

# Package: UComp (via r-universe)

September 26, 2024

**Type** Package

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

**Version** 5.0.4

**Date** 2024-07-25

**Author** Diego J. Pedregal [aut, cre]  
(<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** 7.3.1

**NeedsCompilation** yes

**Repository** CRAN

**Date/Publication** 2024-07-26 08:20:05 UTC

## Contents

|                         |    |
|-------------------------|----|
| Accuracy . . . . .      | 4  |
| acft . . . . .          | 5  |
| AIC.UComp . . . . .     | 6  |
| airpas . . . . .        | 7  |
| ARIMA . . . . .         | 7  |
| ARIMAestim . . . . .    | 9  |
| ARIMAmodel . . . . .    | 10 |
| ARIMAsSetup . . . . .   | 12 |
| ARIMAvalidate . . . . . | 13 |
| arma2tsi . . . . .      | 14 |
| armaFilter . . . . .    | 15 |
| auxInvBoxCox . . . . .  | 15 |
| BIC.UComp . . . . .     | 16 |
| box.cox . . . . .       | 17 |
| ch4 . . . . .           | 17 |
| colMedians . . . . .    | 18 |
| conv . . . . .          | 19 |
| cusum . . . . .         | 19 |
| dif . . . . .           | 20 |
| ETS . . . . .           | 21 |
| ETScomponents . . . . . | 23 |
| ETSeestim . . . . .     | 24 |
| ETSmodel . . . . .      | 25 |
| ETSsetup . . . . .      | 27 |
| ETSvalidate . . . . .   | 29 |
| extract . . . . .       | 30 |
| fitted.ETS . . . . .    | 30 |
| gaussTest . . . . .     | 31 |
| gdp . . . . .           | 32 |
| getp0 . . . . .         | 33 |
| ident . . . . .         | 34 |
| inv.box.cox . . . . .   | 35 |
| invBoxCox . . . . .     | 35 |
| ipi . . . . .           | 36 |
| modelUC2arma . . . . .  | 37 |
| modelUC2PTS . . . . .   | 37 |
| OECDgdp . . . . .       | 38 |
| plot.ARIMA . . . . .    | 38 |
| plot.ETS . . . . .      | 39 |
| plotAcfPacf . . . . .   | 40 |
| plotBar . . . . .       | 41 |
| plotSlide . . . . .     | 41 |
| plus_one . . . . .      | 42 |
| predict.UComp . . . . . | 43 |
| print.ARIMA . . . . .   | 44 |
| PTS . . . . .           | 45 |

|                          |    |
|--------------------------|----|
| PTS2modelUC . . . . .    | 47 |
| PTScomponents . . . . .  | 48 |
| PTSeestim . . . . .      | 49 |
| PTSmodel . . . . .       | 50 |
| PTSsetup . . . . .       | 52 |
| PTSvalidate . . . . .    | 54 |
| removeNaNs . . . . .     | 55 |
| residuals.ETS . . . . .  | 55 |
| roots . . . . .          | 56 |
| rowMedians . . . . .     | 57 |
| sales . . . . .          | 58 |
| size . . . . .           | 58 |
| slide . . . . .          | 59 |
| slideAux . . . . .       | 60 |
| summary.ARIMA . . . . .  | 61 |
| summary.ETS . . . . .    | 62 |
| summary.PTS . . . . .    | 63 |
| summary.TETS . . . . .   | 64 |
| sumStats . . . . .       | 65 |
| tests . . . . .          | 66 |
| TETS . . . . .           | 67 |
| TETScomponents . . . . . | 69 |
| TETSeestim . . . . .     | 70 |
| TETSmodel . . . . .      | 71 |
| TETSsetup . . . . .      | 73 |
| TETSvalidate . . . . .   | 75 |
| tsDisplay . . . . .      | 76 |
| UC . . . . .             | 77 |
| UCcomponents . . . . .   | 80 |
| UCdisturb . . . . .      | 81 |
| UCestim . . . . .        | 82 |
| UCfilter . . . . .       | 83 |
| UChp . . . . .           | 84 |
| UCmodel . . . . .        | 85 |
| UComp . . . . .          | 88 |
| UCsetup . . . . .        | 89 |
| UCsmooth . . . . .       | 92 |
| UCvalidate . . . . .     | 93 |
| USgdp . . . . .          | 94 |
| varTest . . . . .        | 94 |
| zplane . . . . .         | 95 |

---

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

---

**Description**

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

**Usage**

Accuracy(py, y, s = frequency(y), collectFun = mean)

**Arguments**

- py                    matrix of forecasts (h x nMethods x nForecasts)
- y                    a matrix of actual values (n x nForecasts)
- s                    seasonal period, number of observations per year
- collectFun          aggregation function (mean, median, etc.)

**Value**

Table of 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 | <i>acft</i> |
|------|-------------|

---

**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.

## Author(s)

Diego J. Pedregal

## See Also

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

## Examples

```
## Not run:
y <- log(AirPassengers)
m1 <- UCmodel(y, model = "l1t/equal/arma(0,0)")
AIC(m1)

## End(Not run)
```

---

|        |                               |
|--------|-------------------------------|
| 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>

**Examples**

```
## Not run:
airpas

## End(Not run)
```

---

|       |              |
|-------|--------------|
| 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**

|                        |   |
|------------------------|---|
| <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. A vector <code>c(p,d,q,P,D,Q)</code> containing the model orders of an <code>ARIMA(p,d,q)x(P,D,Q)_s</code> model. A constant may be estimated with the <code>cnst</code> input. Use a <code>NULL</code> to automatically identify the ARIMA model. |
| <code>cnst</code>      | flag to include a constant in the model ( <code>TRUE/FALSE/NULL</code> ). Use <code>NULL</code> to estimate   |
| <code>s</code>         | seasonal period of time series (1 for annual, 4 for quarterly, ...)   |
| <code>criterion</code> | information criterion for identification stage (" <code>aic</code> ", " <code>bic</code> ", " <code>aicc</code> ")  |
| <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>verbose</code>   | intermediate estimation output ( <code>TRUE / FALSE</code> )  |
| <code>lambda</code>    | Box-Cox lambda parameter ( <code>NULL</code> : estimate)  |
| <code>maxOrders</code> | a vector <code>c(p,d,q,P,D,Q)</code> 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**

See help of `ARIMAmodel`.

