC, BIC, AICc)
yFor	Forecasted values of output
yForV	Variance of forecasted values of output
ySimul	Bootstrap simulations for forecasting distribution evaluation

After running TETSvalidate:

table	Estimation and validation table
comp	Estimated components in matrix form

After running TETScomponents:

comp	Estimated components in matrix form
------	-------------------------------------

**Author(s)**

Diego J. Pedregal

**See Also**

[TETS](#), [TETSforecast](#), [TETSvalidate](#), [TETScomponents](#)

**Examples**

```
y <- log(AirPassengers)
m1 <- TETSsetup(y)
m1 <- TETSsetup(y, model = "??")
m1 <- TETSsetup(y, model = "?AA")
```

---

TETSvalidate

*TETSvalidate*

---

**Description**

Shows a table of estimation and diagnostics results for TOBIT TETS models

**Usage**

TETSvalidate(m)

**Arguments**

`m` an object of type TETS created with TETSforecast

**Value**

The same input object with the appropriate fields filled in, in particular:

`table` Estimation and validation table

**Author(s)**

Diego J. Pedregal

**See Also**

[TETS](#), [TETSforecast](#), [TETSvalidate](#), [TETScomponents](#)

**Examples**

```
m1 <- TETSforecast(log(gdp))
m1 <- TETSvalidate(m1)
```

---

tsDisplay

*tsDisplay*

---

**Description**

Displays time series plot with autocorrelation functions

**Usage**

```
tsDisplay(y, nCoef = 25, nPar = 0, s = NA)
```

**Arguments**

`y` a vector, ts or tsibble object  
`nCoef` number of autocorrelation coefficients to estimate  
`nPar` number of parameters in a model if `y` is a residual  
`s` seasonal period, number of observations per year

**Value**

No return value, called for side effects

**Author(s)**

Diego J. Pedregal

**See Also**

[colMedians](#), [rowMedians](#), [tests](#), [sumStats](#), [gaussTest](#), [ident](#), [cusum](#), [varTest](#), [conv](#), [armaFilter](#), [dif](#), [roots](#), [zplane](#), [acft](#), [slide](#), [plotSlide](#), [Accuracy](#), [size](#)

**Examples**

```
tsDisplay(AirPassengers)
```

---

 UC

 UC
 

---

**Description**

Runs all relevant functions for UC modelling

**Usage**

```
UC(
  y,
  u = NULL,
  model = "?/none/?/?",
  h = 24,
  lambda = 1,
  outlier = 9999,
  tTest = FALSE,
  criterion = "aic",
  periods = NA,
  verbose = FALSE,
  stepwise = FALSE,
  p0 = -9999.9,
  arma = FALSE,
  TVP = NULL,
  trendOptions = "none/rw/llt/dt",
  seasonalOptions = "none/equal/different",
  irregularOptions = "none/arma(0,0)"
)
```

**Arguments**

- |   |  |
|---|--|
| y | a time series to forecast (it may be either a numerical vector or a time series object). This is the only input required. If a vector, the additional input periods should be supplied compulsorily (see below).                 |
| u | a matrix of external regressors included only in the observation equation. (it may be either a numerical vector or a time series object). If the output wanted to be forecast, matrix u should contain future values for inputs. |

model	<p>the model to estimate. It is a single string indicating the type of model for each component. It allows two formats "trend/seasonal/irregular" or "trend/cycle/seasonal/irregular". The possibilities available for each component are:</p> <ul style="list-style-type: none"> <li>• Trend: ? / none / rw / irw / llt / dt / td;</li> <li>• Seasonal: ? / none / equal / different;</li> <li>• Irregular: ? / none / arma(0, 0) / arma(p, q) - with p and q integer positive orders;</li> <li>• Cycles: ? / none / combination of positive or negative numbers. Positive numbers fix the period of the cycle while negative values estimate the period taking as initial condition the absolute value of the period supplied. Several cycles with positive or negative values are possible and if a question mark is included, the model test for the existence of the cycles specified. The following are valid examples with different meanings: 48, 48?, -48, -48?, 48+60, -48+60, -48-60, 48-60, 48+60?, -48+60?, -48-60?, 48-60?.</li> </ul>
h	forecast horizon. If the model includes inputs h is not used, the length of u is used instead.
lambda	Box-Cox transformation lambda, NULL for automatic estimation
outlier	critical level of outlier tests. If NA it does not carry out any outlier detection (default). A positive value indicates the critical minimum t test for outlier detection in any model during identification. Three types of outliers are identified, namely Additive Outliers (AO), Level Shifts (LS) and Slope Change (SC).
tTest	augmented Dickey Fuller test for unit roots used in stepwise algorithm (TRUE / FALSE). The number of models to search for is reduced, depending on the result of this test.
criterion	information criterion for identification ("aic", "bic" or "aicc").
periods	vector of fundamental period and harmonics required.
verbose	intermediate results shown about progress of estimation (TRUE / FALSE).
stepwise	stepwise identification procedure (TRUE / FALSE).
p0	initial parameter vector for optimisation search.
arma	check for arma models for irregular components (TRUE / FALSE).
TVP	vector of zeros and ones to indicate TVP parameters.
trendOptions	trend models to select amongst (e.g., "rw/llt").
seasonalOptions	seasonal models to select amongst (e.g., "none/differentt").
irregularOptions	irregular models to select amongst (e.g., "none/arma(0,1)").

