

# Package ‘simts’

September 3, 2022

**Type** Package

**Title** Time Series Analysis Tools

**Version** 0.2.1

**Date** 2022-09-02

**LazyData** true

**Maintainer** Stéphane Guerrier <stef.guerrier@gmail.com>

**Description** A system contains easy-to-use tools as a support for time series analysis courses. In particular, it incorporates a technique called Generalized Method of Wavelet Moments (GMWM) as well as its robust implementation for fast and robust parameter estimation of time series models which is described, for example, in Guerrier et al. (2013) <doi:10.1080/01621459.2013.799920>. More details can also be found in the paper linked to via the URL below.

**Depends** R (>= 3.6.0)

**License** AGPL-3 | file LICENSE

**Imports** Rcpp, stats, utils, scales, grDevices, graphics, broom, dplyr, magrittr, methods, purrr, tidyr, robcor

**LinkingTo** Rcpp, RcppArmadillo

**RoxygenNote** 7.2.1

**Encoding** UTF-8

**Suggests** knitr, rmarkdown

**VignetteBuilder** knitr

**URL** <https://github.com/SMAC-Group/simts>,  
<https://arxiv.org/pdf/1607.04543.pdf>

**BugReports** <https://github.com/SMAC-Group/simts/issues>

**SystemRequirements** C++11

**NeedsCompilation** yes

**Author** Stéphane Guerrier [aut, cre, cph],  
James Balamuta [aut, cph],  
Roberto Molinari [aut, cph],

Justin Lee [aut],  
 Lionel Voirol [aut],  
 Yuming Zhang [aut],  
 Wenchao Yang [ctb],  
 Nathanael Claussen [ctb],  
 Yunxiang Zhang [ctb],  
 Christian Gunning [cph],  
 Romain Francois [cph],  
 Ross Ihaka [cph],  
 R Core Team [cph]

**Repository** CRAN

**Date/Publication** 2022-09-03 17:00:02 UTC

## R topics documented:

AIC.fitsimts . . . . .	4
AR . . . . .	5
AR1 . . . . .	6
ar1_to_wv . . . . .	7
ARIMA . . . . .	8
ARMA . . . . .	9
ARMA11 . . . . .	10
arma11_to_wv . . . . .	11
arma_to_wv . . . . .	12
australia . . . . .	13
auto_corr . . . . .	13
best_model . . . . .	15
check . . . . .	15
compare_acf . . . . .	16
corr_analysis . . . . .	17
derivative_first_matrix . . . . .	18
deriv_2nd_ar1 . . . . .	19
deriv_2nd_arma11 . . . . .	20
deriv_2nd_dr . . . . .	21
deriv_2nd_ma1 . . . . .	22
deriv_ar1 . . . . .	23
deriv_arma11 . . . . .	24
deriv_dr . . . . .	25
deriv_ma1 . . . . .	25
deriv_qn . . . . .	26
deriv_rw . . . . .	27
deriv_wn . . . . .	28
diag_boxpierce . . . . .	28
diag_ljungbox . . . . .	29
diag_plot . . . . .	30
diag_portmanteau_ . . . . .	30
DR . . . . .	31

dr_to_wv	32
estimate	33
evaluate	34
FGN	35
gen_ar1blocks	36
gen_bi	37
gen_gts	38
gen_lts	39
gen_nswn	41
GM	42
gmwm	43
gmwm_imu	46
gts	47
hydro	49
imu	49
imu_time	51
is.gts	52
lts	53
M	54
MA	55
MA1	56
ma1_to_wv	57
make_frame	58
MAPE	59
MAT	60
np_boot_sd_med	61
plot.gmwm	61
plot.PACF	62
plot.simtsACF	64
plot_pred	65
PLP	66
predict.fitsimts	67
predict.gmwm	68
QN	69
qn_to_wv	70
read.imu	70
resid_plot	71
rgmwm	72
rtruncated_normal	73
RW	73
RW2dimension	74
rw_to_wv	75
SARIMA	75
SARMA	77
savingrt	78
select	78
select_arima	79
simple_diag_plot	80

simplified_print_SARIMA . . . . .	81
SIN . . . . .	82
summary.fitsimts . . . . .	83
summary.gmwm . . . . .	83
theo_acf . . . . .	84
theo_pacf . . . . .	85
update.gmwm . . . . .	86
update.lts . . . . .	87
value . . . . .	88
WN . . . . .	89
wn_to_wv . . . . .	90
[.imu . . . . .	91

<b>Index</b>	<b>93</b>
--------------	-----------

---

AIC.fitsimts	<i>Akaike's Information Criterion</i>
--------------	---------------------------------------

---

### Description

This function calculates AIC, BIC or HQ for a fitsimts object. This function currently only supports models estimated by the MLE.

### Usage

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

### Arguments

object	A fitsimts object.
k	The penalty per parameter to be used; the default k = 2 is the classical AIC.
...	Optionally more fitted model objects.

### Value

AIC, BIC or HQ

### Author(s)

Stéphane Guerrier

**Examples**

```

set.seed(1)
n = 300
Xt = gen_gts(n, AR(phi = c(0, 0, 0.8), sigma2 = 1))
mod = estimate(AR(3), Xt)

# AIC
AIC(mod)

# BIC
AIC(mod, k = log(n))

# HQ
AIC(mod, k = 2*log(log(n)))

```

AR

*Create an Autoregressive P [AR(P)] Process***Description**

Sets up the necessary backend for the AR(P) process.

**Usage**

```
AR(phi = NULL, sigma2 = 1)
```

**Arguments**

<b>phi</b>	A vector with double values for the $\phi$ of an AR(P) process (see Note for details).
<b>sigma2</b>	A double value for the variance, $\sigma^2$ , of an AR(P) process. (see Note for details).

**Value**

An S3 object with called `ts.model` with the following structure:

**process.desc** Used in summary: "AR-1", "AR-2", ..., "AR-P", "SIGMA2"

**theta**  $\phi_1, \phi_2, \dots, \phi_p, \sigma^2$

**plength** Number of Parameters

**desc** "AR"

**print** String containing simplified model

**obj.desc** Depth of Parameters e.g. `list(p,1)`

**starting** Guess starting values? TRUE or FALSE (e.g. specified value)

**Note**

We consider the following model:

$$X_t = \sum_{j=1}^p \phi_j X_{t-j} + \varepsilon_t$$

, where  $\varepsilon_t$  is iid from a zero mean normal distribution with variance  $\sigma^2$ .

**Author(s)**

James Balamuta

**Examples**

```
AR(1) # Slower version of AR1()
AR(phi=.32, sigma=1.3) # Slower version of AR1()
AR(2) # Equivalent to ARMA(2,0).
```

---

AR1

*Definition of an Autoregressive Process of Order 1*

---

**Description**

Definition of an Autoregressive Process of Order 1

**Usage**

```
AR1(phi = NULL, sigma2 = 1)
```

**Arguments**

**phi** A double value for the parameter  $\phi$  (see Note for details).  
**sigma2** A double value for the variance parameter  $\sigma^2$  (see Note for details).

**Value**

An S3 object containing the specified ts.model with the following structure:

**process.desc** Used in summary: "AR1","SIGMA2"

**theta** Parameter vector including  $\phi, \sigma^2$

**plength** Number of parameters

**print** String containing simplified model

**desc** "AR1"

**obj.desc** Depth of Parameters e.g. list(1,1)

**starting** Find starting values? TRUE or FALSE (e.g. specified value)

**Note**

We consider the following AR(1) model:

$$X_t = \phi X_{t-1} + \varepsilon_t$$

, where  $\varepsilon_t$  is iid from a zero mean normal distribution with variance  $\sigma^2$ .

**Author(s)**

James Balamuta

**Examples**

```
AR1()
AR1(phi=.32, sigma2 = 1.3)
```

---

ar1\_to\_wv

*AR(1) process to WV*


---

**Description**

This function computes the Haar WV of an AR(1) process

**Usage**

```
ar1_to_wv(phi, sigma2, tau)
```

**Arguments**

phi	A double that is the phi term of the AR(1) process
sigma2	A double corresponding to variance of AR(1) process
tau	A vec containing the scales e.g. $2^T$

**Details**

This function is significantly faster than its generalized counter part [arma\\_to\\_wv](#).

**Value**

A vec containing the wavelet variance of the AR(1) process.

**Process Haar Wavelet Variance Formula**

The Autoregressive Order 1 (AR(1)) process has a Haar Wavelet Variance given by:

$$\frac{2\sigma^2 \left( 4\phi^{\frac{\tau_j}{2}+1} - \phi^{\tau_j+1} - \frac{1}{2}\phi^2\tau_j + \frac{\tau_j}{2} - 3\phi \right)}{(1-\phi)^2 (1-\phi^2) \tau_j^2}$$

---

ARIMA	<i>Create an Autoregressive Integrated Moving Average (ARIMA) Process</i>
-------	---

---

### Description

Sets up the necessary backend for the ARIMA process.

### Usage

```
ARIMA(ar = 1, i = 0, ma = 1, sigma2 = 1)
```

### Arguments

ar	A vector or integer containing either the coefficients for $\phi$ 's or the process number $p$ for the Autoregressive (AR) term.
i	An integer containing the number of differences to be done.
ma	A vector or integer containing either the coefficients for $\theta$ 's or the process number $q$ for the Moving Average (MA) term.
sigma2	A double value for the standard deviation, $\sigma$ , of the ARIMA process.

### Details

A variance is required since the model generation statements utilize randomization functions expecting a variance instead of a standard deviation like R.

### Value

An S3 object with called `ts.model` with the following structure:

**process.desc** *AR \* p, MA \* q*  
**theta**  $\sigma$   
**plength** Number of parameters  
**print** String containing simplified model  
**obj.desc** y desc replicated x times  
**obj** Depth of parameters e.g. `list(c(length(ar),length(ma),1))`  
**starting** Guess starting values? TRUE or FALSE (e.g. specified value)

### Note

We consider the following model:

$$\Delta^i X_t = \sum_{j=1}^p \phi_j \Delta^i X_{t-j} + \sum_{j=1}^q \theta_j \varepsilon_{t-j} + \varepsilon_t$$

, where  $\varepsilon_t$  is iid from a zero mean normal distribution with variance  $\sigma^2$ .

**Author(s)**

James Balamuta

**Examples**

```
# Create an ARMA(1,2) process
ARIMA(ar=1,2)
# Creates an ARMA(3,2) process with predefined coefficients.
ARIMA(ar=c(0.23, .43, .59), ma=c(0.4, .3))

# Creates an ARMA(3,2) process with predefined coefficients and standard deviation
ARIMA(ar=c(0.23, .43, .59), ma=c(0.4, .3), sigma2 = 1.5)
```

ARMA

*Create an Autoregressive Moving Average (ARMA) Process***Description**

Sets up the necessary backend for the ARMA process.

**Usage**

```
ARMA(ar = 1, ma = 1, sigma2 = 1)
```

**Arguments**

<code>ar</code>	A vector or integer containing either the coefficients for $\phi$ 's or the process number $p$ for the Autoregressive (AR) term.
<code>ma</code>	A vector or integer containing either the coefficients for $\theta$ 's or the process number $q$ for the Moving Average (MA) term.
<code>sigma2</code>	A double value for the standard deviation, $\sigma$ , of the ARMA process.

**Details**

A variance is required since the model generation statements utilize randomization functions expecting a variance instead of a standard deviation like R.

**Value**

An S3 object with called `ts.model` with the following structure:

```
process.desc AR * p, MA * q
theta  $\sigma$ 
plength Number of Parameters
print String containing simplified model
obj.desc y desc replicated x times
obj Depth of Parameters e.g. list(c(length(ar),length(ma),1))
starting Guess Starting values? TRUE or FALSE (e.g. specified value)
```

**Note**

We consider the following model:

$$X_t = \sum_{j=1}^p \phi_j X_{t-j} + \sum_{j=1}^q \theta_j \varepsilon_{t-j} + \varepsilon_t$$

, where  $\varepsilon_t$  is iid from a zero mean normal distribution with variance  $\sigma^2$ .

**Author(s)**

James Balamuta

**Examples**

```
# Create an ARMA(1,2) process
ARMA(ar=1,2)
# Creates an ARMA(3,2) process with predefined coefficients.
ARMA(ar=c(0.23, .43, .59), ma=c(0.4, .3))

# Creates an ARMA(3,2) process with predefined coefficients and standard deviation
ARMA(ar=c(0.23, .43, .59), ma=c(0.4, .3), sigma2 = 1.5)
```

---

ARMA11

*Definition of an ARMA(1,1)*

---

**Description**

Definition of an ARMA(1,1)

**Usage**

```
ARMA11(phi = NULL, theta = NULL, sigma2 = 1)
```

**Arguments**

phi	A double containing the parameter $\phi_1$ (see Note for details).
theta	A double containing the parameter $\theta_1$ (see Note for details).
sigma2	A double value for the parameter $\sigma^2$ (see Note for details).

**Details**

A variance is required since the model generation statements utilize randomization functions expecting a variance instead of a standard deviation like R.

**Value**

An S3 object with called ts.model with the following structure:

**process.desc** *AR1, MA1, SIGMA2*

**theta**  $\phi, \theta, \sigma^2$

**plength** Number of Parameters: 3

**print** String containing simplified model

**obj.desc** Depth of Parameters e.g. list(c(1,1,1))

**starting** Guess Starting values? TRUE or FALSE (e.g. specified value)

**Note**

We consider the following model:

$$X_t = \phi X_{t-1} + \theta_1 \varepsilon_{t-1} + \varepsilon_t,$$

where  $\varepsilon_t$  is iid from a zero mean normal distribution with variance  $\sigma^2$ .

**Author(s)**

James Balamuta

**Examples**

```
# Creates an ARMA(1,1) process with predefined coefficients.
ARMA11(phi = .23, theta = .1, sigma2 = 1)
```

```
# Creates an ARMA(1,1) process with values to be guessed on calibration.
ARMA11()
```

---

 arma11\_to\_wv

*ARMA(1,1) to WV*


---

**Description**

This function computes the WV (haar) of an Autoregressive Order 1 - Moving Average Order 1 (ARMA(1,1)) process.

**Usage**

```
arma11_to_wv(phi, theta, sigma2, tau)
```

**Arguments**

phi	A double corresponding to the autoregressive term.
theta	A double corresponding to the moving average term.
sigma2	A double the variance of the process.
tau	A vec containing the scales e.g. $2^\tau$

**Details**

This function is significantly faster than its generalized counter part [arma\\_to\\_wv](#)

**Value**

A vec containing the wavelet variance of the ARMA(1,1) process.