**Value**

An object of class `ARIMA`. See `ARIMAmodel`.

**Author(s)**

Diego J. Pedregal

**See Also**

[ARIMAmodel](#), [ARIMAvalidate](#),

**Examples**

```
## Not run:
y <- log(AirPassengers)
m1 <- ARIMA(y)
m1 <- ARIMA(y, lambda = NULL)

## End(Not run)
```



---

**ARIMAestim***ARIMAestim*

---

**Description**

Estimates and forecasts ARIMA models

**Usage**

```
ARIMAestim(m)
```

**Arguments**

**m** an object of type ARIMA created with ARIMAmodel

**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:

|               |   |
|---------------|---|
| <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**

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

**Examples**

```
## Not run:  
m1 <- ARIMAsetup(log(gdp))  
m1 <- ARIMAestim(m1)  
  
## End(Not run)
```

ARIMAmode1

*ARIMAmode1***Description**

Estimates and forecasts ARIMA general univariate models

**Usage**

```
ARIMAmode1(
  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

`ARIMAmode1` 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 `ARIMAmode1` 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

```
## Not run:
y <- log(AirPassengers)
m1 <- ARIMAmode1(y)
m1 <- ARIMAmode1(y, lambda = NULL)

## End(Not run)
```

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 ARIMAmodel.

**Value**

An object of class ARIMA. See ARIMAmodel.

**Author(s)**

Diego J. Pedregal

**See Also**

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

**Examples**

```
## Not run:
y <- log(AirPassengers)
m1 <- ARIMAsetup(y)
m1 <- ARIMAsetup(y, lambda = NULL)

## End(Not run)
```

---

ARIMAvalidate

*ARIMAvalidate*


---

**Description**

Shows a table of estimation and diagnostics results for ARIMA models

**Usage**

```
ARIMAvalidate(m)
```

**Arguments**

**m** an object of type ARIMA created with ARIMAmodel

**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](#), [ARIMAmode1](#), [ARIMAvalidate](#),

**Examples**

```
## Not run:
m1 <- ARIMAmode1(log(gdp))
m1 <- ARIMAvalidate(m1)

## End(Not run)
```

---

|          |                 |
|----------|-----------------|
| 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                                     |

**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`*auxInvBoxCox*

---

**Description**

Inverse of Box-Cox transformation

**Usage**

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

**Arguments**

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

**Author(s)**

Diego J. Pedregal

---

BIC.UComp

*BIC.UComp*

---

**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.

**Author(s)**

Diego J. Pedregal

**See Also**

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

**Examples**

```
## Not run:
y <- log(AirPassengers)
m1 <- UCmodel(y, model = "11t/equal/arma(0,0)")
BIC(m1)

## End(Not run)
```



---

`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. |

**Author(s)**

Diego J. Pedregal

**See Also**

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

**Examples**

```
## Not run:  
y <- box.cox(AirPassengers, 0.5)  
plot(y)  
  
## End(Not run)
```

---

`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

**Source**

CH4 data

**Examples**

```
## Not run:  
ch4  
  
## End(Not run)
```

---

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

**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 | <i>conv</i> |
|------|-------------|

---

**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 | <i>cusum</i> |
|-------|--------------|

---

**Description**

Cusum and cusumsq tests

**Usage**

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

**Arguments**

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

**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)
```

---

dif

*dif*

---

**Description**

Discrete differencing of time series

**Usage**

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

**Arguments**

|      |                                 |
|------|---------------------------------|
| y    | a vector, ts or tsibble 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**

|       |   |
|-------|---|
| 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 lenght of u is used instead.  |

|                           |   |
|---------------------------|---|
| <code>criterion</code>    | information criterion for identification ("aic", "bic" or "aicc").  |
| <code>lambda</code>       | Box-Cox lambda parameter (NULL: estimate)   |
| <code>armaIdent</code>    | check for arma models for error component (TRUE / FALSE).   |
| <code>identAll</code>     | run all models to identify the best one (TRUE / FALSE)  |
| <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) 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 `ETSmodel`.

### Value

An object of class ETS. See `ETSmodel`.

### Author(s)

Diego J. Pedregal

### See Also

[ETSmodel](#), [ETSvalidate](#), [ETScomponents](#), [ETSestim](#)

### Examples

```
## Not run:
y <- log(AirPassengers)
m1 <- ETS(y)
m1 <- ETS(y, model = "MAM")

## End(Not run)
```

---

|               |                      |
|---------------|----------------------|
| ETScomponents | <i>ETScomponents</i> |
|---------------|----------------------|

---

## Description

Estimates components of ETS models

## Usage

```
ETScomponents(m)
```

## Arguments

`m` an object of type ETS created with `ETSmodel`

## 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](#), [ETSmodel](#), [ETSvalidate](#), [ETSestim](#)

## Examples

```
## Not run:
m1 <- ETS(log(gdp))
m1 <- ETScomponents(m1)

## End(Not run)
```

---

**ETSestim***ETSestim*

---

**Description**

Estimates and forecasts ETS models

**Usage**

```
ETSestim(m)
```

**Arguments**

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

**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](#), [ETSmodel](#), [ETSvalidate](#), [ETScomponents](#)

**Examples**

```
## Not run:  
m1 <- ETSsetup(log(gdp))  
m1 <- ETSestim(m1)  
  
## End(Not run)
```



ETSmodel

*ETSmodel***Description**

Estimates and forecasts ETS general univariate models

**Usage**

```
ETSmodel(
  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 lenght 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

ETSmodel 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 ETSmodel or ETSestim:

|          |   |
|----------|---|
| 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 |
|------|-------------------------------------|

**Author(s)**

Diego J. Pedregal

**See Also**[ETS](#), [ETSvalidate](#), [ETScomponents](#), [ETSeestim](#)**Examples**