## Details

UC is a function for modelling and forecasting univariate time series according to Unobserved Components models (UC). It sets up the model with a number of control variables that govern the way the rest of functions in the package work. It also estimates the model parameters by Maximum Likelihood, forecasts the data, performs smoothing, estimates model disturbances, estimates components and shows statistical diagnostics. Standard methods applicable to UComp objects are print, summary, plot, fitted, residuals, logLik, AIC, BIC, coef, predict, tsdiag.

**Value**

An object of class `UComp`. It is a list with fields including all the inputs and the fields listed below as outputs. All the functions in this package fill in part of the fields of any `UComp` object as specified in what follows (function `UC` fills in all of them at once):

After running `UCforecast` or `UCestim`:

- `p`: Estimated parameters
- `v`: Estimated innovations (white noise in correctly specified models)
- `yFor`: Forecasted values of output
- `yForV`: Forecasted values  $\pm$  one standard error
- `criteria`: Value of criteria for estimated model
- `iter`: Number of iterations in estimation
- `grad`: Gradient at estimated parameters
- `covp`: Covariance matrix of parameters

After running `UCvalidate`:

- `table`: Estimation and validation table

After running `UCcomponents`:

- `comp`: Estimated components in matrix form
- `compV`: Estimated components variance in matrix form

After running `UCfilter`, `UCsmooth` or `UCdisturb`:

- `yFit`: Fitted values of output
- `yFitV`: Variance of fitted values of output
- `a`: State estimates
- `P`: Variance of state estimates
- `aFor`: Forecasts of states
- `PFor`: Forecasts of states variances

After running `UCdisturb`:

- `eta`: State perturbations estimates
- `eps`: Observed perturbations estimates

**Author(s)**

Diego J. Pedregal

**See Also**

[UC](#), [UCvalidate](#), [UCfilter](#), [UCsmooth](#), [UCdisturb](#), [UCcomponents](#), [UChp](#)

**Examples**

```
y <- log(AirPassengers)
m1 <- UC(y)
m1 <- UC(y, model = "l1t/different/arma(0,0)")
```

---

UCcommand

*UCcommand*

---

**Description**

Auxiliar function for UC modeling

**Usage**

```
UCcommand(command, sys)
```

**Arguments**

command	Command to execute: "forecast", "validate", "filter", "smooth", "disturb", "components", "all"
sys	A UComp object created with UC

**Value**

The input UComp object with the appropriate fields filled in

**Author(s)**

Diego J. Pedregal

**See Also**

[UC](#), [UCforecast](#), [UCvalidate](#), [UCfilter](#), [UCsmooth](#), [UCcomponents](#), [UCdisturb](#)

**Examples**

```
cycle <- UChp(USgdp)
plot(cycle)
```

---

UCcomponents	<i>UCcomponents</i>
--------------	---------------------

---

**Description**

Estimates unobserved components of UC models Standard methods applicable to UComp objects are print, summary, plot, fitted, residuals, logLik, AIC, BIC, coef, predict, tdiag.