**Process Haar Wavelet Variance Formula**

The Autoregressive Order 1 and Moving Average Order 1 (ARMA(1,1)) process has a Haar Wavelet Variance given by:

$$\nu_j^2(\phi, \theta, \sigma^2) = - \frac{2\sigma^2 \left( -\frac{1}{2}(\theta + 1)^2 (\phi^2 - 1) \tau_j - (\theta + \phi)(\theta\phi + 1) \left( \phi^{\tau_j} - 4\phi^{\frac{\tau_j}{2}} + 3 \right) \right)}{(\phi - 1)^3 (\phi + 1) \tau_j^2}$$

---

 arma\_to\_wv

*ARMA process to WV*


---

**Description**

This function computes the Haar Wavelet Variance of an ARMA process

**Usage**

```
arma_to_wv(ar, ma, sigma2, tau)
```

**Arguments**

ar	A vec containing the coefficients of the AR process
ma	A vec containing the coefficients of the MA process
sigma2	A double containing the residual variance
tau	A vec containing the scales e.g. $2^\tau$

**Details**

The function is a generic implementation that requires a stationary theoretical autocorrelation function (ACF) and the ability to transform an ARMA( $p,q$ ) process into an MA( $\infty$ ) (e.g. infinite MA process).

**Value**

A vec containing the wavelet variance of the ARMA process.

**Process Haar Wavelet Variance Formula**

The Autoregressive Order  $p$  and Moving Average Order  $q$  (ARMA( $p,q$ )) process has a Haar Wavelet Variance given by:

$$\frac{\tau_j [1 - \rho(\frac{\tau_j}{2})] + 2 \sum_{i=1}^{\frac{\tau_j}{2}-1} i [2\rho(\frac{\tau_j}{2} - i) - \rho(i) - \rho(\tau_j - i)]}{\tau_j^2} \sigma_X^2$$

where  $\sigma_X^2$  is given by the variance of the ARMA process. Furthermore, this assumes that stationarity has been achieved as it directly

---

 australia

*Quarterly Increase in Stocks Non-Farm Total, Australia*


---

**Description**

A dataset containing the quarterly increase in stocks non-farm total in Australia, with frequency 4 starting from September 1959 to March 1991 with a total of 127 observations.

**Usage**

australia

**Format**

A data frame with 127 rows and 2 variables:

**Quarter** year and quarter

**Increase** quarterly increase in stocks non-farm total

**Source**

Time Series Data Library (citing: Australian Bureau of Statistics) datamarket

---

 auto\_corr

*Empirical ACF and PACF*


---

**Description**

This function can estimate either the autocovariance / autocorrelation for univariate time series, or the partial autocovariance / autocorrelation for univariate time series.

**Usage**

```

auto_corr(
  x,
  lag.max = NULL,
  pacf = FALSE,
  type = "correlation",
  demean = TRUE,
  robust = FALSE
)

```

**Arguments**

<code>x</code>	A vector or ts object (of length $N > 1$ ).
<code>lag.max</code>	An integer indicating the maximum lag up to which to compute the empirical ACF / PACF.
<code>pacf</code>	A boolean indicating whether to output the PACF. If it's TRUE, then the function will only estimate the empirical PACF. If it's FALSE (the default), then the function will only estimate the empirical ACF.
<code>type</code>	A character string giving the type of acf to be computed. Allowed values are "correlation" (the default) and "covariance".
<code>demean</code>	A boolean indicating whether the data should be detrended (TRUE) or not (FALSE). Defaults to TRUE.
<code>robust</code>	A boolean indicating whether a robust estimator should be used (TRUE) or not (FALSE). Defaults to FALSE. This only works when the function is estimating ACF.

**Details**

lagmax default is  $10 * \log_{10}(N/m)$  where  $N$  is the number of observations and  $m$  is the number of time series being compared. If lagmax supplied is greater than the number of observations  $N$ , then one less than the total will be taken (i.e.  $N - 1$ ).

**Value**

An array of dimensions  $N \times 1 \times 1$ .

**Author(s)**

Yuming Zhang

**Examples**

```

m = auto_corr(datasets::AirPassengers)
m = auto_corr(datasets::AirPassengers, pacf = TRUE)

```

---

best_model	<i>Select the Best Model</i>
------------	------------------------------

---

### Description

This function retrieves the best model from a selection procedure.

### Usage

```
best_model(x, ic = "aic")
```

### Arguments

x	An object of class <code>select_arma</code> , <code>select_ar</code> or <code>select_ma</code> .
ic	A string indicating the type of criterion to use in selecting the best model. Supported criteria include "aic" (AIC), "bic" (BIC) and "hq" (HQ).

### Examples

```
set.seed(18)
xt = gen_arima(N=100, ar=0.3, d=1, ma=0.3)
x = select_arima(xt, d=1L)
best_model(x, ic = "aic")

set.seed(19)
xt = gen_ma1(100, 0.3, 1)
x = select_ma(xt, q.min=2L, q.max=5L)
best_model(x, ic = "bic")

set.seed(20)
xt = gen_arma(100, c(.3,.5), c(.1), 1, 0)
x = select_arma(xt, p.min = 1L, p.max = 4L,
               q.min = 1L, q.max = 3L)
best_model(x, ic = "hq")
```

---

check	<i>Diagnostics on Fitted Time Series Model</i>
-------	--

---

### Description

This function can perform (simple) diagnostics on the fitted time series model. It can output 6 diagnostic plots to assess the model, including (1) residuals plot, (2) histogram of distribution of standardized residuals, (3) Normal Q-Q plot of residuals, (4) ACF plot, (5) PACF plot, (6) Box test results.

**Usage**

```
check(model = NULL, resids = NULL, simple = FALSE)
```

**Arguments**

model	A fitsimts, lm or gam object.
resids	A vector of residuals for diagnostics.
simple	A boolean indicating whether to return simple diagnostic plots or not.

**Author(s)**

Stéphane Guerrier and Yuming Zhang

**Examples**

```
Xt = gen_gts(300, AR(phi = c(0, 0, 0.8), sigma2 = 1))
model = estimate(AR(3), Xt)
check(model)

check(resids = rnorm(100))

Xt = gen_gts(1000, SARIMA(ar = c(0.5, -0.25), i = 0, ma = 0.5, sar = -0.8,
si = 1, sma = 0.25, s = 24, sigma2 = 1))
model = estimate(SARIMA(ar = 2, i = 0, ma = 1, sar = 1, si = 1, sma = 1, s = 24),
Xt, method = "rgmwm")
check(model)
check(model, simple=TRUE)
```

---

compare\_acf

*Comparison of Classical and Robust Correlation Analysis Functions*

---

**Description**

Compare classical and robust ACF of univariate time series.

**Usage**

```
compare_acf(
  x,
  lag.max = NULL,
  demean = TRUE,
  show.ci = TRUE,
  alpha = 0.05,
  plot = TRUE,
  ...
)
```

**Arguments**

x	A vector or "ts" object (of length $N > 1$ ).
lag.max	A integer indicating the maximum lag up to which to compute the ACF and PACF functions.
demean	A bool indicating whether the data should be detrended (TRUE) or not (FALSE). Defaults to TRUE.
show.ci	A bool indicating whether to compute and show the confidence region. Defaults to TRUE.
alpha	A double indicating the level of significance for the confidence interval. By default $\alpha = 0.05$ which gives a $1 - \alpha = 0.95$ confidence interval.
plot	A bool indicating whether a plot of the computed quantities should be produced. Defaults to TRUE.
...	Additional parameters.

**Author(s)**

Yunxiang Zhang

**Examples**

```
# Estimate both the ACF and PACF functions
compare_acf(datasets::AirPassengers)
```

---

corr\_analysis

*Correlation Analysis Functions*

---

**Description**

Correlation Analysis function computes and plots both empirical ACF and PACF of univariate time series.

**Usage**

```
corr_analysis(  
  x,  
  lag.max = NULL,  
  type = "correlation",  
  demean = TRUE,  
  show.ci = TRUE,  
  alpha = 0.05,  
  plot = TRUE,  
  ...  
)
```

**Arguments**

x	A vector or "ts" object (of length $N > 1$ ).
lag.max	A integer indicating the maximum lag up to which to compute the ACF and PACF functions.
type	A character string giving the type of acf to be computed. Allowed values are "correlation" (the default) and "covariance".
demean	A bool indicating whether the data should be detrended (TRUE) or not (FALSE). Defaults to TRUE.
show.ci	A bool indicating whether to compute and show the confidence region. Defaults to TRUE.
alpha	A double indicating the level of significance for the confidence interval. By default $\alpha = 0.05$ which gives a $1 - \alpha = 0.95$ confidence interval.
plot	A bool indicating whether a plot of the computed quantities should be produced. Defaults to TRUE.
...	Additional parameters.

**Value**

Two array objects (ACF and PACF) of dimension  $N \times S \times S$ .

**Author(s)**

Yunxiang Zhang

**Examples**

```
# Estimate both the ACF and PACF functions
corr_analysis(datasets::AirPassengers)
```

---

derivative\_first\_matrix

*Analytic D matrix of Processes*

---

**Description**

This function computes each process to WV (haar) in a given model.

**Usage**

```
derivative_first_matrix(theta, desc, objdesc, tau)
```

**Arguments**

theta	A vec containing the list of estimated parameters.
desc	A vector<string> containing a list of descriptors.
objdesc	A field<vec> containing a list of object descriptors.
tau	A vec containing the scales e.g. $2^\tau$

**Details**

Function returns the matrix effectively known as "D"

**Value**

A matrix with the process derivatives going down the column

**Author(s)**

James Joseph Balamuta (JJB)

---

deriv_2nd_ar1	<i>Analytic second derivative matrix for AR(1) process</i>
---------------	--

---

**Description**

Calculates the second derivative for the AR(1) process and places it into a matrix form. The matrix form in this case is for convenience of the calculation.

**Usage**

```
deriv_2nd_ar1(phi, sigma2, tau)
```

**Arguments**

phi	A double corresponding to the phi coefficient of an AR(1) process.
sigma2	A double corresponding to the error term of an AR(1) process.
tau	A vec containing the scales e.g. $2^\tau$

**Value**

A matrix with the first column containing the second partial derivative with respect to  $\phi$  and the second column contains the second partial derivative with respect to  $\sigma^2$

**Process Haar WV Second Derivative**

Taking the second derivative with respect to  $\phi$  yields:

$$\frac{\partial^2}{\partial \phi^2} \nu_j^2(\phi, \sigma^2) = \frac{2\sigma^2 \left( (\phi^2 - 1) \tau_j \left( 2(\phi(7\phi + 4) + 1)\phi^{\frac{\tau_j}{2}-1} - (\phi(7\phi + 4) + 1)\phi^{\tau_j-1} + 3(\phi + 1)^2 \right) + (\phi^2 - 1)^2 \tau_j^2 \left( \phi^{\tau_j} - 2\phi^{\frac{\tau_j}{2}} - \phi - 1 \right) \right)}{(\phi - 1)^5 (\phi + 1)^3 \tau_j^2}$$

Taking the second derivative with respect to  $\sigma^2$  yields:

$$\frac{\partial^2}{\partial \sigma^4} \nu_j^2(\sigma^2) = 0$$

Taking the derivative with respect to  $\phi$  and  $\sigma^2$  yields:

$$\frac{\partial^2}{\partial \phi \partial \sigma^2} \nu_j^2(\phi, \sigma^2) = \frac{2 \left( (\phi^2 - 1) \tau_j \left( \phi^{\tau_j} - 2\phi^{\frac{\tau_j}{2}} - \phi - 1 \right) - (\phi(3\phi + 2) + 1) \left( \phi^{\tau_j} - 4\phi^{\frac{\tau_j}{2}} + 3 \right) \right)}{(\phi - 1)^4 (\phi + 1)^2 \tau_j^2}$$

**Author(s)**

James Joseph Balamuta (JJB)

---

 deriv\_2nd\_arma11      *Analytic D matrix for ARMA(1,1) process*


---

**Description**

Obtain the second derivative of the ARMA(1,1) process.

**Usage**

deriv\_2nd\_arma11(phi, theta, sigma2, tau)

**Arguments**

phi	A double corresponding to the phi coefficient of an ARMA(1,1) process.
theta	A double corresponding to the theta coefficient of an ARMA(1,1) process.
sigma2	A double corresponding to the error term of an ARMA(1,1) process.
tau	A vec containing the scales e.g. $2^T$

**Value**

A matrix with:

- The **first** column containing the second partial derivative with respect to  $\phi$ ;
- The **second** column containing the second partial derivative with respect to  $\theta$ ;
- The **third** column contains the second partial derivative with respect to  $\sigma^2$ .
- The **fourth** column contains the partial derivative with respect to  $\phi$  and  $\theta$ .
- The **fifth** column contains the partial derivative with respect to  $\sigma^2$  and  $\phi$ .
- The **sixth** column contains the partial derivative with respect to  $\sigma^2$  and  $\theta$ .

**Process Haar WV Second Derivative**Taking the second derivative with respect to  $\phi$  yields:

$$\frac{\partial^2}{\partial \phi^2} \nu_j^2(\phi, \theta, \sigma^2) = \frac{2\sigma^2}{(\phi - 1)^5 (\phi + 1)^3 \tau_j^2} \left( \begin{array}{l} (\phi - 1)^2 \left( (\phi + 1)^2 (\theta^2 \phi + \theta \phi^2 + \theta + \phi) \tau_j^2 \left( \phi^{\frac{\tau_j}{2}} \right. \right. \right. \\ \left. \left. \left. - 12(\phi - 1) \right) \right) \right. \\ \left. + 6(\phi + 1)(\phi - 1) \left( \frac{1}{2}(\theta + 1)^2 (\phi^2 - 1) \tau_j + (\theta + \phi)(\theta \phi + 1) \left( \phi^{\tau_j} - 4\phi^{\frac{\tau_j}{2}} \right) \right) \right) \end{array} \right)$$

Taking the second derivative with respect to  $\theta$  yields:

$$\frac{\partial^2}{\partial \theta^2} \nu_j^2(\phi, \theta, \sigma^2) = \frac{2\sigma^2 \left( (\phi^2 - 1) \tau_j + 2\phi \left( \phi^{\tau_j} - 4\phi^{\frac{\tau_j}{2}} + 3 \right) \right)}{(\phi - 1)^3 (\phi + 1) \tau_j^2}$$

Taking the second derivative with respect to  $\sigma^2$  yields:

$$\frac{\partial^2}{\partial \sigma^4} \nu_j^2(\phi, \theta, \sigma^2) = 0$$

Taking the derivative with respect to  $\sigma^2$  and  $\theta$  yields:

$$\frac{\partial}{\partial \theta} \frac{\partial}{\partial \sigma^2} \nu_j^2(\phi, \theta, \sigma^2) = \frac{2}{(\phi - 1)^3 (\phi + 1) \tau_j^2} \left( (\theta + 1) (\phi^2 - 1) \tau_j + (2\theta\phi + \phi^2 + 1) \left( \phi^{\tau_j} - 4\phi^{\frac{\tau_j}{2}} + 3 \right) \right)$$