```
## Not run:
y <- log(AirPassengers)
m1 <- ETSmodel(y)
m1 <- ETSmodel(y, model = "A?A")

## End(Not run)
```

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  |
| 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 ETSmodel.

**Value**

An object of class ETS. See ETSmodel.

**Author(s)**

Diego J. Pedregal

**See Also**

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

**Examples**

```
## Not run:
y <- log(AirPassengers)
m1 <- ETSsetup(y)
m1 <- ETSsetup(y, model = "??")
m1 <- ETSsetup(y, model = "AA")

## End(Not run)
```

---

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 ETSmodel

**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](#), [ETSmodel](#), [ETSvalidate](#), [ETScomponents](#)

**Examples**

```
## Not run:
m1 <- ETSmodel(log(gdp))
m1 <- ETSvalidate(m1)

## End(Not run)
```

---

|         |                |
|---------|----------------|
| extract | <i>extract</i> |
|---------|----------------|

---

**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

**Author(s)**

Diego J. Pedregal

---

|            |                   |
|------------|-------------------|
| fitted.ETS | <i>fitted.ETS</i> |
|------------|-------------------|

---

**Description**

Fitted output values of ETS object

Fitted output values of PTS object

Fitted output values of TETS object

**Usage**

```
## S3 method for class 'ETS'
fitted(object, ...)
```

```
## S3 method for class 'PTS'
fitted(object, ...)
```

```
## S3 method for class 'TETS'
fitted(object, ...)
```

**Arguments**

object            Object of class “TETS”.  
 ...              Additional inputs to function.

**Details**

See help of ETS.  
 See help of PTS.  
 See help of TETS.

**Author(s)**

Diego J. Pedregal

**See Also**

[ETS](#), [ETSmodel](#), [ETSvalidate](#), [ETScomponents](#), [ETSestim](#)  
[PTS](#), [PTSmodel](#), [PTSvalidate](#), [PTScomponents](#), [PTSestim](#)  
[TETS](#), [TETSmodel](#), [TETSvalidate](#), [TETScomponents](#), [TETSestim](#)

**Examples**

```
## Not run:
m1 <- ETSmodel(log(gdp))
fitted(m1)

## End(Not run)
## Not run:
m1 <- PTSmodel(log(AirPassengers))
fitted(m1)

## End(Not run)
## Not run:
m1 <- TETSmodel(log(gdp))
fitted(m1)

## End(Not run)
```

---

gaussTest

*gaussTest*


---

**Description**

Gaussianity tests

**Usage**

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

**Arguments**

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

**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>

**Examples**

```
## Not run:  
gdp
```

```
## End(Not run)
```



---

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 p0 to UC or UCmodel 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 p0 of an object of class UComp to use as input to [UC](#), [UCmodel](#) or [UCsetup](#).

## 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 <- UCmodel(log(AirPassengers), model = "l1t/equal/arma(0,0)", p0 = p0)

## End(Not run)
```

---

|       |              |
|-------|--------------|
| ident | <i>ident</i> |
|-------|--------------|

---

## Description

Autocorrelation functions of a time series

## Usage

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

## 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 |
| runFromTests | internal check                                     |

## Value

A vector with all the dimensions

## 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**

|                     |  |
|---------------------|--|
| <code>x</code>      | Transformed time series object.              |
| <code>lambda</code> | Lambda parameter used for Box-Cox transform. |

**Author(s)**

Diego J. Pedregal

**See Also**

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

**Examples**

```
## Not run:  
y <- inv.box.cox(box.cox(AirPassengers, 0.5), 0.5)  
plot(y)  
  
## End(Not run)
```

---

`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 |

**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>

**Examples**

```
## Not run:
```

```
  ipi
```

```
## End(Not run)
```

---

`modelUC2arma`*modelUC2arma*

---

**Description**

Extracts arma part of modelUC model

**Usage**

```
modelUC2arma(model)
```

**Arguments**

`model`            a UC model

**Value**

arma orders

**Author(s)**

Diego J. Pedregal

---

`modelUC2PTS`*modelUC2PTS*

---

**Description**

Translates modelUC to model PTS

**Usage**

```
modelUC2PTS(modelUC)
```

**Arguments**

`modelUC`            a UC model

**Value**

a PTS model

**Author(s)**

Diego J. Pedregal

---

OECDgdp

*OECD GDP*


---

**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>

**Examples**

```
## Not run:
```

```
OECDgdp
```

```
## End(Not run)
```

---

plot.ARIMA

*plot.ARIMA*


---

**Description**

Plot zplane of ARIMA object

**Usage**

```
## S3 method for class 'ARIMA'
```

```
plot(x, ...)
```

**Arguments**

x                      Object of class “ARIMA”.

...                    Additional inputs to function.

**Details**

See help of ARIMA.

**Author(s)**

Diego J. Pedregal

**See Also**[ARIMA](#), [ARIMAmode1](#), [ARIMAvalidate](#),**Examples**

```
## Not run:
m1 <- ARIMAmode1(log(gdp))
plot(m1)

## End(Not run)
```

plot.ETS

*plot.ETS***Description**

Plot components of ETS object

Plot components of PTS object

Plot components of TETS object

**Usage**

```
## S3 method for class 'ETS'
plot(x, ...)

## S3 method for class 'PTS'
plot(x, ...)

## S3 method for class 'TETS'
plot(x, ...)
```

**Arguments**

x                      Object of class “TETS”.

...                    Additional inputs to function.

**Details**

See help of ETS.

See help of PTS.

See help of TETS.