**Usage**

```
UCcomponents(sys)
```

**Arguments**

sys                    an object of type UComp created with UC or UCforecast

**Value**

The same input object with the appropriate fields filled in, in particular:

- comp: Estimated components in matrix form
- compV: Estimated components variance in matrix form

**Author(s)**

Diego J. Pedregal

**See Also**

[UC](#), [UCforecast](#), [UCvalidate](#), [UCfilter](#), [UCsmooth](#), [UCdisturb](#), [UChp](#)

**Examples**

```
m1 <- UC(log(AirPassengers))
m1 <- UCcomponents(m1)
```

---

`UCdisturb`*UCdisturb*

---

**Description**

Runs the Disturbance Smoother for UC models Standard methods applicable to UComp objects are print, summary, plot, fitted, residuals, logLik, AIC, BIC, coef, predict, tsdiag.

**Usage**

```
UCdisturb(sys)
```

**Arguments**

`sys` an object of type UComp created with UC

**Value**

The same input object with the appropriate fields filled in, in particular:

- `yFit`: Fitted values of output
- `yFitV`: Variance of fitted values of output
- `a`: State estimates
- `P`: Variance of state estimates (diagonal of covariance matrices)
- `eta`: State perturbations estimates
- `eps`: Observed perturbations estimates

**Author(s)**

Diego J. Pedregal

**See Also**

[UC](#), [UCforecast](#), [UCvalidate](#), [UCfilter](#), [UCsmooth](#), [UCcomponents](#), [UChp](#)

**Examples**

```
m1 <- UC(log(AirPassengers))
m1 <- UCdisturb(m1)
```

---

 UCestim

*UCestim*


---

**Description**

Estimates and forecasts UC models

**Usage**

UCestim(sys)

**Arguments**

sys                    an object of type UComp created with UC

**Details**

UCestim estimates and forecasts a time series using an UC model. The optimization method is a BFGS quasi-Newton algorithm with a backtracking line search using Armijo conditions. Parameter names in output table are the following:

- Damping: Damping factor for DT trend.
- Level: Variance of level disturbance.
- Slope: Variance of slope disturbance.
- Rho(#): Damping factor of cycle #.
- Period(#): Estimated period of cycle #.
- Var(#): Variance of cycle #.
- Seas(#): Seasonal harmonic with period #.
- Irregular: Variance of irregular component.
- AR(#): AR parameter of lag #.
- MA(#): MA parameter of lag #.
- AO#: Additive outlier in observation #.
- LS#: Level shift outlier in observation #.
- SC#: Slope change outlier in observation #.
- Beta(#): Beta parameter of input #.
- Cnst: Constant.

Standard methods applicable to UComp objects are print, summary, plot, fitted, residuals, logLik, AIC, BIC, coef, predict, tsdiag.



**Value**

The same input object with the appropriate fields filled in, in particular:

- yFit: Fitted values of output
- yFitV: Variance of fitted values of output
- a: State estimates
- P: Variance of state estimates (diagonal of covariance matrices)

**Author(s)**

Diego J. Pedregal

**See Also**

[UC](#), [UCforecast](#), [UCvalidate](#), [UCsmooth](#), [UCdisturb](#), [UCcomponents](#), [UChp](#)

**Examples**

```
m1 <- UC(log(AirPassengers))
m1 <- UCfilter(m1)
```

---

UCforecast

*UCforecast*

---

**Description**

Estimates and forecasts UC general univariate models

**Usage**

```
UCforecast(
  y,
  u = NULL,
  model = "?/none/?/?",
  h = 24,
  lambda = 1,
  outlier = 9999,
  tTest = FALSE,
  criterion = "aic",
  periods = NA,
  verbose = FALSE,
  stepwise = FALSE,
  p0 = -9999.9,
  arma = FALSE,
  TVP = NULL,
  trendOptions = "none/rw/llt/dt",
```

```

seasonalOptions = "none/equal/different",
irregularOptions = "none/arma(0,0)"
)