Taking the derivative with respect to  $\sigma^2$  and  $\phi$  yields:

$$\frac{\partial}{\partial \phi} \frac{\partial}{\partial \sigma^2} \nu_j^2(\phi, \theta, \sigma^2) = \frac{2}{(\phi - 1)^4 (\phi + 1)^2 \tau_j^2} \left( \begin{array}{l} -(\phi - 1)(\phi + 1) \left( \begin{array}{l} -(\theta + \phi)(\theta\phi + 1)\tau_j \left( \phi^{\frac{\tau_j}{2}} - 2 \right) \phi^{\frac{\tau_j}{2} - 1} \\ -\theta(\theta + \phi) \left( \phi^{\tau_j} - 4\phi^{\frac{\tau_j}{2}} + 3 \right) \\ -(\theta\phi + 1) \left( \phi^{\tau_j} - 4\phi^{\frac{\tau_j}{2}} + 3 \right) \\ -(\theta + 1)^2 \phi \tau_j \end{array} \right) \\ +(\phi - 1) \left( -\frac{1}{2}(\theta + 1)^2 (\phi^2 - 1) \tau_j - (\theta + \phi)(\theta\phi + 1) \left( \phi^{\tau_j} - 4\phi^{\frac{\tau_j}{2}} + 3 \right) \right) \\ +3(\phi + 1) \left( -\frac{1}{2}(\theta + 1)^2 (\phi^2 - 1) \tau_j - (\theta + \phi)(\theta\phi + 1) \left( \phi^{\tau_j} - 4\phi^{\frac{\tau_j}{2}} + 3 \right) \right) \end{array} \right)$$

Taking the derivative with respect to  $\phi$  and  $\theta$  yields:

$$\frac{\partial}{\partial \theta} \frac{\partial}{\partial \phi} \nu_j^2(\phi, \theta, \sigma^2) = -\frac{2\sigma^2}{(\phi - 1)^4 (\phi + 1)^2 \tau_j^2} \left( \begin{array}{l} \tau_j \left( \begin{array}{l} 2(\theta + 1)(\phi - 1)(\phi + 1)^2 \\ +2(\phi^2 - 1)(2\theta\phi + \phi^2 + 1)\phi^{\frac{\tau_j}{2} - 1} \\ -(\phi^2 - 1)(2\theta\phi + \phi^2 + 1)\phi^{\tau_j - 1} \end{array} \right) \\ +2(\theta(\phi(3\phi + 2) + 1) + \phi(\phi^2 + \phi + 3) + 1) \left( \phi^{\tau_j} - 4\phi^{\frac{\tau_j}{2}} + 3 \right) \end{array} \right)$$

### Author(s)

James Joseph Balamuta (JJB)

---

deriv\_2nd\_dr

Analytic second derivative matrix for drift process

---

### Description

To ease a later calculation, we place the result into a matrix structure.

### Usage

deriv\_2nd\_dr(tau)

### Arguments

tau                      A vec containing the scales e.g.  $2^\tau$

**Value**

A matrix with the first column containing the second partial derivative with respect to  $\omega$ .

**Author(s)**

James Joseph Balamuta (JJB)

---

deriv\_2nd\_ma1

*Analytic second derivative for MA(1) process*

---

**Description**

To ease a later calculation, we place the result into a matrix structure.

**Usage**

deriv\_2nd\_ma1(theta, sigma2, tau)

**Arguments**

theta	A double corresponding to the theta coefficient of an MA(1) process.
sigma2	A double corresponding to the error term of an MA(1) process.
tau	A vec containing the scales e.g. $2^\tau$

**Value**

A matrix with the first column containing the second partial derivative with respect to  $\theta$ , the second column contains the partial derivative with respect to  $\theta$  and  $\sigma^2$ , and lastly we have the second partial derivative with respect to  $\sigma^2$ .

**Process Haar WV Second Derivative**

Taking the second derivative with respect to  $\theta$  yields:

$$\frac{\partial^2}{\partial \theta^2} \nu_j^2(\theta, \sigma^2) = \frac{2\sigma^2}{\tau_j}$$

Taking the second derivative with respect to  $\sigma^2$  yields:

$$\frac{\partial^2}{\partial \sigma^4} \nu_j^2(\theta, \sigma^2) = 0$$

Taking the first derivative with respect to  $\theta$  and  $\sigma^2$  yields:

$$\frac{\partial}{\partial \theta} \frac{\partial}{\partial \sigma^2} \nu_j^2(\theta, \sigma^2) = \frac{2(\theta + 1)\tau_j - 6}{\tau_j^2}$$

**Author(s)**

James Joseph Balamuta (JJB)

---

 deriv\_ar1

*Analytic D matrix for AR(1) process*


---

**Description**

Obtain the first derivative of the AR(1) process.

**Usage**

```
deriv_ar1(phi, sigma2, tau)
```

**Arguments**

phi            A double corresponding to the phi coefficient of an AR(1) process.  
 sigma2        A double corresponding to the error term of an AR(1) process.  
 tau            A vec containing the scales e.g.  $2^\tau$

**Value**

A matrix with the first column containing the partial derivative with respect to  $\phi$  and the second column contains the partial derivative with respect to  $\sigma^2$

**Process Haar WV First Derivative**

Taking the derivative with respect to  $\phi$  yields:

$$\frac{\partial}{\partial \phi} \nu_j^2(\phi, \sigma^2) = \frac{2\sigma^2 \left( (\phi^2 - 1) \tau_j \left( -2\phi^{\frac{\tau_j}{2}} + \phi^{\tau_j} - \phi - 1 \right) - (\phi(3\phi + 2) + 1) \left( -4\phi^{\frac{\tau_j}{2}} + \phi^{\tau_j} + 3 \right) \right)}{(\phi - 1)^4 (\phi + 1)^2 \tau_j^2}$$

Taking the derivative with respect to  $\sigma^2$  yields:

$$\frac{\partial}{\partial \sigma^2} \nu_j^2(\phi, \sigma^2) = \frac{(\phi^2 - 1) \tau_j + 2\phi \left( -4\phi^{\frac{\tau_j}{2}} + \phi^{\tau_j} + 3 \right)}{(\phi - 1)^3 (\phi + 1) \tau_j^2}$$

**Author(s)**

James Joseph Balamuta (JJB)

deriv\_arma11

*Analytic D matrix for ARMA(1,1) process***Description**

Obtain the first derivative of the ARMA(1,1) process.

**Usage**

```
deriv_arma11(phi, theta, sigma2, tau)
```

**Arguments**

phi	A double corresponding to the phi coefficient of an ARMA(1,1) process.
theta	A double corresponding to the theta coefficient of an ARMA(1,1) process.
sigma2	A double corresponding to the error term of an ARMA(1,1) process.
tau	A vec containing the scales e.g. $2^\tau$

**Value**

A matrix with:

- The **first** column containing the partial derivative with respect to  $\phi$ ;
- The **second** column containing the partial derivative with respect to  $\theta$ ;
- The **third** column contains the partial derivative with respect to  $\sigma^2$ .

**Process Haar WV First Derivative**

Taking the derivative with respect to  $\phi$  yields:

$$\frac{\partial}{\partial \phi} \nu_j^2(\phi, \theta, \sigma^2) = \frac{2\sigma^2}{(\phi - 1)^4(\phi + 1)^2\tau_j^2} \left( \begin{array}{l} \tau_j \left( -(\theta + 1)^2(\phi - 1)(\phi + 1)^2 - 2(\phi^2 - 1)(\theta + \phi)(\theta\phi + 1)\phi^{\frac{\tau_j}{2}-1} + (\phi^2 - 1) \right. \\ \left. - (\theta^2((3\phi + 2)\phi + 1) + 2\theta((\phi^2 + \phi + 3)\phi + 1) + (3\phi + 2)\phi + 1) \right) \end{array} \right)$$

Taking the derivative with respect to  $\theta$  yields:

$$\frac{\partial}{\partial \theta} \nu_j^2(\phi, \theta, \sigma^2) = \frac{2\sigma^2 \left( (\theta + 1)(\phi^2 - 1)\tau_j + (2\theta\phi + \phi^2 + 1) \left( \phi^{\tau_j} - 4\phi^{\frac{\tau_j}{2}} + 3 \right) \right)}{(\phi - 1)^3(\phi + 1)\tau_j^2}$$

Taking the derivative with respect to  $\sigma^2$  yields:

$$\frac{\partial}{\partial \sigma^2} \nu_j^2(\phi, \theta, \sigma^2) = \frac{2\sigma^2 \left( (\phi^2 - 1)\tau_j + 2\phi \left( \phi^{\tau_j} - 4\phi^{\frac{\tau_j}{2}} + 3 \right) \right)}{(\phi - 1)^3(\phi + 1)\tau_j^2}$$

**Author(s)**

James Joseph Balamuta (JJB)

---

deriv_dr	<i>Analytic D matrix for Drift (DR) Process</i>
----------	---

---

**Description**

Obtain the first derivative of the Drift (DR) process.

**Usage**

```
deriv_dr(omega, tau)
```

**Arguments**

omega	A double that is the slope of the drift.
tau	A vec containing the scales e.g. $2^\tau$

**Value**

A matrix with the first column containing the partial derivative with respect to  $\omega$ .

**Process Haar WV First Derivative**

Taking the derivative with respect to  $\omega$  yields:

$$\frac{\partial}{\partial \omega} \nu_j^2(\omega) = \frac{\tau_j^2 \omega}{8}$$

**Note:** We are taking the derivative with respect to  $\omega$  and not  $\omega^2$  as the  $\omega$  relates to the slope of the process and not the processes variance like RW and WN. As a result, a second derivative exists and is not zero.

**Author(s)**

James Joseph Balamuta (JJB)

---

deriv_ma1	<i>Analytic D matrix for MA(1) process</i>
-----------	--

---

**Description**

Obtain the first derivative of the MA(1) process.

**Usage**

```
deriv_ma1(theta, sigma2, tau)
```

**Arguments**

theta	A double corresponding to the theta coefficient of an MA(1) process.
sigma2	A double corresponding to the error term of an MA(1) process.
tau	A vec containing the scales e.g. $2^\tau$

**Value**

A matrix with the first column containing the partial derivative with respect to  $\theta$  and the second column contains the partial derivative with respect to  $\sigma^2$

**Process Haar WV First Derivative**

Taking the derivative with respect to  $\theta$  yields:

$$\frac{\partial}{\partial \theta} \nu_j^2(\theta, \sigma^2) = \frac{\sigma^2 (2(\theta + 1)\tau_j - 6)}{\tau_j^2}$$

Taking the derivative with respect to  $\sigma^2$  yields:

$$\frac{\partial}{\partial \sigma^2} \nu_j^2(\theta, \sigma^2) = \frac{(\theta + 1)^2 \tau_j - 6\theta}{\tau_j^2}$$

**Author(s)**

James Joseph Balamuta (JJB)

---

deriv\_qn

*Analytic D matrix for Quantization Noise (QN) Process*

---

**Description**

Obtain the first derivative of the Quantization Noise (QN) process.

**Usage**

deriv\_qn(tau)

**Arguments**

tau	A vec containing the scales e.g. $2^\tau$
-----	---

**Value**

A matrix with the first column containing the partial derivative with respect to  $Q^2$ .

**Process Haar WV First Derivative**

Taking the derivative with respect to  $Q^2$  yields:

$$\frac{\partial}{\partial Q^2} \nu_j^2(Q^2) = \frac{6}{\tau_j^2}$$

**Author(s)**

James Joseph Balamuta (JJB)

---

 deriv\_rw

---

*Analytic D matrix Random Walk (RW) Process*


---

**Description**

Obtain the first derivative of the Random Walk (RW) process.

**Usage**

deriv\_rw(tau)

**Arguments**

tau                    A vec containing the scales e.g.  $2^\tau$

**Value**

A matrix with the first column containing the partial derivative with respect to  $\gamma^2$ .

**Process Haar WV First Derivative**

Taking the derivative with respect to  $\gamma^2$  yields:

$$\frac{\partial}{\partial \gamma^2} \nu_j^2(\gamma^2) = \frac{\tau_j^2 + 2}{12\tau_j}$$

**Author(s)**

James Joseph Balamuta (JJB)

---

 deriv\_wn

*Analytic D Matrix for a Gaussian White Noise (WN) Process*


---

**Description**

Obtain the first derivative of the Gaussian White Noise (WN) process.

**Usage**

```
deriv_wn(tau)
```

**Arguments**

tau                    A vec containing the scales e.g.  $2^\tau$

**Value**

A matrix with the first column containing the partial derivative with respect to  $\sigma^2$ .

**Process Haar WV First Derivative**

Taking the derivative with respect to  $\sigma^2$  yields:

$$\frac{\partial}{\partial \sigma^2} \nu_j^2(\sigma^2) = \frac{1}{\tau_j}$$

**Author(s)**

James Joseph Balamuta (JJB)

---

 diag\_boxpierce

*Box-Pierce*


---

**Description**

Performs the Box-Pierce test to assess the Null Hypothesis of Independence in a Time Series

**Usage**

```
diag_boxpierce(x, order = NULL, stop_lag = 20, stdres = FALSE, plot = TRUE)
```

**Arguments**

x	An arima or data set.
order	An integer indicating the degrees of freedom. If 'x' is not a series of residuals, then set equal to 0.
stop_lag	An integer indicating the length of lags that should be calculated.
stdres	A boolean indicating whether to standardize the residualizes (e.g. $res/sd(res)$ ) or not.
plot	A logical. If TRUE (the default) a plot should be produced.

**Author(s)**

James Balamuta, Stéphane Guerrier, Yuming Zhang

---

diag_ljungbox	<i>Ljung-Box</i>
---------------	------------------

---

**Description**

Performs the Ljung-Box test to assess the Null Hypothesis of Independence in a Time Series

**Usage**

```
diag_ljungbox(x, order = NULL, stop_lag = 20, stdres = FALSE, plot = TRUE)
```

**Arguments**

x	An arima or data set.
order	An integer indicating the degrees of freedom. If 'x' is not a series of residuals, then set equal to 0.
stop_lag	An integer indicating the length of lags that should be calculated.
stdres	A boolean indicating whether to standardize the residualizes (e.g. $res/sd(res)$ ) or not.
plot	A logical. If TRUE (the default) a plot should be produced.

**Author(s)**

James Balamuta, Stéphane Guerrier, Yuming Zhang

diag\_plot

*Diagnostic Plot of Residuals***Description**

This function will plot 8 diagnostic plots to assess the model used to fit the data. These include: (1) residuals plot, (2) residuals vs fitted values, (3) histogram of distribution of standardized residuals, (4) Normal Q-Q plot of residuals, (5) ACF plot, (6) PACF plot, (7) Haar Wavelet Variance Representation, (8) Box test results.

**Usage**

```
diag_plot(Xt = NULL, model = NULL, resids = NULL, std = FALSE)
```

**Arguments**

Xt	The data used to construct said model.
model	A fitsimts, lm or gam object.
resids	A vector of residuals for diagnostics.
std	A boolean indicating whether we use standardized residuals for (1) residuals plot and (8) Box test results.