**Author(s)**

Diego J. Pedregal

**See Also**

[ETS](#), [ETSmodel](#), [ETSvalidate](#), [ETScomponents](#), [ETSeestim](#)  
[PTS](#), [PTSmodel](#), [PTSvalidate](#), [PTScomponents](#), [PTSeestim](#)  
[TETS](#), [TETSmodel](#), [TETSvalidate](#), [TETScomponents](#), [TETSeestim](#)

**Examples**

```
## Not run:
m1 <- ETSmodel(log(gdp))
plot(m1)

## End(Not run)
## Not run:
m1 <- PTS(log(AirPassengers))
plot(m1)

## End(Not run)
## Not run:
m1 <- TETSmodel(log(gdp))
plot(m1)

## End(Not run)
```

---

|             |                    |
|-------------|--------------------|
| 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 |



**Author(s)**

Diego J. Pedregal

---

plotBar*plotBar*

---

**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*plotSlide*

---

**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 or matrix 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**

Results

**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

*plus\_one*

---

**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**

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

**Author(s)**

Diego J. Pedregal

**See Also**

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

**Examples**

```
## Not run:
y <- log(AirPassengers)
m1 <- UCmodel(y, model = "l1t/eq/arma(0,0)")
f1 <- predict(m1)

## End(Not run)
```

---

print.ARIMA

---

*print.ARIMA*


---

### Description

Prints an ARIMA object  
 Prints an ETS object  
 Prints a PTS object  
 Prints a TOBIT TETS object

### Usage

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

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

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

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

### Arguments

|     |  |
|-----|--|
| x   | Object of class “TETS”.                              |
| ... | Additional inputs to handle the way to print output. |

### Details

See help of ARIMA.  
 See help of ETS.  
 See help of PTS.  
 See help of TETS.

### Author(s)

Diego J. Pedregal

### See Also

[ARIMA](#), [ARIMAmode1](#), [ARIMAvaiidate](#),  
[ETS](#), [ETSmodel](#), [ETSvaidate](#), [ETScomponents](#), [ETSestim](#)  
[PTS](#), [PTSmodel](#), [PTSvaidate](#), [PTScomponents](#), [PTSestim](#)  
[TETS](#), [TETSmodel](#), [TETSvaidate](#), [TETScomponents](#), [TETSestim](#)

**Examples**

```
## Not run:
m1 <- ARIMAmoel(log(gdp))
print(m1)

## End(Not run)
## Not run:
m1 <- ETsmoel(log(gdp))
print(m1)

## End(Not run)
## Not run:
m1 <- PTsmoel(log(AirPassengers))
print(m1)

## End(Not run)
## Not run:
m1 <- TETsmoel(log(gdp))
print(m1)

## End(Not run)
```

---

PTS

---

*PTS*


---

**Description**

Estimates, forecasts and smooth PTS general univariate models

**Usage**

```
PTS(
  y,
  u = NULL,
  model = "???",
  s = frequency(y),
  h = 2 * s,
  criterion = "aicc",
  lambda = 1,
  armaIdent = FALSE,
  verbose = 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).

|                        |   |
|------------------------|---|
| <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: ? / N / A</li> <li>• Trend: ? / N / A / Ad / L</li> <li>• Seasonal: ? / N / A / D (trigonometric with different variances)</li> </ul> |
| <code>s</code>         | 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>lambda</code>    | Box-Cox lambda parameter (NULL: estimate)   |
| <code>armaIdent</code> | check for arma models for error component (TRUE / FALSE).   |
| <code>verbose</code>   | intermediate estimation output (TRUE / FALSE)   |

### Details

PTS is a function for modelling and forecasting univariate time series according to Power-Trend-Seasonal (PTS). 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 and forecasts the data. Standard methods applicable to UComp objects are `print`, `summary`, `plot`, `fitted`, `residuals`, `logLik`, `AIC`, `BIC`, `coef`, `predict`, `tsdiag`.

### Value

An object of class PTS. 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 PTS object as specified in what follows (function PTS fills in all of them at once):

After running `PTSmodel` or `PTSestim`:

- `p0`: Initial values for parameter search
- `p`: Estimated parameters
- `lambda`: Estimated Box-Cox lambda parameter
- `v`: Estimated innovations (white noise in correctly specified models)
- `yFor`: Forecasted values of output
- `yForV`: Variance of forecasted values of output

After running `PTSvalidate`:

- `table`: Estimation and validation table

After running `PTScomponents`:

- `comp`: Estimated components in matrix form

**Author(s)**

Diego J. Pedregal

**See Also**

[PTSmodel](#), [PTSsetup](#), [PTSvalidate](#), [PTScomponents](#), [PTSeestim](#)

**Examples**

```
## Not run:  
m1 <- PTS(log(AirPassengers))  
  
## End(Not run)
```

---

PTS2modelUC

*PTS2modelUC*

---

**Description**

Translates PTS model to UC model

**Usage**

```
PTS2modelUC(model, armaOrders = c(0, 0))
```

**Arguments**

|            |                            |
|------------|----------------------------|
| model      | a PTS model                |
| armaOrders | arma orders of noise model |

**Value**

a UC model

**Author(s)**

Diego J. Pedregal

---

|               |                      |
|---------------|----------------------|
| PTScomponents | <i>PTScomponents</i> |
|---------------|----------------------|

---

**Description**

Estimates components of PTS models

**Usage**

```
PTScomponents(m)
```

**Arguments**

`m` an object of type PTS created with `PTSmol`

**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**

[PTSmol](#), [PTSsetup](#), [PTSe](#), [PTStest](#), [PTStest](#), [PTStest](#)

**Examples**

```
## Not run:  
m1 <- PTS(log(AirPassengers))  
m1 <- PTScomponents(m1)  
  
## End(Not run)
```



---

**PTSeestim***PTSeestim*

---

**Description**

Estimates and forecasts PTS models

**Usage**

PTSeestim(m)

**Arguments**

m                      an object of type PTS created with PTSmodel

**Details**

PTSeestim estimates and forecasts a time series using an a PTS model

**Value**

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

- p0: Initial values for parameter search
- p: Estimated parameters
- lambda: Estimated Box-Cox lambda parameter
- v: Estimated innovations (white noise in correctly specified models)
- yFor: Forecasted values of output
- yForV: Variance of forecasted values of output

**Author(s)**

Diego J. Pedregal

**See Also**

[PTSmodel](#), [PTSsetup](#), [PTSvalidate](#), [PTScomponents](#), [PTS](#)

**Examples**