```

### Arguments

y	a time series to forecast (it may be either a numerical vector or a time series object). This is the only input required. If a vector, the additional input periods should be supplied compulsorily (see below).
u	a matrix of external regressors included only in the observation equation. (it may be either a numerical vector or a time series object). If the output wanted to be forecast, matrix u should contain future values for inputs.
model	the model to estimate. It is a single string indicating the type of model for each component. It allows two formats "trend/seasonal/irregular" or "trend/cycle/seasonal/irregular". The possibilities available for each component are: <ul style="list-style-type: none"> <li>• Trend: ? / none / rw / irw / llt / dt / td;</li> <li>• Seasonal: ? / none / equal / different;</li> <li>• Irregular: ? / none / arma(0, 0) / arma(p, q) - with p and q integer positive orders;</li> <li>• Cycles: ? / none / combination of positive or negative numbers. Positive numbers fix the period of the cycle while negative values estimate the period taking as initial condition the absolute value of the period supplied. Several cycles with positive or negative values are possible and if a question mark is included, the model test for the existence of the cycles specified. The following are valid examples with different meanings: 48, 48?, -48, -48?, 48+60, -48+60, -48-60, 48-60, 48+60?, -48+60?, -48-60?, 48-60?.</li> </ul>
h	forecast horizon. If the model includes inputs h is not used, the length of u is used instead.
lambda	Box-Cox transformation lambda, NULL for automatic estimation
outlier	critical level of outlier tests. If NA it does not carry out any outlier detection (default). A positive value indicates the critical minimum t test for outlier detection in any model during identification. Three types of outliers are identified, namely Additive Outliers (AO), Level Shifts (LS) and Slope Change (SC).
tTest	augmented Dickey Fuller test for unit roots used in stepwise algorithm (TRUE / FALSE). The number of models to search for is reduced, depending on the result of this test.
criterion	information criterion for identification ("aic", "bic" or "aicc").
periods	vector of fundamental period and harmonics required.
verbose	intermediate results shown about progress of estimation (TRUE / FALSE).
stepwise	stepwise identification procedure (TRUE / FALSE).
p0	initial parameter vector for optimisation search.
arma	check for arma models for irregular components (TRUE / FALSE).
TVP	vector of zeros and ones to indicate TVP parameters.
trendOptions	trend models to select amongst (e.g., "rw/llt").



**Author(s)**

Diego J. Pedregal

**See Also**

[UC](#), [UCvalidate](#), [UCfilter](#), [UCsmooth](#), [UCdisturb](#), [UCcomponents](#), [UChp](#)

**Examples**

```
y <- log(AirPassengers)
m1 <- UCforecast(y)
m1 <- UCforecast(y, model = "1lt/equal/arma(0,0)")
```

---

UChp

*UChp*

---

**Description**

Hodrick-Prescott filter estimation

**Usage**

```
UChp(y, lambda = 1600)
```

**Arguments**

y	A time series object
lambda	Smoothing constant (default: 1600)

**Value**

The cycle estimation

**Author(s)**

Diego J. Pedregal

**See Also**

[UC](#), [UCforecast](#), [UCvalidate](#), [UCfilter](#), [UCsmooth](#), [UCcomponents](#), [UCdisturb](#)

**Examples**

```
cycle <- UChp(USgdp)
plot(cycle)
```

---

*UComp**UComp*

---

**Description**

Package for time series modelling and forecasting of times series models inspired on different sources:

**Details**

- Unobserved Components models due to A.C. Harvey (Basic Structural Model: BSM), enhanced with automatic identification tools by Diego J. Pedregal.
- ExponentTial Smoothing by R.J. Hyndman and colaborators.
- ARIMA models by V. Gómez and A. Maravall
- Tobit ETS models by Pedregal, Trapero and Holgado

The package is designed for automatic identification among a wide range of possible models. The models may include exogenous variables. ARMA irregular components and automatic detection of outliers in some instances.

**References**

- Harvey AC (1989). *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press.
- de Jong, P & Penzer, J (1998). Diagnosing Shocks in Time Series, *Journal of the American Statistical Association*, 93, 442, 796-806.
- Pedregal, DJ, & Young, PC (2002). Statistical approaches to modelling and forecasting time series. In M. Clements, & D. Hendry (Eds.), *Companion to economic forecasting* (pp. 69–104). Oxford: Blackwell Publishers.
- Durbin J, Koopman SJ (2012). *Time Series Analysis by State Space Methods*. 38. Oxford University Press.
- Proietti T and Luati A (2013). Maximum likelihood estimation of time series models: the Kalman filter and beyond, in *Handbook of research methods and applications in empirical macroeconomics*, ed. Nigar Hashimzade and Michael Thornton, E. Elgar, UK.
- Hyndman RJ, Koehler AB, Ord JK and Snyder RD (2008), *Forecasting with exponential smoothing, The State Sapce approach*, Berlin, Springer-Verlag.
- Gómez V and Maravall, A (2000), Automatic methods for univariate series. In Peña, D., Tiao, G.C. and Tsay R.S., *A course in time series analysis*. Wiley.
- Trapero JR, Holgado E, Pedregal DJ (2024), Demand forecasting under lost sales stock policies, *International Journal of Forecasting*, 40, 1055-1068.