**Author(s)**

Yuming Zhang

diag\_portmanteau\_

*Portmanteau Tests***Description**

Performs the Portmanteau test to assess the Null Hypothesis of Independence in a Time Series

**Usage**

```
diag_portmanteau_(
  x,
  order = NULL,
  stop_lag = 20,
  stdres = FALSE,
  test = "Ljung-Box",
  plot = TRUE
)
```

**Arguments**

<code>x</code>	An arima or data set.
<code>order</code>	An integer indicating the degrees of freedom. If 'x' is not a series of residuals, then set equal to 0.
<code>stop_lag</code>	An integer indicating the length of lags that should be calculated.
<code>stdres</code>	A boolean indicating whether to standardize the residuals (e.g. $res/sd(res)$ ) or not.
<code>test</code>	A string indicating whether to perform Ljung-Box test or Box-Pierce test.
<code>plot</code>	A logical. If TRUE (the default) a plot should be produced.

**Author(s)**

James Balamuta, Stéphane Guerrier, Yuming Zhang

---

DR *Create an Drift (DR) Process*

---

**Description**

Sets up the necessary backend for the DR process.

**Usage**

```
DR(omega = NULL)
```

**Arguments**

<code>omega</code>	A double value for the slope of a DR process (see Note for details).
--------------------	--

**Value**

An S3 object with called `ts.model` with the following structure:

**process.desc** Used in summary: "DR"

**theta** slope

**print** String containing simplified model

**plength** Number of parameters

**obj.desc** y desc replicated x times

**obj** Depth of parameters e.g. `list(1)`

**starting** Guess starting values? TRUE or FALSE (e.g. specified value)

**Note**

We consider the following model:

$$Y_t = \omega t$$

**Author(s)**

James Balamuta

**Examples**

```
DR()
DR(omega=3.4)
```

---

dr\_to\_wv
*Drift to WV***Description**

This function compute the WV (haar) of a Drift process

**Usage**

```
dr_to_wv(omega, tau)
```

**Arguments**

omega	A double corresponding to the slope of the drift
tau	A vec containing the scales e.g. $2^\tau$

**Value**

A vec containing the wavelet variance of the drift.

**Process Haar Wavelet Variance Formula**

The Drift (DR) process has a Haar Wavelet Variance given by:

$$\nu_j^2(\omega) = \frac{\tau_j^2 \omega^2}{16}$$

---

 estimate

*Fit a Time Series Model to Data*


---

**Description**

This function can fit a time series model to data using different methods.

**Usage**

```
estimate(model, Xt, method = "mle", demean = TRUE)
```

**Arguments**

model	A time series model.
Xt	A vector of time series data.
method	A string indicating the method used for model fitting. Supported methods include mle, yule-walker, gmwm and rgmwm.
demean	A boolean indicating whether the model includes a mean / intercept term or not.

**Author(s)**

Stéphane Guerrier and Yuming Zhang

**Examples**

```
Xt = gen_gts(300, AR(phi = c(0, 0, 0.8), sigma2 = 1))
plot(Xt)
estimate(AR(3), Xt)
```

```
Xt = gen_gts(300, MA(theta = 0.5, sigma2 = 1))
plot(Xt)
estimate(MA(1), Xt, method = "gmwm")
```

```
Xt = gen_gts(300, ARMA(ar = c(0.8, -0.5), ma = 0.5, sigma2 = 1))
plot(Xt)
estimate(ARMA(2,1), Xt, method = "rgmwm")
```

```
Xt = gen_gts(300, ARIMA(ar = c(0.8, -0.5), i = 1, ma = 0.5, sigma2 = 1))
plot(Xt)
estimate(ARIMA(2,1,1), Xt, method = "mle")
```

```
Xt = gen_gts(1000, SARIMA(ar = c(0.5, -0.25), i = 0, ma = 0.5, sar = -0.8,
si = 1, sma = 0.25, s = 24, sigma2 = 1))
plot(Xt)
estimate(SARIMA(ar = 2, i = 0, ma = 1, sar = 1, si = 1, sma = 1, s = 24), Xt,
method = "rgmwm")
```

---

 evaluate

*Evaluate a time series or a list of time series models*


---

**Description**

This function calculates AIC, BIC and HQ or the MAPE for a list of time series models. This function currently only supports models estimated by the MLE.

**Usage**

```
evaluate(
  models,
  Xt,
  criterion = "IC",
  start = 0.8,
  demean = TRUE,
  print = TRUE
)
```

**Arguments**

models	A time series model or a list of time series models.
Xt	A time series (i.e gts object).
criterion	Either "IC" for AIC, BIC and HQ or "MAPE" for MAPE.
start	A numeric indicating the starting proportion of the data that is used for prediction (assuming criterion = "MAPE").
demean	A boolean indicating whether the model includes a mean / intercept term or not.
print	logical. If TRUE (the default) results are printed.

**Value**

AIC, BIC and HQ or MAPE

**Author(s)**

Stéphane Guerrier

**Examples**

```
set.seed(18)
n = 300
Xt = gen_gts(n, AR(phi = c(0, 0, 0.8), sigma2 = 1))
evaluate(AR(1), Xt)
evaluate(list(AR(1), AR(3), MA(3), ARMA(1,2),
SARIMA(ar = 1, i = 0, ma = 1, sar = 1, si = 1, sma = 1, s = 12)), Xt)
evaluate(list(AR(1), AR(3)), Xt, criterion = "MAPE")
```

**Description**

Definition of a Fractional Gaussian Noise (FGN) Process

**Usage**

```
FGN(sigma2 = 1, H = 0.9999)
```

**Arguments**

sigma2	A double.
H	A double.

**Value**

An S3 object containing the specified ts.model with the following structure:

**process.desc** Used in summary: "SIGMA2","H"

**theta** Parameter vector including  $\sigma^2$ ,  $H$

**plength** Number of parameters

**print** String containing simplified model

**desc** "FGN"

**obj.desc** Depth of Parameters e.g. list(1,1)

**starting** Find starting values? TRUE or FALSE (e.g. specified value)

**Author(s)**

Lionel Voirol, Davide Cucci

**Examples**

```
FGN()  
FGN(sigma2 = 1, H = 0.9999)
```

---

gen_ar1blocks	<i>Generate AR(1) Block Process</i>
---------------	-------------------------------------

---

### Description

This function allows us to generate a non-stationary AR(1) block process.

### Usage

```
gen_ar1blocks(phi, sigma2, n_total, n_block, scale = 10,  
title = NULL, seed = 135, ...)
```

### Arguments

phi	A double value for the autocorrection parameter $\phi$ .
sigma2	A double value for the variance parameter $\sigma^2$ .
n_total	An integer indicating the length of the simulated AR(1) block process.
n_block	An integer indicating the length of each block of the AR(1) block process.
scale	An integer indicating the number of levels of decomposition. The default value is 10.
title	A string indicating the name of the time series data.
seed	An integer defined for simulation replication purposes.
...	Additional parameters.

### Value

A vector containing the AR(1) block process.

### Note

This function generates a non-stationary AR(1) block process whose theoretical maximum overlapping allan variance (MOAV) is different from the theoretical MOAV of a stationary AR(1) process. This difference in the value of the allan variance between stationary and non-stationary processes has been shown through the calculation of the theoretical allan variance given in "A Study of the Allan Variance for Constant-Mean Non-Stationary Processes" by Xu et al. (IEEE Signal Processing Letters, 2017), preprint available: <https://arxiv.org/abs/1702.07795>.

### Author(s)

Yuming Zhang and Haotian Xu

**Examples**

```
Xt = gen_ar1blocks(phi = 0.9, sigma2 = 1,
n_total = 1000, n_block = 10, scale = 100)
plot(Xt)
```

```
Yt = gen_ar1blocks(phi = 0.5, sigma2 = 5, n_total = 800,
n_block = 20, scale = 50)
plot(Yt)
```

---

gen_bi	<i>Generate Bias-Instability Process</i>
--------	--

---

**Description**

This function allows to generate a non-stationary bias-instability process.

**Usage**

```
gen_bi(sigma2, n_total, n_block, title = NULL, seed = 135, ...)
```

**Arguments**

sigma2	A double value for the variance parameter $\sigma^2$ .
n_total	An integer indicating the length of the simulated bias-instability process.
n_block	An integer indicating the length of each block of the bias-instability process.
title	A string defining the name of the time series data.
seed	An integer defined for simulation replication purposes.
...	Additional parameters.

**Value**

A vector containing the bias-instability process.

**Note**

This function generates a non-stationary bias-instability process whose theoretical maximum overlapping allan variance (MOAV) is close to the theoretical MOAV of the best approximation of this process through a stationary AR(1) process over some scales. However, this approximation is not good enough when considering the logarithmic representation of the allan variance. Therefore, the exact form of the allan variance of this non-stationary process allows us to better interpret the signals characterized by bias-instability, as shown in "A Study of the Allan Variance for Constant-Mean Non-Stationary Processes" by Xu et al. (IEEE Signal Processing Letters, 2017), preprint available: <https://arxiv.org/abs/1702.07795>.

**Author(s)**

Yuming Zhang

**Examples**

```
Xt = gen_bi(sigma2 = 1, n_total = 1000, n_block = 10)
plot(Xt)
```

```
Yt = gen_bi(sigma2 = 0.8, n_total = 800, n_block = 20,
title = "non-stationary bias-instability process")
plot(Yt)
```

---

gen\_gts

---

*Simulate a simts TS object using a theoretical model*


---

**Description**

Create a gts object based on a time series model.

**Usage**

```
gen_gts(
  n,
  model,
  start = 0,
  end = NULL,
  freq = 1,
  unit_ts = NULL,
  unit_time = NULL,
  name_ts = NULL,
  name_time = NULL
)
```

**Arguments**

n	An integer containing the length of the time series.
model	A ts.model or simts object containing the available models in the simts package.
start	A numeric that provides the time of the first observation.
end	A numeric that provides the time of the last observation.
freq	A numeric that provides the rate of samples. Default value is 1.
unit_ts	A string that contains the unit expression of the time series. Default value is NULL.
unit_time	A string that contains the unit expression of the time. Default value is NULL.
name_ts	A string that provides an identifier for the time series data. Default value is NULL.
name_time	A string that provides an identifier for the time. Default value is NULL.

**Details**

This function accepts either a `ts.model` object (e.g. `AR1(phi = .3, sigma2 = 1) + WN(sigma2 = 1)`) or a `simts` object.

**Value**

A `gts` object

**Author(s)**

James Balamuta and Wenchao Yang

**Examples**

```
# Set seed for reproducibility
set.seed(1336)
n = 1000

# AR1 + WN
model = AR1(phi = .5, sigma2 = .1) + WN(sigma2=1)
x = gen_gts(n, model)
plot(x)

# Reset seed
set.seed(1336)

# GM + WN
# Convert from AR1 to GM values
m = ar1_to_gm(c(.5, .1), 10)

# Beta = 6.9314718, Sigma2_gm = 0.1333333
model = GM(beta = m[1], sigma2_gm = m[2]) + WN(sigma2=1)
x2 = gen_gts(n, model, freq = 10, unit_time = 'sec')
plot(x2)

# Same time series
all.equal(x, x2, check.attributes = FALSE)
```

---

`gen_lts`*Generate a Latent Time Series Object Based on a Model*

---

**Description**

Simulate a `lts` object based on a supplied time series model.

**Usage**

```

gen_lts(
  n,
  model,
  start = 0,
  end = NULL,
  freq = 1,
  unit_ts = NULL,
  unit_time = NULL,
  name_ts = NULL,
  name_time = NULL,
  process = NULL
)

```

**Arguments**

n	An interger indicating the amount of observations generated in this function.
model	A <code>ts.model</code> or <code>simts</code> object containing one of the allowed models.
start	A numeric that provides the time of the first observation.
end	A numeric that provides the time of the last observation.
freq	A numeric that provides the rate/frequency at which the time series is sampled. The default value is 1.
unit_ts	A string that contains the unit of measure of the time series. The default value is NULL.
unit_time	A string that contains the unit of measure of the time. The default value is NULL.
name_ts	A string that provides an identifier for the time series data. Default value is NULL.
name_time	A string that provides an identifier for the time. Default value is NULL.
process	A vector that contains model names of each column in the data object where the last name is the sum of the previous names.

**Details**

This function accepts either a `ts.model` object (e.g. `AR1(phi = .3, sigma2 = 1) + WN(sigma2 = 1)`) or a `simts` object.

**Value**

A `lts` object with the following attributes:

**start** The time of the first observation.

**end** The time of the last observation.

**freq** Numeric representation of the sampling frequency/rate.

**unit** A string reporting the unit of measurement.

**name** Name of the generated dataset.

**process** A vector that contains model names of decomposed and combined processes

**Author(s)**

James Balamuta, Wenchao Yang, and Justin Lee

**Examples**

```
# AR
set.seed(1336)
model = AR1(phi = .99, sigma2 = 1) + WN(sigma2 = 1)
test = gen_lts(1000, model)
plot(test)
```

---

gen\_nswn

*Generate Non-Stationary White Noise Process*

---

**Description**

This function allows to generate a non-stationary white noise process.

**Usage**

```
gen_nswn(n_total, title = NULL, seed = 135, ...)
```

**Arguments**

n_total	An integer indicating the length of the simulated non-stationary white noise process.
title	A string defining the name of the time series data.
seed	An integer defined for simulation replication purposes.
...	Additional parameters.

**Value**

A vector containing the non-stationary white noise process.

**Note**

This function generates a non-stationary white noise process whose theoretical maximum overlapping allan variance (MOAV) corresponds to the theoretical MOAV of the stationary white noise process. This example confirms that the allan variance is unable to distinguish between a stationary white noise process and a white noise process whose second-order behavior is non-stationary, as pointed out in the paper "A Study of the Allan Variance for Constant-Mean Non-Stationary Processes" by Xu et al. (IEEE Signal Processing Letters, 2017), preprint available: <https://arxiv.org/abs/1702.07795>.

**Author(s)**

Yuming Zhang

**Examples**

```
Xt = gen_nswn(n_total = 1000)
plot(Xt)

Yt = gen_nswn(n_total = 2000, title = "non-stationary
white noise process", seed = 1960)
plot(Yt)
```

GM

*Create a Gauss-Markov (GM) Process***Description**

Sets up the necessary backend for the GM process.

**Usage**

```
GM(beta = NULL, sigma2_gm = 1)
```

**Arguments**

**beta** A double value for the  $\beta$  of an GM process (see Note for details).  
**sigma2\_gm** A double value for the variance,  $\sigma_{gm}^2$ , of a GM process (see Note for details).

**Details**

When supplying values for  $\beta$  and  $\sigma_{gm}^2$ , these parameters should be of a GM process and NOT of an AR1. That is, do not supply AR1 parameters such as  $\phi$ ,  $\sigma^2$ .

Internally, GM parameters are converted to AR1 using the 'freq' supplied when creating data objects ([gts](#)) or specifying a 'freq' parameter in `simts` or `simts.imu`.

The 'freq' of a data object takes precedence over the 'freq' set when modeling.