```
## Not run:
m1 <- PTSsetup(log(AirPassengers))
m1 <- PTSeestim(m1)

## End(Not run)
```

---

PTsmodel

*PTsmodel*


---

## Description

Estimates and forecasts PTS general univariate models

## Usage

```
PTsmodel(
  y,
  u = NULL,
  model = "???",
  s = frequency(y),
  h = 2 * s,
  criterion = "aicc",
  lambda = 1,
  armaIdent = FALSE,
  verbose = 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. 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: ? / N / A</li> <li>• Trend: ? / N / A / Ad / L</li> <li>• Seasonal: ? / N / A / D (trigonometric with different variances)</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).   |
| verbose   | intermediate estimation output (TRUE / FALSE)   |

## Details

PTsmodel is a function for modelling and forecasting univariate time series according to Power-Trend-Seasonal (PTS). 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 and forecasts the data. Standard methods applicable to UComp objects are print, summary, plot, fitted, residuals, logLik, AIC, BIC, coef, predict, tsdiag.

## Value

An object of class PTS. 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 PTS object as specified in what follows (function PTS fills in all of them at once):

After running PTsmodel or PTsestim:

- p0: Initial values for parameter search
- p: Estimated parameters
- lambda: Estimated Box-Cox lambda parameter
- v: Estimated innovations (white noise in correctly specified models)
- yFor: Forecasted values of output
- yForV: Variance of forecasted values of output

After running PTSvalidate:

- table: Estimation and validation table

After running PTScomponents:

- comp: Estimated components in matrix form

## Author(s)

Diego J. Pedregal

## See Also

[PTS](#), [PTSsetup](#), [PTSvalidate](#), [PTScomponents](#), [PTsestim](#)

## Examples

```
## Not run:  
m1 <- PTsmodel(log(AirPassengers))  
  
## End(Not run)
```

PTSsetup

*PTSsetup***Description**

Run up PTS general univariate MSOE models

**Usage**

```
PTSsetup(
  y,
  u = NULL,
  model = "???",
  s = frequency(y),
  h = 2 * s,
  criterion = "aic",
  lambda = 1,
  armaIdent = FALSE,
  verbose = 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. 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: ? / N / A</li> <li>• Trend: ? / N / A / Ad / L</li> <li>• Seasonal: ? / N / A / D (trigonometric with different variances)</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 lenght 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).   |
| verbose   | intermediate estimation output (TRUE / FALSE)   |

**Details**

See help of PTS.

**Value**

An object of class PTS. 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 PTS object as specified in what follows (function PTS fills in all of them at once):

After running PTSmodel or PTSeestim:

- p0: Initial values for parameter search
- p: Estimated parameters
- lambda: Estimated Box-Cox lambda parameter
- v: Estimated innovations (white noise in correctly specified models)
- yFor: Forecasted values of output
- yForV: Variance of forecasted values of output

After running PTSvalidate:

- table: Estimation and validation table

After running PTScomponents:

- comp: Estimated components in matrix form

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

**Author(s)**

Diego J. Pedregal

**See Also**

[PTS](#), [PTSmodel](#), [PTSvalidate](#), [PTScomponents](#), [PTSeestim](#)

**Examples**

```
## Not run:  
m1 <- PTSsetup(log(AirPassengers))  
  
## End(Not run)
```

PTValidate

*PTValidate*

---

**Description**

Shows a table of estimation and diagnostics results for PTS models

**Usage**

```
PTValidate(m, verbose = TRUE)
```

**Arguments**

|                      |   |
|----------------------|---|
| <code>m</code>       | an object of type PTS created with <code>PTModel</code> |
| <code>verbose</code> | verbose mode TRUE/FALSE                                 |

**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**

[PTModel](#), [PTSetup](#), [PTSEstim](#), [PTComponents](#), [PTS](#)

**Examples**

```
## Not run:  
m1 <- PTModel(log(AirPassengers))  
m1 <- PTValidate(m1)  
  
## End(Not run)
```

---

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

---

residuals.ETS

*residuals.ETS*


---

**Description**

Residuals of ETS object

Residuals of PTS object

Residuals of TETS object

**Usage**

```
## S3 method for class 'ETS'
residuals(object, ...)
```

```
## S3 method for class 'PTS'
residuals(object, ...)
```

```
## S3 method for class 'TETS'
residuals(object, ...)
```

Arguments

object            Object of class “TETS”.  
...               Additional inputs to function.

Details

See help of ETS.  
See help of PTS.  
See help of TETS.

Author(s)

Diego J. Pedregal

See Also

[ETS](#), [ETSmodel](#), [ETSvalidate](#), [ETScomponents](#), [ETSeestim](#)  
[PTS](#), [PTSmodel](#), [PTSvalidate](#), [PTScomponents](#), [PTSeestim](#)  
[TETS](#), [TETSmodel](#), [TETSvalidate](#), [TETScomponents](#), [TETSeestim](#)

Examples

```
## Not run:  
m1 <- ETSmodel(log(gdp))  
residuals(m1)  
  
## End(Not run)  
## Not run:  
m1 <- PTSmodel(log(AirPassengers))  
residuals(m1)  
  
## End(Not run)  
## Not run:  
m1 <- TETSmodel(log(gdp))  
residuals(m1)  
  
## End(Not run)
```

---

|       |              |
|-------|--------------|
| roots | <i>roots</i> |
|-------|--------------|

---

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 | <i>rowMedians</i> |
|------------|-------------------|

---

**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

**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>

Examples

```
## Not run:  
sales  
  
## End(Not run)
```

---

|      |             |
|------|-------------|
| size | <i>size</i> |
|------|-------------|

---

Description

size of vectors or matrices  
Size of vector, matrix or array

Usage

```
size(y)  
  