**Maintainer**

Diego J. Pedregal

**Author(s)**

Diego J. Pedregal

UCsetup

*UCsetup***Description**

Sets up UC general univariate models

**Usage**

```
UCsetup(
  y,
  u = NULL,
  model = "?/none/?/?",
  h = 24,
  lambda = 1,
  outlier = 9999,
  tTest = FALSE,
  criterion = "aic",
  periods = NA,
  verbose = FALSE,
  stepwise = FALSE,
  p0 = -9999.9,
  arma = FALSE,
  TVP = NULL,
  trendOptions = "none/rw/llt/dt",
  seasonalOptions = "none/equal/different",
  irregularOptions = "none/arma(0,0)"
)
```

**Arguments**

- |       |  |
|-------|--|
| y     | a time series to forecast (it may be either a numerical vector or a time series object). This is the only input required. If a vector, the additional input periods should be supplied compulsorily (see below).   |
| u     | a matrix of external regressors included only in the observation equation. (it may be either a numerical vector or a time series object). If the output wanted to be forecast, matrix u should contain future values for inputs.   |
| model | the model to estimate. It is a single string indicating the type of model for each component. It allows two formats "trend/seasonal/irregular" or "trend/cycle/seasonal/irregular". The possibilities available for each component are: <ul style="list-style-type: none"> <li>• Trend: ? / none / rw / irw / llt / dt / td;</li> <li>• Seasonal: ? / none / equal / different;</li> </ul> |

- Irregular: ? / none / arma(0, 0) / arma(p, q) - with p and q integer positive orders;
- Cycles: ? / none / combination of positive or negative numbers. Positive numbers fix the period of the cycle while negative values estimate the period taking as initial condition the absolute value of the period supplied. Several cycles with positive or negative values are possible and if a question mark is included, the model test for the existence of the cycles specified. The following are valid examples with different meanings: 48, 48?, -48, -48?, 48+60, -48+60, -48-60, 48-60, 48+60?, -48+60?, -48-60?, 48-60?.

h	forecast horizon. If the model includes inputs h is not used, the length of u is used instead.
lambda	Box-Cox transformation lambda, NULL for automatic estimation
outlier	critical level of outlier tests. If NA it does not carry out any outlier detection (default). A positive value indicates the critical minimum t test for outlier detection in any model during identification. Three types of outliers are identified, namely Additive Outliers (AO), Level Shifts (LS) and Slope Change (SC).
tTest	augmented Dickey Fuller test for unit roots used in stepwise algorithm (TRUE / FALSE). The number of models to search for is reduced, depending on the result of this test.
criterion	information criterion for identification ("aic", "bic" or "aicc").
periods	vector of fundamental period and harmonics required.
verbose	intermediate results shown about progress of estimation (TRUE / FALSE).
stepwise	stepwise identification procedure (TRUE / FALSE).
p0	initial parameter vector for optimisation search.
arma	check for arma models for irregular components (TRUE / FALSE).
TVP	vector of zeros and ones to indicate TVP parameters.
trendOptions	trend models to select amongst (e.g., "rw/lt").
seasonalOptions	seasonal models to select amongst (e.g., "none/differentt").
irregularOptions	irregular models to select amongst (e.g., "none/arma(0,1)").

## Details

See help of UC.

## Value

An object of class UComp. It is a list with fields including all the inputs and the fields listed below as outputs. All the functions in this package fill in part of the fields of any UComp object as specified in what follows (function UC fills in all of them at once):

After running UCforecast:

- p: Estimated parameters

- v: Estimated innovations (white noise in correctly specified models)
- yFor: Forecasted values of output
- yForV: Variance of forecasts
- criteria: Value of criteria for estimated model
- iter: Number of iterations in estimation
- grad: Gradient at estimated parameters
- covp: Covariance matrix of parameters

After running UCvalidate:

- table: Estimation and validation table

After running UCcomponents:

- comp: Estimated components in matrix form
- compV: Estimated components variance in matrix form

After running UCfilter, UCsmooth or UCdisturb:

- yFit: Fitted values of output
- yFitV: Estimated fitted values variance
- a: State estimates
- P: Variance of state estimates
- aFor: Forecasts of states
- PFor: Forecasts of states variances

After running UCdisturb:

- eta: State perturbations estimates
- eps: Observed perturbations estimates

Standard methods applicable to UComp objects are print, summary, plot, fitted, residuals, logLik, AIC, BIC, coef, predict, tsdiag.