**Value**

An S3 object with called `ts.model` with the following structure:

**process.desc** Used in summary: "BETA", "SIGMA2"

**theta**  $\beta$ ,  $\sigma_{gm}^2$

**plength** Number of parameters

**print** String containing simplified model

**desc** "GM"

**obj.desc** Depth of parameters e.g. `list(1,1)`

**starting** Guess starting values? TRUE or FALSE (e.g. specified value)

**Note**

We consider the following model:

$$X_t = e^{(-\beta)} X_{t-1} + \varepsilon_t$$

, where  $\varepsilon_t$  is iid from a zero mean normal distribution with variance  $\sigma^2(1 - e^{2\beta})$ .

**Author(s)**

James Balamuta

**Examples**

```
GM()
GM(beta=.32, sigma2_gm=1.3)
```

---

gmwm

*Generalized Method of Wavelet Moments (GMWM)*


---

**Description**

Performs estimation of time series models by using the GMWM estimator.

**Usage**

```
gmwm(
  model,
  data,
  model.type = "imu",
  compute.v = "auto",
  robust = FALSE,
  eff = 0.6,
  alpha = 0.05,
  seed = 1337,
  G = NULL,
  K = 1,
  H = 100,
  freq = 1
)
```

**Arguments**

model	A <code>ts.model</code> object containing one of the allowed models.
data	A matrix or <code>data.frame</code> object with only column (e.g. $N \times 1$ ), a <code>lts</code> object, or a <code>gts</code> object.
model.type	A string containing the type of GMWM needed: "imu" or "ssm".

compute.v	A string indicating the type of covariance matrix solver. Valid values are: "fast", "bootstrap", "diag" (asymptotic diag), "full" (asymptotic full). By default, the program will fit a "fast" model.
robust	A boolean indicating whether to use the robust computation (TRUE) or not (FALSE).
eff	A double between 0 and 1 that indicates the efficiency.
alpha	A double between 0 and 1 that correspondings to the $\frac{\alpha}{2}$ value for the wavelet confidence intervals.
seed	An integer that controls the reproducibility of the auto model selection phase.
G	An integer to sample the space for IMU and SSM models to ensure optimal identitability.
K	An integer that controls how many times the bootstrapping procedure will be initiated.
H	An integer that indicates how many different samples the bootstrap will be collect.
freq	A double that indicates the sampling frequency. By default, this is set to 1 and only is important if GM() is in the model

## Details

This function is under work. Some of the features are active. Others... Not so much.

The V matrix is calculated by:  $diag \left[ (Hi - Lo)^2 \right]$ .

The function is implemented in the following manner: 1. Calculate MODWT of data with levels = floor(log2(data)) 2. Apply the brick.wall of the MODWT (e.g. remove boundary values) 3. Compute the empirical wavelet variance (WV Empirical). 4. Obtain the V matrix by squaring the difference of the WV Empirical's Chi-squared confidence interval (hi - lo)^2 5. Optimize the values to obtain  $\hat{\theta}$  6. If FAST = TRUE, return these results. Else, continue.

Loop k = 1 to K Loop h = 1 to H 7. Simulate  $x_t$  under  $F_{\hat{\theta}}$  8. Compute WV Empirical END 9. Calculate the covariance matrix 10. Optimize the values to obtain  $\hat{\theta}$  END 11. Return optimized values.

The function estimates a variety of time series models. If type = "imu" or "ssm", then parameter vector should indicate the characters of the models that compose the latent or state-space model. The model options are:

- "AR1": a first order autoregressive process with parameters  $(\phi, \sigma^2)$
- "GM": a guass-markov process  $(\beta, \sigma_{gm}^2)$
- "ARMA": an autoregressive moving average process with parameters  $(\phi_p, \theta_q, \sigma^2)$
- "DR": a drift with parameter  $\omega$
- "QN": a quantization noise process with parameter  $Q$
- "RW": a random walk process with parameter  $\sigma^2$
- "WN": a white noise process with parameter  $\sigma^2$

If only an ARMA() term is supplied, then the function takes conditional least squares as starting values If robust = TRUE the function takes the robust estimate of the wavelet variance to be used in the GMWM estimation procedure.

**Value**

A gmwm object with the structure:

- estimate: Estimated Parameters Values from the GMWM Procedure
- init.guess: Initial Starting Values given to the Optimization Algorithm
- vv.empir: The data's empirical wavelet variance
- ci\_low: Lower Confidence Interval
- ci\_high: Upper Confidence Interval
- orgV: Original V matrix
- V: Updated V matrix (if bootstrapped)
- omega: The V matrix inversed
- obj.fun: Value of the objective function at Estimated Parameter Values
- theo: Summed Theoretical Wavelet Variance
- decomp.theo: Decomposed Theoretical Wavelet Variance by Process
- scales: Scales of the GMWM Object
- robust: Indicates if parameter estimation was done under robust or classical
- eff: Level of efficiency of robust estimation
- model.type: Models being guessed
- compute.v: Type of V matrix computation
- augmented: Indicates moments have been augmented
- alpha: Alpha level used to generate confidence intervals
- expect.diff: Mean of the First Difference of the Signal
- N: Length of the Signal
- G: Number of Guesses Performed
- H: Number of Bootstrap replications
- K: Number of V matrix bootstraps
- model: `ts.model` supplied to gmwm
- model.hat: A new value of `ts.model` object supplied to gmwm
- starting: Indicates whether the procedure used the initial guessing approach
- seed: Randomization seed used to generate the guessing values
- freq: Frequency of data

gmwm\_imu

*GMWM for (Robust) Inertial Measurement Units (IMUs)***Description**

Performs the GMWM estimation procedure using a parameter transform and sampling scheme specific to IMUs.

**Usage**

```
gmwm_imu(model, data, compute.v = "fast", robust = F, eff = 0.6, ...)
```

**Arguments**

model	A <code>ts.model</code> object containing one of the allowed models.
data	A matrix or data.frame object with only column (e.g. $N \times 1$ ), or a <code>lts</code> object, or a <code>gts</code> object.
compute.v	A string indicating the type of covariance matrix solver. "fast", "bootstrap", "asyp.diag", "asyp.comp", "fft"
robust	A boolean indicating whether to use the robust computation (TRUE) or not (FALSE).
eff	A double between 0 and 1 that indicates the efficiency.
...	Other arguments passed to the main gmwm function

**Details**

This version of the gmwm function has customized settings ideal for modeling with an IMU object. If you seek to model with an Gauss Markov, GM, object. Please note results depend on the freq specified in the data construction step within the imu. If you wish for results to be stable but lose the ability to interpret with respect to freq, then use AR1 terms.

**Value**

A gmwm object with the structure:

- estimateEstimated Parameters Values from the GMWM Procedure
- init.guessInitial Starting Values given to the Optimization Algorithm
- ww.empirThe data's empirical wavelet variance
- ci\_lowLower Confidence Interval
- ci\_highUpper Confidence Interval
- orgVOriginal V matrix
- VUpdated V matrix (if bootstrapped)
- omegaThe V matrix inverted
- obj.funValue of the objective function at Estimated Parameter Values

- theoSummed Theoretical Wavelet Variance
- decomp.theoDecomposed Theoretical Wavelet Variance by Process
- scalesScales of the GMWM Object
- robustIndicates if parameter estimation was done under robust or classical
- effLevel of efficiency of robust estimation
- model.typeModels being guessed
- compute.vType of V matrix computation
- augmentedIndicates moments have been augmented
- alphaAlpha level used to generate confidence intervals
- expect.diffMean of the First Difference of the Signal
- NLength of the Signal
- GNumber of Guesses Performed
- HNumber of Bootstrap replications
- KNumber of V matrix bootstraps
- model.ts.model supplied to gmwm
- model.hatA new value of ts.model object supplied to gmwm
- startingIndicates whether the procedure used the initial guessing approach
- seedRandomization seed used to generate the guessing values
- freqFrequency of data

---

gts

---

*Create a simts TS object using time series data*


---

### Description

Takes a time series and turns it into a time series oriented object that can be used for summary and graphing functions in the simts package.

### Usage

```
gts(
  data,
  start = 0,
  end = NULL,
  freq = 1,
  unit_ts = NULL,
  unit_time = NULL,
  name_ts = NULL,
  name_time = NULL,
  data_name = NULL,
  Time = NULL,
  time_format = NULL
)
```

**Arguments**

<code>data</code>	A one-column matrix, <code>data.frame</code> , or a numeric vector.
<code>start</code>	A numeric that provides the time of the first observation.
<code>end</code>	A numeric that provides the time of the last observation.
<code>freq</code>	A numeric that provides the rate/frequency at which the time series is sampled. The default value is 1.
<code>unit_ts</code>	A string that contains the unit of measure of the time series. The default value is NULL.
<code>unit_time</code>	A string that contains the unit of measure of the time. The default value is NULL.
<code>name_ts</code>	A string that provides an identifier for the time series data. Default value is NULL.
<code>name_time</code>	A string that provides an identifier for the time. Default value is NULL.
<code>data_name</code>	A string that contains the name of the time series data.
<code>Time</code>	A numeric or character vector containing the times of observations. Default value is NULL. See <code>x</code> object in <code>as.Date</code> function.
<code>time_format</code>	A string specifying the format of 'Time'. If not provided, 'Time' is assumed to be all integers. Default value is NULL. See <code>format</code> argument in <code>as.Date</code> function.

**Value**

A `gts` object

**Author(s)**

James Balamuta and Wenchao Yang

**Examples**

```
m = data.frame(rnorm(50))
x = gts(m, unit_time = 'sec', name_ts = 'example')
plot(x)

x = gen_gts(50, WN(sigma2 = 1))
x = gts(x, freq = 100, unit_time = 'sec')
plot(x)
```

---

hydro

*Mean Monthly Precipitation, from 1907 to 1972*

---

**Description**

Hydrology data that indicates a robust approach may be preferred to a classical approach when estimating time series.

**Usage**

hydro

**Format**

A time series object with frequency 12 starting at 1907 and going to 1972 for a total of 781 observations.

**Source**

datamarket, mean-monthly-precipitation-1907-1972

---

imu

*Create an IMU Object*

---

**Description**

Builds an IMU object that provides the program with gyroscope, accelerometer, and axis information per column in the dataset.

**Usage**

```
imu(  
  data,  
  gyros = NULL,  
  accels = NULL,  
  axis = NULL,  
  freq = NULL,  
  unit = NULL,  
  name = NULL  
)
```

**Arguments**

<code>data</code>	A vector which contains data, or a matrix or <code>data.frame</code> which contains the data in each column.
<code>gyros</code>	A vector that contains the index of columns where gyroscope data (such as Gyro. X, Gyro. Y and Gyro. Z) is placed.
<code>accels</code>	A vector that contains the index of columns where accelerometer data (such as Accel. X, Accel. Y and Accel. Z) is placed.
<code>axis</code>	A vector that indicates the axes, such as 'X', 'Y', 'Z'. Please supply the axes for gyroscope data before that for accelerometer data, if gyroscope data exists.
<code>freq</code>	An integer that provides the frequency for the data.
<code>unit</code>	A string that contains the unit expression of the frequency. Default value is NULL.
<code>name</code>	A string that provides an identifier to the data. Default value is NULL.

**Details**

`data` can be a numeric vector, matrix or data frame.

`gyros` and `accels` cannot be NULL at the same time, but it will be fine if one of them is NULL. In the new implementation, the length of `gyros` and `accels` do not need to be equal.

In `axis`, duplicate elements are not allowed for each sensor. In the new implementation, please specify the axis for each column of data. `axis` will be automatically generated if there are less than or equal to 3 axes for each sensor.

**Value**

An `imu` object in the following attributes:

**sensor** A vector that indicates whether data contains gyroscope sensor, accelerometer sensor, or both.

**num.sensor** A vector that indicates how many columns of data are for gyroscope sensor and accelerometer sensor.

**axis** Axis value such as 'X', 'Y', 'Z'.

**freq** Observations per second.

**unit** String representation of the unit.

**name** Name of the dataset.

**Author(s)**

James Balamuta and Wenchao Yang

**Examples**

```
## Not run:
if(!require("imudata")){
  install_imudata()
  library("imudata")
}

data(imu6)

# Example 1 - Only gyros
test1 = imu(imu6, gyros = 1:3, axis = c('X', 'Y', 'Z'), freq = 100)
df1 = wvar.imu(test1)
plot(df1)

# Example 2 - One gyro and one accelerometer
test2 = imu(imu6, gyros = 1, accels = 4, freq = 100)
df2 = wvar.imu(test2)
plot(df2)

# Example 3 - 3 gyros and 3 accelerometers
test3 = imu(imu6, gyros = 1:3, accels = 4:6, axis =
           c('X', 'Y', 'Z', 'X', 'Y', 'Z'), freq = 100)
df3 = wvar.imu(test3)
plot(df3)

# Example 4 - Custom axis
test4 = imu(imu6, gyros = 1:2, accels = 4:6, axis =
           c('X', 'Y', 'X', 'Y', 'Z'), freq = 100)
df4 = wvar.imu(test4)
plot(df4)

## End(Not run)
```

---

imu\_time

*Pulls the IMU time from the IMU object*


---

**Description**

Helper function for the IMU object to access rownames() with a numeric conversion.

**Usage**

```
imu_time(x)
```

**Arguments**

x                    A imu object

**Value**

A vector with numeric information.

---

`is.gts`*Is simts Object*

---

**Description**

Is the object a gts, imu, or lts object?

**Usage**`is.gts(x)``is.imu(x)``is.lts(x)``is.ts.model(x)`**Arguments**

`x` A gts, imu, lts object.

**Details**

Uses [inherits](#) over [is](#) for speed.

**Value**

A logical value that indicates whether the object is of that class (TRUE) or not (FALSE).

**Author(s)**

James Balamuta

lts

*Generate a Latent Time Series Object from Data***Description**

Create a `lts` object based on a supplied matrix or data frame. The latent time series is obtained by the sum of underlying time series.

**Usage**

```
lts(
  data,
  start = 0,
  end = NULL,
  freq = 1,
  unit_ts = NULL,
  unit_time = NULL,
  name_ts = NULL,
  name_time = NULL,
  process = NULL
)
```

**Arguments**

<code>data</code>	A multiple-column matrix or data frame. It must contain at least 3 columns of which the last represents the latent time series obtained through the sum of the previous columns.
<code>start</code>	A numeric that provides the time of the first observation.
<code>end</code>	A numeric that provides the time of the last observation.
<code>freq</code>	A numeric that provides the rate/frequency at which the time series is sampled. The default value is 1.
<code>unit_ts</code>	A string that contains the unit of measure of the time series. The default value is NULL.
<code>unit_time</code>	A string that contains the unit of measure of the time. The default value is NULL.
<code>name_ts</code>	A string that provides an identifier for the time series data. Default value is NULL.
<code>name_time</code>	A string that provides an identifier for the time. Default value is NULL.
<code>process</code>	A vector that contains model names of each column in the data object where the last name is the sum of the previous names.