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 or matrix 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 a 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**

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](#), [plotSlide](#), [Accuracy](#), [tsDisplay](#), [size](#)

**Examples**

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

---

slideAux

slideAux

---

**Description**

Auxiliary function run from slide

**Usage**

```

slideAux(
  y,
  orig,
  forecFun,
  h = 12,
  step = 1,
  output = TRUE,
  graph = TRUE,
  window = NA,
  parallel = 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                                       |
| ...      | rest of inputs to forecFun function                             |

**Value**

Next time stamp

**Author(s)**

Diego J. Pedregal

---

summary.ARIMA

summary.ARIMA

---

**Description**

Prints an ARIMA object on screen

**Usage**

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

**Arguments**

|        |                                |
|--------|--------------------------------|
| object | Object of class “ARIMA”.       |
| ...    | Additional inputs to function. |

**Details**

See help of ARIMA.

**Author(s)**

Diego J. Pedregal

**See Also**

[ARIMA](#), [ARIMAmode1](#), [ARIMAvalidate](#),

**Examples**

```
## Not run:  
m1 <- ARIMAmode1(log(gdp))  
summary(m1)  
  
## End(Not run)
```

---

summary.ETS

*summary.ETS*

---

**Description**

Prints an ETS object on screen

**Usage**

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

**Arguments**

|        |                                |
|--------|--------------------------------|
| object | Object of class “ETS”.         |
| ...    | Additional inputs to function. |

**Details**

See help of ETS.

**Author(s)**

Diego J. Pedregal

**See Also**

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

**Examples**

```
## Not run:
m1 <- ETSmodel(log(gdp))
summary(m1)

## End(Not run)
```

---

summary.PTS

*summary.PTS*

---

**Description**

Prints an PTS object on screen

**Usage**

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

**Arguments**

|        |                                |
|--------|--------------------------------|
| object | Object of class “PTS”.         |
| ...    | Additional inputs to function. |

**Details**

See help of PTS.

**Author(s)**

Diego J. Pedregal

**See Also**

[PTS](#), [PTSmodel](#), [PTSvalidate](#), [PTScomponents](#), [PTSestim](#)

## Examples

```
## Not run:
m1 <- PTSmodel(log(AirPassengers))
summary(m1)

## End(Not run)
```

---

summary.TETS

*summary.TETS*

---

## Description

Prints a TOBIT TETS object on screen

## Usage

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

## Arguments

|        |                                |
|--------|--------------------------------|
| object | Object of class “TETS”.        |
| ...    | Additional inputs to function. |

## Details

See help of TETS.

## Author(s)

Diego J. Pedregal

## See Also

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

## Examples

```
## Not run:
m1 <- TETSmodel(log(gdp))
summary(m1)

## End(Not run)
```



---

|          |                 |
|----------|-----------------|
| 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

**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 | <i>tests</i> |
|-------|--------------|

---

**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**

- y                    a vector, ts or tsibble object
- parts                proportion of sample to include in ratio of variances test
- 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
- avoid                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,

**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 | <i>TETS</i> |
|------|-------------|

---

## 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, ...)  |
| 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").   |

|                           |   |
|---------------------------|---|
| <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                            |

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

## Details

See help of `TETSmodel`.

## Value

An object of class `TETS`. See `TETSmodel`.

## Author(s)

Diego J. Pedregal

## See Also

[TETSmodel](#), [TETSvalidate](#), [TETScomponents](#), [TETSestim](#)

## Examples

```
## Not run:
y <- log(AirPassengers)
m1 <- TETS(y)
m1 <- TETS(y, model = "MAM")

## End(Not run)
```

---

|                |                       |
|----------------|-----------------------|
| TETScomponents | <i>TETScomponents</i> |
|----------------|-----------------------|

---

**Description**

Estimates components of TOBIT TETS models

**Usage**

TETScomponents(m)

**Arguments**

m                      an object of type TETS created with TETSmodel

**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](#), [TETSmodel](#), [TETSvalidate](#), [TETSestim](#)

**Examples**

```
## Not run:
m1 <- TETS(log(gdp))
m1 <- TETScomponents(m1)

## End(Not run)
```

---

TETSestim

*TETSestim*


---

## Description

Estimates and forecasts TOBIT TETS models

## Usage

```
TETSestim(m)
```

## Arguments

**m** an object of type TETS created with TETSmodel

## 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:

|               |   |
|---------------|---|
| <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

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

## Examples

```
## Not run:
m1 <- TETSsetup(log(gdp))
m1 <- TETSestim(m1)

## End(Not run)
```

TETSmodel

*TETSmodel***Description**

Estimates and forecasts TOBIT TETS general univariate models

**Usage**

```
TETSmodel(
  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  |
|              | <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

TETSmodel 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 TETSmodel or TETSest im:

|          |   |
|----------|---|
| 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](#), [TETSestim](#)

**Examples**

```
## Not run:
y <- log(AirPassengers)
m1 <- TETSmodel(y)
m1 <- TETSmodel(y, model = "A?A")

## End(Not run)
```

---

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 TETSmodel.

**Value**

An object of class TETS. See TETSmodel.