### Author(s)

Diego J. Pedregal

### See Also

[UC](#), [UCforecast](#), [UCvalidate](#), [UCfilter](#), [UCsmooth](#), [UCdisturb](#), [UCcomponents](#), [UChp](#)

### Examples

```
y <- log(AirPassengers)
m1 <- UCsetup(y)
m1 <- UCsetup(y, outlier = 4)
m1 <- UCsetup(y, model = "11t/equal/arma(0,0)")
m1 <- UCsetup(y, model = "?/?/?/?")
m1 <- UCsetup(y, model = "11t/?/equal/?", outlier = 4)
```

---

`UCsmooth`*UCsmooth*

---

**Description**

Runs the Fixed Interval Smoother for UC models. Standard methods applicable to UComp objects are `print`, `summary`, `plot`, `fitted`, `residuals`, `logLik`, `AIC`, `BIC`, `coef`, `predict`, `tsdiag`.

**Usage**

```
UCsmooth(sys)
```

**Arguments**

`sys` an object of type UComp created with UC

**Value**

The same input object with the appropriate fields filled in, in particular:

- `yFit`: Fitted values of output
- `yFitV`: Variance of fitted values of output
- `a`: State estimates
- `P`: Variance of state estimates (diagonal of covariance matrices)

**Author(s)**

Diego J. Pedregal

**See Also**

[UC](#), [UCforecast](#), [UCvalidate](#), [UCfilter](#), [UCdisturb](#), [UCcomponents](#), [UChp](#)

**Examples**

```
m1 <- UC(log(AirPassengers))
m1 <- UCsmooth(m1)
```

UCvalidate

*UCvalidate*

---

**Description**

Shows a table of estimation and diagnostics results for UC models. Equivalent to print or summary. The table shows information in four sections: Firstly, information about the model estimated, the relevant periods of the seasonal component included, and further information about convergence. Secondly, parameters with their names are provided, the asymptotic standard errors, the ratio of the two, and the gradient at the optimum. One asterisk indicates concentrated-out parameters and two asterisks signals parameters constrained during estimation. Thirdly, information criteria and the value of the log-likelihood. Finally, diagnostic statistics about innovations, namely, the Ljung-Box Q test of absence of autocorrelation statistic for several lags, the Jarque-Bera gaussianity test, and a standard ratio of variances test.

**Usage**

```
UCvalidate(sys, printScreen = TRUE)
```

**Arguments**

sys	an object of type UComp created with UC
printScreen	print to screen or just return output table

**Value**

The same input object with the appropriate fields filled in, in particular:

- table: Estimation and validation table

**Author(s)**

Diego J. Pedregal

**See Also**

[UC](#), [UCforecast](#), [UCfilter](#), [UCsmooth](#), [UCdisturb](#), [UCcomponents](#), [UChp](#)

**Examples**

```
m1 <- UC(log(gdp))
m1 <- UCvalidate(m1)
```

---

USgdp

*US GDP*

---

**Description**

Seasonally adjusted quarterly US real gross domestic product (USgdp).

**Usage**

USgdp

**Format**

Time series objects.

Quarterly data from 1962 to 2019

**Value**

No return value, called for side effects

**Source**

USgdp

**Examples**

USgdp

---

varTest

*varTest*

---

**Description**

Ratio of variances test

**Usage**

varTest(y, parts = 1/3)

**Arguments**

y                    a vector, ts or tsibble object

parts                portion of sample to estimate variances

**Value**

Table with test results

**Author(s)**

Diego J. Pedregal

**See Also**

[colMedians](#), [rowMedians](#), [tests](#), [sumStats](#), [gaussTest](#), [ident](#), [cusum](#), [conv](#), [armaFilter](#), [dif](#), [roots](#), [zplane](#), [acft](#), [slide](#), [plotSlide](#), [Accuracy](#), [tsDisplay](#), [size](#)

**Examples**

```
varTest(AirPassengers)
```

---

WEEKforecast

*WEEKforecast*

---

**Description**

Estimates and forecasts weekly time series with monthly and annual seasonal patterns

**Usage**

```
WEEKforecast(
  y,
  u = NULL,
  initial_date,
  h = 24,
  lambda = 1,
  criterion = "aic",
  p0 = -9999.9,
  verbose = FALSE
)
```

**Arguments**

<code>y</code>	A numeric vector, ts object or list containing weekly observations of the dependent variable.
<code>u</code>	Optional matrix or vector of exogenous regressors. If provided, it must cover both the estimation and forecast periods.
<code>initial_date</code>	A character or Date object indicating the calendar start date of the series.
<code>h</code>	Integer indicating the forecast horizon (number of weeks ahead).
<code>lambda</code>	Box–Cox transformation parameter.
<code>criterion</code>	Information criterion used for model selection (AIC, BIC or AICc).
<code>p0</code>	Initial parameter vector for Maximum Likelihood estimation.
<code>verbose</code>	if TRUE, prints estimation details.