**Value**

A `lts` object

**Author(s)**

Wenchao Yang and Justin Lee

**Examples**

```
model1 = AR1(phi = .99, sigma2 = 1)
model2 = WN(sigma2 = 1)
col1 = gen_gts(1000, model1)
col2 = gen_gts(1000, model2)
testMat = cbind(col1, col2, col1+col2)
testLts = lts(testMat, unit_time = 'sec', process = c('AR1', 'WN', 'AR1+WN'))
plot(testLts)
```

---

M

*Definition of a Mean deterministic vector returned by the matrix by vector product of matrix  $X$  and vector  $\beta$*

---

**Description**

Definition of a Mean deterministic vector returned by the matrix by vector product of matrix  $X$  and vector  $\beta$

**Usage**

`M(X, beta)`

**Arguments**

`X` A Matrix with dimension  $n \times p$ .  
`beta` A vector with dimension  $p \times 1$

**Value**

An S3 object containing the specified ts.model with the following structure:

**process.desc** Used in summary: "X","BETA"

**theta** Matrix X, vector beta

**plength** Number of parameters

**print** String containing simplified model

**desc** "M"

**obj.desc** Depth of Parameters e.g. list(1,1)

**starting** Find starting values? TRUE or FALSE (e.g. specified value)

**Author(s)**

Lionel Voirol, Davide Cucci

**Examples**

```
X = matrix(rnorm(15*5), nrow = 15, ncol = 5)
beta=seq(5)
M(X = X, beta = beta)
```

MA

*Create an Moving Average Q [MA(Q)] Process***Description**

Sets up the necessary backend for the MA(Q) process.

**Usage**

```
MA(theta = NULL, sigma2 = 1)
```

**Arguments**

**theta** A double value for the parameter  $\theta$  (see Note for details).  
**sigma2** A double value for the variance parameter  $\sigma^2$  (see Note for details).

**Value**

An S3 object with called `ts.model` with the following structure:

**process.desc** Used in summary: "MA-1", "MA-2", ..., "MA-Q", "SIGMA2"

**theta**  $\theta_1, \theta_2, \dots, \theta_q, \sigma^2$

**plength** Number of parameters

**desc** "MA"

**print** String containing simplified model

**obj.desc** Depth of parameters e.g. `list(q,1)`

**starting** Guess starting values? TRUE or FALSE (e.g. specified value)

**Note**

We consider the following model:

$$X_t = \sum_{j=1}^q \theta_j \varepsilon_{t-1} + \varepsilon_t$$

, where  $\varepsilon_t$  is iid from a zero mean normal distribution with variance  $\sigma^2$ .

**Author(s)**

James Balamuta

**Examples**

```
MA(1) # One theta
MA(2) # Two thetas!

MA(theta=.32, sigma=1.3) # 1 theta with a specific value.
MA(theta=c(.3,.5), sigma=.3) # 2 thetas with specific values.
```

MA1

*Definition of an Moving Average Process of Order 1***Description**

Definition of an Moving Average Process of Order 1

**Usage**

```
MA1(theta = NULL, sigma2 = 1)
```

**Arguments**

**theta** A double value for the parameter  $\theta$  (see Note for details).  
**sigma2** A double value for the variance parameter  $\sigma^2$  (see Note for details).

**Value**

An S3 object with called ts.model with the following structure:

**process.desc** Used in summary: "MA1","SIGMA2"

**theta**  $\theta, \sigma^2$

**plength** Number of parameters

**print** String containing simplified model

**desc** "MA1"

**obj.desc** Depth of parameters e.g. list(1,1)

**starting** Guess starting values? TRUE or FALSE (e.g. specified value)

**Note**

We consider the following model:

$$X_t = \theta\varepsilon_{t-1} + \varepsilon_t$$

, where  $\varepsilon_t$  is iid from a zero mean normal distribution with variance  $\sigma^2$ .

**Author(s)**

James Balamuta

**Examples**

```
MA1()
MA1(theta = .32, sigma2 = 1.3)
```

---

ma1\_to\_wv

---

*Moving Average Order 1 (MA(1)) to WV*


---

**Description**

This function computes the WV (haar) of a Moving Average order 1 (MA1) process.

**Usage**

```
ma1_to_wv(theta, sigma2, tau)
```

**Arguments**

theta	A double corresponding to the moving average term.
sigma2	A double the variance of the process.
tau	A vec containing the scales e.g. $2^{\tau}$

**Details**

This function is significantly faster than its generalized counter part [arma\\_to\\_wv](#).

**Value**

A vec containing the wavelet variance of the MA(1) process.

**Process Haar Wavelet Variance Formula**

The Moving Average Order 1 (MA(1)) process has a Haar Wavelet Variance given by:

$$\nu_j^2(\theta, \sigma^2) = \frac{((\theta + 1)^2 \tau_j - 6\theta) \sigma^2}{\tau_j^2}$$

---

 make\_frame

*Default utility function for various plots titles*


---

**Description**

Adds title, grid, and required x- and y-axes.

**Usage**

```
make_frame(
  x_range,
  y_range,
  xlab,
  ylab,
  main = "",
  mar = c(5.1, 5.1, 1, 2.1),
  add_axis_x = TRUE,
  add_axis_y = TRUE,
  col_box = "black",
  col_grid = "grey95",
  col_band = "grey95",
  col_title = "black",
  add_band = TRUE,
  title_band_width = 0.09,
  grid_lty = 1
)
```

**Arguments**

x_range	A numeric providing the range of values for the x-axis.
y_range	A numeric providing the range of values for the y-axis.
xlab	A string that gives a title for the x-axis.
ylab	A string that gives a title for the y-axis.
main	A string that gives an overall title for the plot. Default is an empty string.
mar	A vector indicating overall margin values for the plot.
add_axis_x	A boolean indicating whether a x-axis should be added.
add_axis_y	A boolean indicating whether a y-axis should be added.
col_box	A string indicating the color for the title box.
col_grid	A string indicating the color of the grid for the plot.
col_band	A string indicating the color of the band.
col_title	A string indicating the color of the plot title.
add_band	A boolean indicating whether there should be a band.
title_band_width	A double providing the value of the band width. Default is 0.09.
grid_lty	A integer indicating the line type of the grid lines.

**Value**

Added title, grid, and axes.

**Author(s)**

Stephane Guerrier and Justin Lee

**Examples**

```
make_frame(x_range = c(0, 1), y_range = c(0, 1), xlab = "my xlab",
           ylab = "my ylab", main = "my title")
```

```
make_frame(x_range = c(0, 1), y_range = c(0, 1), xlab = "my xlab",
           ylab = "my ylab", add_band = FALSE)
```

```
make_frame(x_range = c(0, 1), y_range = c(0, 1), xlab = "my xlab",
           ylab = "my ylab", main = "my title", col_band = "blue3",
           col_title = "white", col_grid = "lightblue", grid_lty = 3)
```

```
make_frame(x_range = c(0, 1), y_range = c(0, 1), xlab = "my xlab",
           ylab = "my ylab", main = "my title", col_band = "blue3",
           col_title = "white", col_grid = "lightblue", grid_lty = 3,
           title_band_width = 0.18)
```

---

MAPE

---

*Median Absolute Prediction Error*


---

**Description**

This function calculates Median Absolute Prediction Error (MAPE), which assesses the prediction performance with respect to point forecasts of a given model. It is calculated based on one-step ahead prediction and reforecasting.

**Usage**

```
MAPE(model, Xt, start = 0.8, plot = TRUE)
```

**Arguments**

model	A time series model.
Xt	A vector of time series data.
start	A numeric indicating the starting proportion of the data that is used for prediction.
plot	A boolean indicating whether a model accuracy plot based on MAPE is returned or not.

**Value**

The MAPE calculated based on one-step ahead prediction and reforecasting is returned along with its standard deviation.

**Author(s)**

Stéphane Guerrier and Yuming Zhang

---

 MAT

*Definition of a Matérn Process*


---

**Description**

Definition of a Matérn Process

**Usage**

```
MAT(sigma2 = 1, lambda = 0.35, alpha = 0.9)
```

**Arguments**

sigma2	A double.
lambda	A double.
alpha	A double.

**Value**

An S3 object containing the specified ts.model with the following structure:

**process.desc** Used in summary: "SIGMA2","LAMBDA""ALPHA"

**theta** Parameter vector including  $\sigma^2, \lambda, \alpha$

**plength** Number of parameters

**print** String containing simplified model

**desc** "MAT"

**obj.desc** Depth of Parameters e.g. list(1,1,1)

**starting** Find starting values? TRUE or FALSE (e.g. specified value)

**Author(s)**

Lionel Voirol, Davide Cucci

**Examples**

```
MAT()
MAT(sigma2 = 1, lambda = 0.35, alpha = 0.9)
```

---

np_boot_sd_med	<i>Bootstrap standard error for the median</i>
----------------	--

---

**Description**

Non-parametric bootstrap to obtain the standard of the median of iid data.

**Usage**

```
np_boot_sd_med(x, B = 5000)
```

**Arguments**

x	A vector of data.
B	A numeric indicating the number of simulations.

**Value**

Bootstrap standard error for the median

---

plot.gmwm	<i>Plot the GMWM with the Wavelet Variance</i>
-----------	--

---

**Description**

Displays a plot of the Wavelet Variance (WV) with the CI values and the WV implied by the estimated parameters.

**Usage**

```
## S3 method for class 'gmwm'
plot(
  x,
  decomp = FALSE,
  units = NULL,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  col_wv = NULL,
  col_ci = NULL,
  nb_ticks_x = NULL,
  nb_ticks_y = NULL,
  legend_position = NULL,
  ci_wv = NULL,
  point_cex = NULL,
  point_pch = NULL,
  ...
)
```

**Arguments**

x	A gmwm object.
decomp	A boolean that determines whether the contributions of each individual model are plotted.
units	A string that specifies the units of time plotted on the x axis.
xlab	A string that gives a title for the x axis.
ylab	A string that gives a title for the y axis.
main	A string that gives an overall title for the plot.
col_wv	A string that specifies the color of the wavelet variance line.
col_ci	A string that specifies the color of the shaded area covered by the confidence intervals.
nb_ticks_x	An integer that specifies the maximum number of ticks for the x-axis.
nb_ticks_y	An integer that specifies the maximum number of ticks for the y-axis.
legend_position	A string that specifies the position of the legend (use legend_position = NA to remove legend).
ci_wv	A boolean that determines whether to plot the confidence interval shaded area.
point_cex	A double that specifies the size of each symbol to be plotted.
point_pch	A double that specifies the symbol type to be plotted.
...	Additional arguments affecting the plot.

**Value**

Plot of WV and relative confidence intervals for each scale.

**Author(s)**

Stephane Guerrier and Yuming Zhang

---

plot.PACF

*Plot Partial Auto-Covariance and Correlation Functions*

---

**Description**

The function plots the output of the [theo\\_pacf](#) and [auto\\_corr](#) functions (partial autocovariance or autocorrelation functions).

**Usage**

```
## S3 method for class 'PACF'
plot(
  x,
  xlab = NULL,
  ylab = NULL,
  show.ci = TRUE,
  alpha = NULL,
  col_ci = NULL,
  transparency = NULL,
  main = NULL,
  parValue = NULL,
  ...
)
```

**Arguments**

x	A "PACF" object output from <a href="#">theo_pacf</a> or <a href="#">auto_corr</a> .
xlab	A string indicating the label of the x axis: the default name is 'Lags'.
ylab	A string indicating the label of the y axis: the default name is 'PACF'.
show.ci	A bool indicating whether to show the confidence region. Defaults to TRUE.
alpha	A double indicating the level of significance for the confidence interval. By default $\alpha = 0.05$ which gives a $1 - \alpha = 0.95$ confidence interval.
col_ci	A string that specifies the color of the region covered by the confidence intervals (confidence region).
transparency	A double between 0 and 1 indicating the transparency level of the color defined in col_ci. Defaults to 0.25.
main	A string indicating the title of the plot. Default name is "Variable name PACF plot".
parValue	A vector defining the margins for the plot.
...	Additional parameters

**Author(s)**

Yunxiang Zhang and Yuming Zhang

**Examples**

```
# Plot the Partial Autocorrelation
m = auto_corr(datasets::AirPassengers, pacf = TRUE)
plot(m)

# More customized CI
plot(m, xlab = "my xlab", ylab = "my ylab", show.ci = TRUE,
alpha = NULL, col_ci = "grey", transparency = 0.5, main = "my main")
```

---

plot.simtsACF

*Plot Auto-Covariance and Correlation Functions*


---

### Description

The function plots the output of the `theo_acf` and `auto_corr` functions (autocovariance or auto-correlation functions).

### Usage

```
## S3 method for class 'simtsACF'
plot(
  x,
  xlab = NULL,
  ylab = NULL,
  show.ci = TRUE,
  alpha = NULL,
  col_ci = NULL,
  transparency = NULL,
  main = NULL,
  parValue = NULL,
  ...
)
```

### Arguments

<code>x</code>	An "ACF" object output from <code>theo_acf</code> and <code>auto_corr</code> .
<code>xlab</code>	A string indicating the label of the x axis: the default name is 'Lags'.
<code>ylab</code>	A string indicating the label of the y axis: the default name is 'ACF'.
<code>show.ci</code>	A bool indicating whether to show the confidence region. Defaults to TRUE.
<code>alpha</code>	A double indicating the level of significance for the confidence interval. By default $\alpha = 0.05$ which gives a $1 - \alpha = 0.95$ confidence interval.
<code>col_ci</code>	A string that specifies the color of the region covered by the confidence intervals (confidence region).
<code>transparency</code>	A double between 0 and 1 indicating the transparency level of the color defined in <code>col_ci</code> . Defaults to 0.25.
<code>main</code>	A string indicating the title of the plot. Default name is "Variable name ACF plot".
<code>parValue</code>	A vector defining the margins for the plot.
<code>...</code>	Additional parameters

### Author(s)

Yunxiang Zhang, Stéphane Guerrier and Yuming Zhang

**Examples**

```
# Calculate the Autocorrelation
m = auto_corr(datasets::AirPassengers)

# Plot with 95% CI
plot(m)

# Plot with 90% CI
plot(m, alpha = 0.1)

# Plot without 95% CI
plot(m, show.ci = FALSE)

# More customized CI
plot(m, xlab = "my xlab", ylab = "my ylab", show.ci = TRUE,
alpha = NULL, col_ci = "grey", transparency = 0.5, main = "my main")
```

---

`plot_pred`*Plot Time Series Forecast Function*

---

**Description**

This function plots the time series output from a forecast method with approximate 68

**Usage**

```
plot_pred(
  x,
  model,
  n.ahead,
  level = NULL,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  ...
)
```

**Arguments**

<code>x</code>	A gts object
<code>model</code>	A ts model
<code>n.ahead</code>	An integer indicating number of units of time ahead for which to make forecasts
<code>level</code>	A double or vector indicating confidence level of prediction interval. By default, it uses the levels of 0.50 and 0.95.
<code>xlab</code>	A string for the title of x axis

<code>ylab</code>	A string for the title of y axis
<code>main</code>	A string for the over all title of the plot
<code>...</code>	Additional parameters

**Author(s)**

Yuming Zhang

---

 PLP

*Definition of a Power Law Process*


---

**Description**

Definition of a Power Law Process

**Usage**

```
PLP(sigma2 = 1, d = 0.4)
```

**Arguments**

<code>sigma2</code>	A double.
<code>d</code>	A double.