Author(s)

Diego J. Pedregal

See Also

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

Examples

```
## Not run:
y <- log(AirPassengers)
m1 <- TETSsetup(y)
m1 <- TETSsetup(y, model = "??")
m1 <- TETSsetup(y, model = "?AA")

## End(Not run)
```

---

|              |                     |
|--------------|---------------------|
| TETSvalidate | <i>TETSvalidate</i> |
|--------------|---------------------|

---

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 TETSmodel

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

**Examples**

```
## Not run:
m1 <- TETSmodel(log(gdp))
m1 <- TETSvalidate(m1)

## End(Not run)
```

---

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   |

**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 = 9999,
  lambda = 1,
  outlier = 9999,
  tTest = FALSE,
  criterion = "aic",
  periods = NA,
  verbose = FALSE,
  stepwise = FALSE,
  p0 = -9999.9,
  arma = TRUE,
  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> </ul> |

|                  |  |
|------------------|--|
|                  | <ul style="list-style-type: none"> <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/lt").  |
| 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 UCmodel 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

```
## Not run:
y <- log(AirPassengers)
m1 <- UC(y)
m1 <- UC(y, model = "l1t/different/arma(0,0)")

## End(Not run)
```

---

`UCcomponents`*UCcomponents*

---

**Description**

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

**Usage**

```
UCcomponents(sys)
```

**Arguments**

`sys` an object of type UComp created with UC or UCmodel

**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](#), [UCmodel](#), [UCvalidate](#), [UCfilter](#), [UCsmooth](#), [UCdisturb](#), [UChp](#)

**Examples**

```
## Not run:  
m1 <- UC(log(AirPassengers))  
m1 <- UCcomponents(m1)  
  
## End(Not run)
```



---

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](#), [UCmodel](#), [UCvalidate](#), [UCfilter](#), [UCsmooth](#), [UCcomponents](#), [UChp](#)

**Examples**

```
## Not run:
m1 <- UC(log(AirPassengers))
m1 <- UCdisturb(m1)

## End(Not run)
```

---

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:

- p: Estimated transformed parameters
- v: Estimated innovations (white noise in correctly specified models)
- yFor: Forecast values of output
- yForV: Forecasted values variance
- criteria: Value of criteria for estimated model
- covp: Covariance matrix of estimated transformed parameters
- grad: Gradient of log-likelihood at the optimum
- iter: Estimation iterations

**Author(s)**

Diego J. Pedregal

**See Also**

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

**Examples**

```
## Not run:
m1 <- UCsetup(log(AirPassengers))
m1 <- UCestim(m1)

## End(Not run)
```

---

UCfilter

*UCfilter*


---

**Description**

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

**Usage**

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

**Examples**

```
## Not run:
m1 <- UC(log(AirPassengers))
m1 <- UCfilter(m1)

## End(Not run)
```

---

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](#), [UCmodel](#), [UCvalidate](#), [UCfilter](#), [UCsmooth](#), [UCcomponents](#), [UCdisturb](#)

**Examples**

```
## Not run:
cycle <- UChp(USgdp)
plot(cycle)

## End(Not run)
```

---

UCmodel

*UCmodel*


---

**Description**

Estimates and forecasts UC general univariate models

**Usage**

```
UCmodel(
  y,
  u = NULL,
  model = "?/none/?/?",
  h = 9999,
  lambda = 1,
  outlier = 9999,
  tTest = FALSE,
  criterion = "aic",
  periods = NA,
  verbose = FALSE,
  stepwise = FALSE,
  p0 = -9999.9,
  arma = TRUE,
  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

UCmodel 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 and forecasts the data. 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 UCmodel or UCestim:

- p: Estimated parameters
- v: Estimated innovations (white noise in correctly specified models)
- yFor: Forecasted values of output
- yForV: Forecasted values +- 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

```
## Not run:
y <- log(AirPassengers)
m1 <- UCmodel(y)
m1 <- UCmodel(y, model = "l1t/equal/arma(0,0)")

## End(Not run)
```

---

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.
- ExponenTial 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

---

|                |                |
|----------------|----------------|
| <i>UCsetup</i> | <i>UCsetup</i> |
|----------------|----------------|

---

### Description

Sets up UC general univariate models

### Usage

```
UCsetup(
  y,
  u = NULL,
  model = "?/none/?/?",
  h = 9999,
  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

|          |   |
|----------|---|
| <i>y</i> | 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 <i>periods</i> 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").  |
| 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 UCmodel or UCestim:

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

**Examples**

```
## Not run:
y <- log(AirPassengers)
m1 <- UCsetup(y)
m1 <- UCsetup(y, outlier = 4)
m1 <- UCsetup(y, model = "l1t/equal/arma(0,0)")
m1 <- UCsetup(y, model = "?/?/?/?")
m1 <- UCsetup(y, model = "l1t/?/equal/?", outlier = 4)

## End(Not run)
```

---

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](#), [UCmodel](#), [UCvalidate](#), [UCfilter](#), [UCdisturb](#), [UCcomponents](#), [UChp](#)

**Examples**

```
## Not run:
m1 <- UC(log(AirPassengers))
m1 <- UCsmooth(m1)

## End(Not run)
```

---

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**

|                          |   |
|--------------------------|---|
| <code>sys</code>         | an object of type <code>UComp</code> created with <code>UC</code> |
| <code>printScreen</code> | 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](#), [UCmodel](#), [UCfilter](#), [UCsmooth](#), [UCdisturb](#), [UCcomponents](#), [UChp](#)

Examples

```
## Not run:
m1 <- UC(log(gdp))
m1 <- UCvalidate(m1)

## End(Not run)
```

---

|       |               |
|-------|---------------|
| USgdp | <i>US GDP</i> |
|-------|---------------|

---

Description

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

Usage

USgdp

Format

Time series objects.  
Quarterly data from 1962 to 2019

Source

USgdp

Examples

```
## Not run:
USgdp

## End(Not run)
```

---

|         |                |
|---------|----------------|
| varTest | <i>varTest</i> |
|---------|----------------|