**Details**

WEEKforecast estimates and forecasts univariate weekly time series using Dynamic Harmonic Regression (DHR).

The function is designed for weekly data but builds the seasonal structure from underlying daily calendar information. In particular, it incorporates:

- Monthly seasonality via harmonic terms derived from calendar months
- Annual seasonality via multiple harmonic components based on calendar years

Standard methods applicable to UComp objects are available.

**Value**

An object of class UComp. It is a list containing the model specification, estimated parameters, and forecasting results.

**Author(s)**

Diego J. Pedregal

**See Also**

[UC](#), [UCforecast](#), [UCfilter](#), [UCsmooth](#), [UCdisturb](#), [UCcomponents](#)

**Examples**

```
y <- rnorm(400)
m <- WEEKforecast(y, initial_date = "2015-01-01", h = 12)
```

---

zplane

*zplane*

---

**Description**

Real-imaginary plane to show roots of digital filters (ARMA)

**Usage**

```
zplane(MApoly = 1, ARpoly = 1)
```

**Arguments**

MApoly	coefficients of numerator polynomial in descending order
ARpoly	coefficients of denominator polynomial in descending order

**Details**

Shows the real-imaginary plane to show zeros (roots of numerator or MA polynomial) and poles (roots of denominator of AR polynomial). Unit roots and real vs imaginary roots can be seen by eye

**Value**

No return value, called for side effects

**Author(s)**

Diego J. Pedregal

**See Also**

[colMedians](#), [rowMedians](#), [tests](#), [sumStats](#), [gaussTest](#), [ident](#), [cusum](#), [varTest](#), [conv](#), [armaFilter](#), [dif](#), [roots](#), [acft](#), [slide](#), [plotSlide](#), [Accuracy](#), [tsDisplay](#), [size](#)

**Examples**

```
zplane(c(1, -2, 1), c(1, -0.8))
```

# Index

- \* **datasets**
  - airpas, 6
  - ch4, 17
  - gdp, 32
  - ipi, 36
  - OECDgdp, 37
  - sales, 43
  - USgdp, 77
- Accuracy, 3, 4, 14, 18, 19, 21, 32, 34, 39, 42–45, 47, 48, 59, 78, 80
- acft, 4, 4, 14, 18, 19, 21, 32, 34, 39, 42–45, 47, 48, 59, 78, 80
- AIC.UComp, 5
- airpas, 6
- ARIMA, 6, 10, 12, 13
- ARIMAestim, 8
- ARIMAforecast, 8, 9, 12, 13
- ARIMAsSetup, 11
- ARIMAvAlidate, 8, 10, 12, 13, 13
- arma2tsi, 13
- armaFilter, 4, 14, 18, 19, 21, 32, 34, 39, 42–45, 47, 48, 59, 78, 80
- auxInvBoxCox, 15
- BIC.UComp, 15
- box.cox, 16, 35
- ch4, 17
- colMedians, 4, 14, 17, 18, 19, 21, 32, 34, 39, 42–45, 47, 48, 59, 78, 80
- conv, 4, 14, 18, 18, 19, 21, 32, 34, 39, 42–45, 47, 48, 59, 78, 80
- cusum, 4, 14, 18, 19, 21, 32, 34, 39, 42–45, 47, 48, 59, 78, 80
- DAYforecast, 20
- dif, 4, 14, 18, 19, 21, 32, 34, 39, 42–45, 47, 48, 59, 78, 80
- ETS, 22, 24, 25, 28, 30, 31
- ETScomponents, 24, 24, 25, 28, 30, 31
- ETSestim, 25
- ETSforecast, 24, 25, 26, 30, 31
- ETSsetup, 28
- ETSvAlidate, 24, 25, 28, 30, 30, 31
- extract, 31
- gaussTest, 4, 14, 18, 19, 21, 32, 34, 39, 42–45, 47, 48, 59, 78, 80
- gdp, 32
- getp0, 33
- ident, 4, 14, 18, 19, 21, 32, 34, 39, 42–45, 47, 48, 59, 78, 80
- inv.box.cox, 16, 35
- invBoxCox, 35
- ipi, 36
- OECDgdp, 37
- plotAcfPacf, 37
- plotBar, 38
- plotSlide, 4, 14, 18, 19, 21, 32, 34, 39, 42–45, 47, 48, 59, 78, 80
- plus\_one, 40
- predict.UComp, 40
- removeNaNs, 41
- roots, 4, 14, 18, 19, 21, 32, 34, 39, 42, 43–45, 47, 48, 59, 78, 80
- rowMedians, 4, 14, 18, 19, 21, 32, 34, 39, 42, 42, 44, 45, 47, 48, 59, 78, 80
- sales, 43
- size, 4, 14, 18, 19, 21, 32, 34, 39, 42, 43, 44, 45, 47, 48, 59, 78, 80
- slide, 4, 14, 18, 19, 21, 32, 34, 39, 42–44, 44, 47, 48, 59, 78, 80
- slideAux, 46
- sumStats, 4, 14, 18, 19, 21, 32, 34, 39, 42–45, 47, 48, 59, 78, 80