**Value**

An S3 object containing the specified ts.model with the following structure:

**process.desc** Used in summary: "SIGMA2","d"

**theta** Parameter vector including  $\sigma^2$ ,  $d$

**plength** Number of parameters

**print** String containing simplified model

**desc** "PLP"

**obj.desc** Depth of Parameters e.g. list(1,1)

**starting** Find starting values? TRUE or FALSE (e.g. specified value)

**Author(s)**

Lionel Voirol, Davide Cucci

**Examples**

```
PLP()
PLP(sigma2 = 1, d = 0.4)
```

---

predict.fitsimts      *Time Series Prediction*

---

### Description

This function plots the time series forecast.

### Usage

```
## S3 method for class 'fitsimts'
predict(
  object,
  n.ahead = 10,
  show_last = 100,
  level = NULL,
  xlab = NULL,
  ylab = NULL,
  main = NULL,
  plot = TRUE,
  ...
)
```

### Arguments

object	A fitsimts object obtained from estimate function.
n.ahead	An integer indicating number of units of time ahead for which to make forecasts.
show_last	A integer indicating the number of last observations to show in the forecast plot.
level	A double or vector indicating confidence level of prediction interval. By default, it uses the levels of 0.50 and 0.95.
xlab	A string for the title of x axis.
ylab	A string for the title of y axis.
main	A string for the over all title of the plot.
plot	A logical value. logical. If TRUE(the default) the predictions are plotted.
...	Additional arguments.

### Author(s)

Stéphane Guerrier and Yuming Zhang

**Examples**

```
Xt = gen_gts(300, AR(phi = c(0, 0, 0.8), sigma2 = 1))
model = estimate(AR(3), Xt)
predict(model)
predict(model, level = 0.95)

x = gts(as.vector(lynx), start = 1821, end = 1934, freq = 1,
unit_ts = bquote(paste(10^8, " ", m^3)), name_ts = "Numbers",
unit_time = "year", data_name = "Annual Numbers of Lynx Trappings")
model = estimate(AR(1), x)
predict(model, n.ahead = 20)
predict(model, n.ahead = 20, level = 0.95)
predict(model, n.ahead = 20, level = c(0.50, 0.80, 0.95))
```

---

predict.gmwm	<i>Predict future points in the time series using the solution of the Generalized Method of Wavelet Moments</i>
--------------	---

---

**Description**

Creates a prediction using the estimated values of GMWM through the ARIMA function within R.

**Usage**

```
## S3 method for class 'gmwm'
predict(object, data.in.gmwm, n.ahead = 1, ...)
```

**Arguments**

object	A gmwm object
data.in.gmwm	The data SAME EXACT DATA used in the GMWM estimation
n.ahead	Number of observations to forecast
...	Additional parameters passed to ARIMA Predict

**Value**

A predict.gmwm object with:

- predPredictions
- seStandard Errors
- residResiduals from ARIMA ML Fit

---

**QN***Create an Quantisation Noise (QN) Process*

---

**Description**

Sets up the necessary backend for the QN process.

**Usage**

```
QN(q2 = NULL)
```

**Arguments**

**q2** A double value for the  $Q^2$  of a QN process.

**Value**

An S3 object with called `ts.model` with the following structure:

**process.desc** Used in summary: "QN"

**theta**  $Q^2$

**plength** Number of parameters

**print** String containing simplified model

**desc** y desc replicated x times

**obj.desc** Depth of parameters e.g. `list(1)`

**starting** Guess starting values? TRUE or FALSE (e.g. specified value)

**Author(s)**

James Balamuta

**Examples**

```
QN()  
QN(q2=3.4)
```

---

 qn\_to\_wv

*Quantisation Noise (QN) to WV*


---

**Description**

This function compute the Haar WV of a Quantisation Noise (QN) process

**Usage**

```
qn_to_wv(q2, tau)
```

**Arguments**

q2	A double corresponding to variance of drift
tau	A vec containing the scales e.g. $2^{\tau}$

**Value**

A vec containing the wavelet variance of the QN.

**Process Haar Wavelet Variance Formula**

The Quantization Noise (QN) process has a Haar Wavelet Variance given by:

$$\nu_j^2(Q^2) = \frac{6Q^2}{\tau_j^2}$$

---

 read.imu

*Read an IMU Binary File into R*


---

**Description**

Process binary files within the

**Usage**

```
read.imu(file, type, unit = NULL, name = NULL)
```

**Arguments**

file	A string containing file names or paths.
type	A string that contains a supported IMU type given below.
unit	A string that contains the unit expression of the frequency. Default value is NULL.
name	A string that provides an identifier to the data. Default value is NULL.

## Details

Currently supports the following IMUs:

- IMAR
- LN200
- LN200IG
- IXSEA
- NAVCHIP\_INT
- NAVCHIP\_FLT

## Value

An imu object that contains 3 gyroscopes and 3 accelerometers in that order.

## Author(s)

James Balamuta We hope to soon be able to support delimited files.

## References

Thanks goes to Philipp Clausen of Labo TOPO, EPFL, Switzerland, topo.epfl.ch, Tel:+41(0)21 693 27 55 for providing a matlab function that reads in IMUs. This function is a heavily modified port of MATLAB code into Armadillo/C++.

## Examples

```
## Not run:
# Relative
setwd("F:/")

a = read.imu(file = "Documents/James/short_test_data.imu", type = "IXSEA")

# Fixed path
b = read.imu(file = "F:/Desktop/short_test_data.imu", type = "IXSEA")

## End(Not run)
```

---

resid\_plot

*Plot the Distribution of (Standardized) Residuals*

---

## Description

This function plots a histogram (with kernel density function and normal distribution) of the standardized residuals or a basic plot the (standardized) residuals, or both.

## Usage

```
resid_plot(res, std = FALSE, type = "hist", ...)
```

**Arguments**

<code>res</code>	A vector of residuals.
<code>std</code>	A boolean indicating whether the residuals plot is for standardized residuals or original residuals.
<code>type</code>	A string indicating either: "hist" (standardized residual histogram with superimposed kernel density estimator and normal distribution), "resid" (standard residual plot), or "both"
<code>...</code>	Additional parameters

**Author(s)**

Yuming Zhang

---

 rgmwm

---

*GMWM for Robust/Classical Comparison*


---

**Description**

Creates a `rgmwm` object to compare the results generated by robust/classical method.

**Usage**

```
rgmwm(model, data, eff = c(0.9, 0.8, 0.6), ...)
```

**Arguments**

<code>model</code>	A <code>ts.model</code> object containing one of the allowed models.
<code>data</code>	A <code>matrix</code> or <code>data.frame</code> object with only one column (e.g. $N \times 1$ ), or a <code>lts</code> object, or a <code>gts</code> object.
<code>eff</code>	A double vector between 0 and 1 that indicates the efficiency.
<code>...</code>	Other arguments passed to the main <code>gmwm</code> function.

**Details**

By default, the `rgmwm` function will fit a classical `gmwm` object. From there, the user has the ability to specify any `eff` that is less than or equal to 0.99.

**Value**

A `rgmwm` object

---

rtruncated_normal	<i>Truncated Normal Distribution Sampling Algorithm</i>
-------------------	---

---

**Description**

Enables sampling from a truncated normal

**Usage**

```
rtruncated_normal(n, mu, sigma, a, b)
```

**Arguments**

n	An unsigned int indicating the number of observations to generate.
mu	A double indicating the mean of the normal.
sigma	A double indicating the standard deviation of the normal.
a	A double that is the lower bound of the truncated normal.
b	A double that is the upper bound of the truncated normal.

---

RW	<i>Create an Random Walk (RW) Process</i>
----	---

---

**Description**

Sets up the necessary backend for the RW process.

**Usage**

```
RW(gamma2 = NULL)
```

**Arguments**

gamma2	A double value for the variance $\gamma^2$
--------	--

**Value**

An S3 object with called ts.model with the following structure:

**process.desc** Used in summary: "RW"

**theta**  $\sigma$

**plength** Number of parameters

**print** String containing simplified model

**desc** y desc replicated x times

**obj.desc** Depth of parameters e.g. list(1)

**starting** Guess starting values? TRUE or FALSE (e.g. specified value)

**Note**

We consider the following model:

$$Y_t = \sum_{t=0}^T \gamma_0 * Z_t$$

where  $Z_t$  is iid and follows a standard normal distribution.

**Author(s)**

James Balamuta

**Examples**

```
RW()  
RW(gamma2=3.4)
```

---

RW2dimension

*Function to Compute Direction Random Walk Moves*

---

**Description**

The RW2dimension function computes direction random walk moves.

**Usage**

```
RW2dimension(steps = 100, probs = c(0.25, 0.5, 0.75))
```

**Arguments**

steps	An integer that counts the number of steps of the random walk.
probs	A vector of double that specifies the probabilities to choose each direction.

**Author(s)**

Stéphane Guerrier

**Examples**

```
RW2dimension(steps = 50, probs = c(0.2, 0.5, 0.6))
```

---

 rw\_to\_wv

*Random Walk to WV*


---

**Description**

This function compute the WV (haar) of a Random Walk process

**Usage**

```
rw_to_wv(gamma2, tau)
```

**Arguments**

gamma2	A double corresponding to variance of RW
tau	A vec containing the scales e.g. $2^\tau$

**Value**

A vec containing the wavelet variance of the random walk.

**Process Haar Wavelet Variance Formula**

The Random Walk (RW) process has a Haar Wavelet Variance given by:

$$\nu_j^2(\gamma^2) = \frac{(\tau_j^2 + 2)\gamma^2}{12\tau_j}$$

---

 SARIMA

*Create a Seasonal Autoregressive Integrated Moving Average (SARIMA) Process*


---

**Description**

Sets up the necessary backend for the SARIMA process.

**Usage**

```
SARIMA(ar = 1, i = 0, ma = 1, sar = 1, si = 0, sma = 1, s = 12, sigma2 = 1)
```

**Arguments**

ar	A vector or integer containing either the coefficients for $\phi$ 's or the process number $p$ for the Autoregressive (AR) term.
i	An integer containing the number of differences to be done.
ma	A vector or integer containing either the coefficients for $\theta$ 's or the process number $q$ for the Moving Average (MA) term.
sar	A vector or integer containing either the coefficients for $\Phi$ 's or the process number $P$ for the Seasonal Autoregressive (SAR) term.
si	An integer containing the number of seasonal differences to be done.
sma	A vector or integer containing either the coefficients for $\Theta$ 's or the process number $Q$ for the Seasonal Moving Average (SMA) term.
s	An integer containing the seasonality.
sigma2	A double value for the standard deviation, $\sigma$ , of the SARMA process.

**Details**

A variance is required since the model generation statements utilize randomization functions expecting a variance instead of a standard deviation unlike R.

**Value**

An S3 object with called `ts.model` with the following structure:

**process.desc**  $AR * p, MA * q, SAR * P, SMA * Q$

**theta**  $\sigma$

**plength** Number of parameters

**desc** Type of model

**desc.simple** Type of model (after simplification)

**print** String containing simplified model

**obj.desc**  $y$  desc replicated  $x$  times

**obj** Depth of Parameters e.g. `list(c(length(ar), length(ma), length(sar), length(sma), 1, i, si))`

**starting** Guess Starting values? TRUE or FALSE (e.g. specified value)

**Author(s)**

James Balamuta

**Examples**

```
# Create an SARIMA(1,1,2)x(1,0,1) process
SARIMA(ar = 1, i = 1, ma = 2, sar = 1, si = 0, sma = 1)

# Creates an SARMA(1,0,1)x(1,1,1) process with predefined coefficients.
SARIMA(ar=0.23, i = 0, ma=0.4, sar = .3, sma = .3)
```

SARMA

*Create a Seasonal Autoregressive Moving Average (SARMA) Process***Description**

Sets up the necessary backend for the SARMA process.

**Usage**

```
SARMA(ar = 1, ma = 1, sar = 1, sma = 1, s = 12, sigma2 = 1)
```

**Arguments**

ar	A vector or integer containing either the coefficients for $\phi$ 's or the process number $p$ for the Autoregressive (AR) term.
ma	A vector or integer containing either the coefficients for $\theta$ 's or the process number $q$ for the Moving Average (MA) term.
sar	A vector or integer containing either the coefficients for $\Phi$ 's or the process number $P$ for the Seasonal Autoregressive (SAR) term.
sma	A vector or integer containing either the coefficients for $\Theta$ 's or the process number $Q$ for the Seasonal Moving Average (SMA) term.
s	A integer indicating the seasonal value of the data.
sigma2	A double value for the standard deviation, $\sigma$ , of the SARMA process.

**Details**

A variance is required since the model generation statements utilize randomization functions expecting a variance instead of a standard deviation unlike R.

**Value**

An S3 object with called `ts.model` with the following structure:

**process.desc**  $AR * p, MA * q, SAR * P, SMA * Q$

**theta**  $\sigma$

**plength** Number of Parameters

**print** String containing simplified model

**obj.desc**  $y$  desc replicated  $x$  times

**obj** Depth of Parameters e.g. `list(c(length(ar), length(ma), length(sar), length(sma), 1) )`

**starting** Guess Starting values? TRUE or FALSE (e.g. specified value)

**Author(s)**

James Balamuta

**Examples**

```
# Create an SARMA(1,2)x(1,1) process
SARMA(ar = 1, ma = 2, sar = 1, sma = 1)

# Creates an SARMA(1,1)x(1,1) process with predefined coefficients.
SARMA(ar=0.23, ma=0.4, sar = .3, sma = .3)
```

---

savingrt	<i>Personal Saving Rate</i>
----------	-----------------------------

---

**Description**

Personal saving as a percentage of disposable personal income (DPI), frequently referred to as "the personal saving rate," is calculated as the ratio of personal saving to DPI.

**Usage**

```
savingrt
```

**Format**

A gts time series object with frequency 12 starting at 1959 and going to 2016 for a total of 691 observations.

**Source**

<https://fred.stlouisfed.org/series/PSAVERT>

---

select	<i>Time Series Model Selection</i>
--------	------------------------------------

---

**Description**

This function performs model fitting and calculates the model selection criteria to be plotted.

**Usage**

```
select(model, Xt, include.mean = TRUE, criterion = "aic", plot = TRUE)
```

**Arguments**

model	A time series model (only ARIMA are currently supported).
Xt	A vector of time series data.
include.mean	A boolean indicating whether to fit ARIMA with the mean or not.
criterion	A string indicating which model selection criterion should be used (possible values: "aic" (default), "bic", "hq").
plot	A boolean indicating whether a model selection plot is returned or not.

**Author(s)**

Stéphane Guerrier and Yuming Zhang

**Examples**

```
set.seed(763)
Xt = gen_gts(100, AR(phi = c(0.2, -0.5, 0.4), sigma2 = 1))
select(AR(5), Xt, include.mean = FALSE)

Xt = gen_gts(100, MA(theta = c(0.2, -0.5, 0.4), sigma2 = 1))
select(MA(5), Xt, include.mean = FALSE)

Xt = gen_gts(500, ARMA(ar = 0.5, ma = c(0.5, -0.5, 0.4), sigma2 = 1))
select(ARMA(5,3), Xt, criterion = "hq", include.mean = FALSE)
```

---

select\_arima

*Run Model Selection Criteria on ARIMA Models*

---

**Description**

This function performs model fitting and calculates the model selection criteria to be plotted or used in best\_model function.