---

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)
```

---

|        |               |
|--------|---------------|
| zplane | <i>zplane</i> |
|--------|---------------|

---

**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

**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, 7  
ch4, 17  
gdp, 32  
ipi, 36  
OECDgdp, 38  
sales, 58  
USgdp, 94

Accuracy, 4, 5, 15, 18–20, 32, 34, 42, 57–60, 65, 66, 76, 95, 96  
acft, 4, 5, 15, 18–20, 32, 34, 42, 57–60, 65, 66, 76, 95, 96  
AIC.UComp, 6  
airpas, 7  
ARIMA, 7, 9, 11, 13, 14, 39, 44, 62  
ARIMAestim, 9  
ARIMAmodel, 8, 9, 10, 13, 14, 39, 44, 62  
ARIMAsetup, 12  
ARIMAvalidate, 8, 9, 11, 13, 14, 39, 44, 62  
arma2tsi, 14  
armaFilter, 4, 5, 15, 18–20, 32, 34, 42, 57–60, 65, 66, 76, 95, 96  
auxInvBoxCox, 15

BIC.UComp, 16  
box.cox, 17, 35

ch4, 17  
colMedians, 4, 5, 15, 18, 19, 20, 32, 34, 42, 57–60, 65, 66, 76, 95, 96  
conv, 4, 5, 15, 18, 19, 20, 32, 34, 42, 57–60, 65, 66, 76, 95, 96  
cusum, 4, 5, 15, 18, 19, 20, 32, 34, 42, 57–60, 65, 66, 76, 95, 96

dif, 4, 5, 15, 18–20, 20, 32, 34, 42, 57–60, 65, 66, 76, 95, 96

ETS, 21, 23, 24, 27, 29, 31, 40, 44, 56, 63

ETScomponents, 22, 23, 24, 27, 29, 31, 40, 44, 56, 63  
ETSeestim, 22, 23, 24, 27, 29, 31, 40, 44, 56, 63  
ETSmodel, 22–24, 25, 29, 31, 40, 44, 56, 63  
ETSsetup, 27  
ETSvalidate, 22–24, 27, 29, 29, 31, 40, 44, 56, 63  
extract, 30

fitted.ETS, 30  
fitted.PTS (fitted.ETS), 30  
fitted.TETS (fitted.ETS), 30

gaussTest, 4, 5, 15, 18–20, 31, 34, 42, 57–60, 65, 66, 76, 95, 96

gdp, 32  
getp0, 33

ident, 4, 5, 15, 18–20, 32, 34, 42, 57–60, 65, 66, 76, 95, 96  
inv.box.cox, 17, 35  
invBoxCox, 35  
ipi, 36

modelUC2arma, 37  
modelUC2PTS, 37

OECDgdp, 38

plot.ARIMA, 38  
plot.ETS, 39  
plot.PTS (plot.ETS), 39  
plot.TETS (plot.ETS), 39  
plotAcfPacf, 40  
plotBar, 41  
plotSlide, 4, 5, 15, 18–20, 32, 34, 41, 57–60, 65, 66, 76, 95, 96  
plus\_one, 42  
predict.UComp, 43  
print.ARIMA, 44  
print.ETS (print.ARIMA), 44

- print.PTS (print.ARIMA), 44
- print.TETS (print.ARIMA), 44
- PTS, 31, 40, 44, 45, 48, 49, 51, 53, 54, 56, 63
- PTS2modelUC, 47
- PTScomponents, 31, 40, 44, 47, 48, 49, 51, 53, 54, 56, 63
- PTSeestim, 31, 40, 44, 47, 48, 49, 51, 53, 54, 56, 63
- PTSmodel, 31, 40, 44, 47–49, 50, 53, 54, 56, 63
- PTSsetup, 47–49, 51, 52, 54
- PTSvalidate, 31, 40, 44, 47–49, 51, 53, 54, 56, 63
  
- removeNaNs, 55
- residuals.ETS, 55
- residuals.PTS (residuals.ETS), 55
- residuals.TETS (residuals.ETS), 55
- roots, 4, 5, 15, 18–20, 32, 34, 42, 56, 58–60, 65, 66, 76, 95, 96
- rowMedians, 4, 5, 15, 18–20, 32, 34, 42, 57, 59, 60, 65, 66, 76, 95, 96
  
- sales, 58
- size, 4, 5, 15, 18–20, 32, 34, 42, 57, 58, 58, 60, 65, 66, 76, 95, 96
- slide, 4, 5, 15, 18–20, 32, 34, 42, 57–59, 59, 65, 66, 76, 95, 96
- slideAux, 60
- summary.ARIMA, 61
- summary.ETS, 62
- summary.PTS, 63
- summary.TETS, 64
- sumStats, 4, 5, 15, 18–20, 32, 34, 42, 57–60, 65, 66, 76, 95, 96
  
- tests, 4, 5, 15, 18–20, 32, 34, 42, 57–60, 65, 66, 76, 95, 96
- TETS, 31, 40, 44, 56, 64, 67, 69, 70, 73, 75
- TETScomponents, 31, 40, 44, 56, 64, 68, 69, 70, 73, 75
- TETSeestim, 31, 40, 44, 56, 64, 68, 69, 70, 73, 75
- TETSmodel, 31, 40, 44, 56, 64, 68–70, 71, 75
- TETSsetup, 73
- TETSvalidate, 31, 40, 44, 56, 64, 68–70, 73, 75, 75
- tsDisplay, 4, 5, 15, 18–20, 32, 34, 42, 57–60, 65, 66, 76, 95, 96
- UC, 6, 16, 17, 33, 35, 43, 77, 79–81, 83–85, 87, 91–93
- UCcomponents, 6, 16, 17, 33, 35, 43, 79, 80, 81, 83–85, 87, 91–93
- UCdisturb, 6, 16, 17, 33, 35, 43, 79, 80, 81, 83–85, 87, 91–93
- UCestim, 82
- UCfilter, 6, 16, 17, 33, 35, 43, 79–81, 83, 83, 85, 87, 91–93
- UChp, 33, 79–81, 83, 84, 84, 87, 91–93
- UCmodel, 6, 16, 17, 33, 35, 43, 80, 81, 83–85, 85, 91–93
- UComp, 88
- UComp-package (UComp), 88
- UCsetup, 33, 89
- UCsmooth, 6, 16, 17, 33, 35, 43, 79–81, 83–85, 87, 91, 92, 93
- UCvalidate, 6, 16, 17, 33, 35, 43, 79–81, 83–85, 87, 91, 92, 93
- USgdp, 94
  
- varTest, 4, 5, 15, 18–20, 32, 34, 42, 57–60, 65, 66, 76, 94, 96
  
- zplane, 4, 5, 15, 18–20, 32, 34, 42, 57–60, 65, 66, 76, 95, 95