- tests, [4](#), [14](#), [18](#), [19](#), [21](#), [32](#), [34](#), [39](#), [42–45](#), [47](#),  
[48](#), [59](#), [78](#), [80](#)
- TETS, [49](#), [51](#), [52](#), [55](#), [57](#), [58](#)
- TETScomponents, [51](#), [51](#), [52](#), [55](#), [57](#), [58](#)
- TETSestim, [52](#)
- TETSforecast, [51](#), [52](#), [53](#), [57](#), [58](#)
- TETSsetup, [55](#)
- TETSvalidate, [51](#), [52](#), [55](#), [57](#), [57](#), [58](#)
- tsDisplay, [4](#), [14](#), [18](#), [19](#), [21](#), [32](#), [34](#), [39](#),  
[42–45](#), [47](#), [48](#), [58](#), [78](#), [80](#)
  
- UC, [5](#), [16](#), [21](#), [33](#), [35](#), [41](#), [59](#), [61–64](#), [66](#), [67](#), [70](#),  
[74–76](#), [79](#)
- UCcommand, [62](#)
- UCcomponents, [5](#), [16](#), [21](#), [33](#), [35](#), [41](#), [61](#), [62](#),  
[63](#), [64](#), [66](#), [67](#), [70](#), [74–76](#), [79](#)
- UCdisturb, [5](#), [16](#), [21](#), [33](#), [35](#), [41](#), [61–63](#), [64](#),  
[66](#), [67](#), [70](#), [74–76](#), [79](#)
- UCestim, [65](#)
- UCfilter, [5](#), [16](#), [21](#), [33](#), [35](#), [41](#), [61–64](#), [66](#), [66](#),  
[70](#), [74–76](#), [79](#)
- UCforecast, [5](#), [16](#), [21](#), [33](#), [35](#), [41](#), [62–64](#), [66](#),  
[67](#), [67](#), [70](#), [74–76](#), [79](#)
- UChp, [33](#), [61](#), [63](#), [64](#), [66](#), [67](#), [70](#), [70](#), [74–76](#)
- UComp, [71](#)
- UComp-package (UComp), [71](#)
- UCsetup, [72](#)
- UCsmooth, [5](#), [16](#), [21](#), [33](#), [35](#), [41](#), [61–64](#), [66](#), [67](#),  
[70](#), [74](#), [75](#), [76](#), [79](#)
- UCvalidate, [5](#), [16](#), [33](#), [35](#), [41](#), [61–64](#), [66](#), [67](#),  
[70](#), [74](#), [75](#), [76](#)
- USgdp, [77](#)
  
- varTest, [4](#), [14](#), [18](#), [19](#), [21](#), [32](#), [34](#), [39](#), [42–45](#),  
[47](#), [48](#), [59](#), [77](#), [80](#)
  
- WEEKforecast, [78](#)
  
- zplane, [4](#), [14](#), [18](#), [19](#), [21](#), [32](#), [34](#), [39](#), [42–45](#),  
[47](#), [48](#), [59](#), [78](#), [79](#)