**Usage**

```
select_arima(
  xt,
  p.min = 0L,
  p.max = 3L,
  d = 0L,
  q.min = 0L,
  q.max = 3L,
  include.mean = TRUE,
  plot = TRUE
)

select_arma(
  xt,
  p.min = 0L,
  p.max = 3L,
  q.min = 0L,
  q.max = 3L,
  include.mean = TRUE,
  plot = TRUE
)

select_ar(xt, p.min = 0L, p.max = 3L, include.mean = TRUE, plot = TRUE)
```

```
select_ma(xt, q.min = 0L, q.max = 3L, include.mean = TRUE, plot = TRUE)
```

### Arguments

<code>xt</code>	A vector of univariate time series.
<code>p.min</code>	An integer indicating the lowest order of AR(p) process to search.
<code>p.max</code>	An integer indicating the highest order of AR(p) process to search.
<code>d</code>	An integer indicating the differencing order for the data.
<code>q.min</code>	An integer indicating the lowest order of MA(q) process to search.
<code>q.max</code>	An integer indicating the highest order of MA(q) process to search.
<code>include.mean</code>	A bool indicating whether to fit ARIMA with the mean or not.
<code>plot</code>	A logical. If TRUE (the default) a plot should be produced.

### Examples

```
xt = gen_arima(N=100, ar=0.3, d=1, ma=0.3)
x = select_arima(xt, d=1L)
```

```
xt = gen_ma1(100, 0.3, 1)
x = select_ma(xt, q.min=2L, q.max=5L)
best_model(x)
```

```
xt = gen_arma(10, c(.4,.5), c(.1), 1, 0)
x = select_arma(xt, p.min = 1L, p.max = 4L,
               q.min = 1L, q.max = 3L)
```

---

simple\_diag\_plot

*Basic Diagnostic Plot of Residuals*

---

### Description

This function will plot four diagnostic plots to assess how well the model fits the data. These plots are: (1) residuals plot, (2) histogram of (standardized) residuals, (3) normal Q-Q plot of residuals and (4) residuals vs fitted values plot.

### Usage

```
simple_diag_plot(Xt, model, std = FALSE)
```

### Arguments

<code>Xt</code>	The original time series data.
<code>model</code>	The arima model fit to the data.
<code>std</code>	A boolean indicating whether we use standardized residuals for the (1) residuals plot and the (2) histogram of (standardized) residuals.

**Author(s)**

Yuming Zhang

---

simplified\_print\_SARIMA

*Simplify and print SARIMA model*

---

**Description**

Simplify and print SARIMA model

**Usage**

simplified\_print\_SARIMA(p, i, q, P, si, Q, s)

**Arguments**

p	An integer denoting the length of ar.
i	An integer containing the number of differences to be done.
q	An integer denoting the length of ma.
P	An integer denoting the length of sma.
si	An integer containing the number of seasonal differences to be done.
Q	An integer denoting the length of sar.
s	An integer indicating the seasonal value of the data.

**Value**

An S3 object with the following structure:

**print** String containing simplified model

**simplified** Type of model (after simplification)

**Author(s)**

Stephane Guerrier

SIN

*Definition of a Sinusoidal (SIN) Process***Description**

Definition of a Sinusoidal (SIN) Process

**Usage**

```
SIN(alpha2 = 9e-04, beta = 0.06, U = NULL)
```

**Arguments**

alpha2	A double value for the squared amplitude parameter $\alpha^2$ (see Note for details).
beta	A double value for the angular frequency parameter $\beta$ (see Note for details).
U	A double value for the phase parameter $U$ (see Note for details).

**Value**

An S3 object containing the specified ts.model with the following structure:

**process.desc** Used in summary: "ALPHA2","BETA"

**theta** Parameter vector including  $\alpha^2, \beta$

**plength** Number of parameters

**print** String containing simplified model

**desc** "SIN"

**obj.desc** Depth of Parameters e.g. list(1,1)

**starting** Find starting values? TRUE or FALSE (e.g. specified value)

**Note**

We consider the following sinusoidal process :

$$X_t = \alpha \sin(\beta t + U)$$

, where  $U \sim \mathcal{U}(0, 2\pi)$  and  $\beta \in (0, \frac{\pi}{2})$

**Author(s)**

Lionel Voirol

**Examples**

```
SIN()
SIN(alpha2 = .5, beta = .05)
```

---

summary.fitsimts	<i>Summary of fitsimts object</i>
------------------	-----------------------------------

---

**Description**

Displays summary information about fitsimts object

**Usage**

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

**Arguments**

object	A fitsimts object
...	Other arguments passed to specific methods

**Value**

Estimated parameters values with confidence intervals and standard errors.

**Author(s)**

Stéphane Guerrier

---

summary.gmwm	<i>Summary of GMWM object</i>
--------------	-------------------------------

---

**Description**

Displays summary information about GMWM object

**Usage**

```
## S3 method for class 'gmwm'  
summary(  
  object,  
  inference = NULL,  
  bs.gof = NULL,  
  bs.gof.p.ci = NULL,  
  bs.theta.est = NULL,  
  bs.ci = NULL,  
  B = 100,  
  ...  
)
```

**Arguments**

object	A GMWM object
inference	A value containing either: NULL (auto), TRUE, or FALSE
bs.gof	A value containing either: NULL (auto), TRUE, FALSE
bs.gof.p.ci	A value containing either: NULL (auto), TRUE, FALSE
bs.theta.est	A value containing either: NULL (auto), TRUE, FALSE
bs.ci	A value containing either: NULL (auto), TRUE, FALSE
B	An int that indicates how many bootstraps should be performed.
...	Other arguments passed to specific methods

**Value**

A `summary.gmwm` object with:

- estimateEstimated Theta Values
- testinfoGoodness of Fit Information
- inferenceInference performed? T/F
- bs.gofBootstrap GOF? T/F
- bs.gof.p.ciBootstrap GOF P-Value CI? T/F
- bs.theta.estBootstrap Theta Estimates? T/F
- bs.ciBootstrap CI? T/F
- startingIndicates if program supplied initial starting values
- seedSeed used during guessing / bootstrapping
- obj.funValue of obj.fun at minimized theta
- NLength of Time Series

**Author(s)**

JJB

---

theo\_acf

*Theoretical Autocorrelation (ACF) of an ARMA process*

---

**Description**

This function computes the theoretical Autocorrelation (ACF) of an ARMA process.

**Usage**

```
theo_acf(ar, ma = NULL, lagmax = 20)
```

**Arguments**

ar	A vector containing the AR coefficients.
ma	A vector containing the MA coefficients.
lagmax	An integer indicating the maximum lag up to which to compute the theoretical ACF.

**Author(s)**

Yuming Zhang

**Examples**

```
# Compute the theoretical ACF for an ARMA(1,0) (i.e. a first-order autoregressive model: AR(1))
theo_acf(ar = -0.25, ma = NULL)
# Computes the theoretical ACF for an ARMA(2, 1)
theo_acf(ar = c(.50, -0.25), ma = 0.20, lagmax = 10)
```

---

theo\_pacf

*Theoretical Partial Autocorrelation (PACF) of an ARMA process*

---

**Description**

This function computes the theoretical Partial Autocorrelation (PACF) of an ARMA process.

**Usage**

```
theo_pacf(ar, ma = NULL, lagmax = 20)
```

**Arguments**

ar	A vector containing the AR coefficients.
ma	A vector containing the MA coefficients.
lagmax	An integer indicating the maximum lag up to which to compute the theoretical PACF.

**Author(s)**

Yuming Zhang

**Examples**

```
# Computes the theoretical ACF for an ARMA(1,0) (i.e. a first-order autoregressive model: AR(1))
theo_pacf(ar = -0.25, ma = NULL, lagmax = 7)
# Computes the theoretical ACF for an ARMA(2, 1)
theo_pacf(ar = c(.50, -0.25), ma = .20, lagmax = 10)
```

---

 update.gmwm

*Update (Robust) GMWM object for IMU or SSM*


---

### Description

Provides a way to estimate different models over the previously estimated wavelet variance values and covariance matrix.

### Usage

```
## S3 method for class 'gmwm'
update(object, model, ...)
```

### Arguments

object	A gmwm object.
model	A ts.model object containing one of the allowed models
...	Additional parameters (not used)

### Value

A gmwm object with the structure:

- estimateEstimated Parameters Values from the GMWM Procedure
- init.guessInitial Starting Values given to the Optimization Algorithm
- ww.empirThe data's empirical wavelet variance
- ci\_lowLower Confidence Interval
- ci\_highUpper Confidence Interval
- orgVOriginal V matrix
- VUpdated V matrix (if bootstrapped)
- omegaThe V matrix inverted
- obj.funValue of the objective function at Estimated Parameter Values
- theoSummed Theoretical Wavelet Variance
- decomp.theoDecomposed Theoretical Wavelet Variance by Process
- scalesScales of the GMWM Object
- robustIndicates if parameter estimation was done under robust or classical
- effLevel of efficiency of robust estimation
- model.typeModels being guessed
- compute.vType of V matrix computation
- augmentedIndicates moments have been augmented
- alphaAlpha level used to generate confidence intervals

- expect.diffMean of the First Difference of the Signal
- NLength of the Signal
- GNumber of Guesses Performed
- HNumber of Bootstrap replications
- KNumber of V matrix bootstraps
- modelts.model supplied to gmwm
- model.hatA new value of ts.model object supplied to gmwm
- startingIndicates whether the procedure used the initial guessing approach
- seedRandomization seed used to generate the guessing values
- freqFrequency of data

---

 update.lts

*Update Object Attribute*


---

### Description

Update the attributes of lts, gts and imu object

### Usage

```
## S3 method for class 'lts'
update(object, type, new, keep.start = T, ...)
```

```
## S3 method for class 'gts'
update(object, type, new, keep.start = T, ...)
```

```
## S3 method for class 'imu'
update(object, type, new, ...)
```

### Arguments

object	A lts, gts or imu object
type	A string that contains the attribute to be updated
new	The updated value for the attribute
keep.start	A boolean value that indicates whether 'start' or 'end' should remain the same when 'freq' is updated
...	Further arguments passed to or from other methods.

## Details

This function is able to update some attributes for `gts`, `lts` and `imu` objects. For `lts` object, the attributes that can be updated are `'start'`, `'end'`, `'freq'`, `'unit_time'`, `'name_ts'` and `'process'`. For `gts` object, the attributes that can be updated are `'start'`, `'end'`, `'freq'`, `'unit_time'` and `'name_ts'`. For `imu` object, the attributes that can be updated are `'axis'`, `'freq'`, `'unit_time'` and `'name_ts'`.

If one between `'start'` and `'end'` is updated, the other one will also be updated, since `end-start == (N-1)/freq` must be `TRUE`, where `N` is the number of observations in the object.

If `'freq'` is updated, by default `'start'` will remain the same, and `'end'` will be updated at the same time, unless you set `'keep.start = F'`.

If `'unit_time'` is updated, the old `unit_time` will be replaced by the new one, and other attributes will remain the same. It is different from the `unit_time` conversion feature.

## Value

An object with the updated attribute.

## Examples

```
gts1 = gts(rnorm(50), freq = 1, unit_time = 'sec', name_ts = 'test1')
gts2 = update(gts1, 'unit_time', 'min')
attr(gts2, 'unit_time')

gts3 = update(gts1, 'name_ts', 'test2')
attr(gts3, 'name_ts')
```

---

value

*Obtain the value of an object's properties*

---

## Description

Used to access different properties of the `gts`, `imu`, or `lts` object.

## Usage

```
value(x, type)

## S3 method for class 'imu'
value(x, type)
```

## Arguments

`x` A `gts`, `imu`, or `lts` object.  
`type` A string indicating the field to be retrieved.

**Details**

To access information about imu properties use:

"accel" Returns the number of accelerometers

"gyro" Returns the number of gyroscopes

"sensors" Returns total number of sensors

**Value**

The method will return a single numeric or string result depending on the slot being accessed.

**Methods (by class)**

- value(imu): Access imu object properties

**Author(s)**

James Balamuta

---

 WN

---

*Create an White Noise (WN) Process*


---

**Description**

Sets up the necessary backend for the WN process.

**Usage**

```
WN(sigma2 = NULL)
```

**Arguments**

sigma2            A double value for the variance,  $\sigma^2$ , of a WN process.

**Value**

An S3 object with called ts.model with the following structure:

**process.desc** Used in summary: "WN"

**theta**  $\sigma$

**plength** Number of Parameters

**print** String containing simplified model

**desc** y desc replicated x times

**obj.desc** Depth of Parameters e.g. list(1)

**starting** Guess Starting values? TRUE or FALSE (e.g. specified value)

**Note**

In this process,  $Y_t$  is iid from a zero mean normal distribution with variance  $\sigma^2$

**Author(s)**

James Balamuta

**Examples**

```
WN()
WN(sigma2=3.4)
```

---

 wn\_to\_wv

*Gaussian White Noise to WV*


---

**Description**

This function compute the Haar WV of a Gaussian White Noise process

**Usage**

```
wn_to_wv(sigma2, tau)
```

**Arguments**

sigma2	A double corresponding to variance of WN
tau	A vec containing the scales e.g. $2^\tau$

**Value**

A vec containing the wavelet variance of the white noise.

**Process Haar Wavelet Variance Formula**

The Gaussian White Noise (WN) process has a Haar Wavelet Variance given by:

$$\nu_j^2(\sigma^2) = \frac{\sigma^2}{\tau_j^2}$$

[.imu

*Subset an IMU Object***Description**

Enables the IMU object to be subsettable. That is, you can load all the data in and then select certain properties.

**Usage**

```
## S3 method for class 'imu'
x[i, j, drop = FALSE]
```

**Arguments**

x	A imu object
i	A integer vector that specifies the rows to subset. If blank, all rows are selected.
j	A integer vector that specifies the columns to subset. Special rules apply see details.
drop	A boolean indicating whether the structure should be preserved or simplified.

**Details**

When using the subset operator, note that all the Gyroscopes are placed at the front of object and, then, the Accelerometers are placed.

**Value**

An imu object class.

**Examples**

```
## Not run:
if(!require("imudata")){
  install_imudata()
  library("imudata")
}

data(imu6)

# Create an IMU Object that is full.
ex = imu(imu6, gyros = 1:3, accels = 4:6, axis = c('X', 'Y', 'Z', 'X', 'Y', 'Z'), freq = 100)

# Create an IMU object that has only gyros.
ex.gyro = ex[,1:3]
ex.gyro2 = ex[,c("Gyro. X", "Gyro. Y", "Gyro. Z")]
```

```
# Create an IMU object that has only accels.
ex.accel = ex[,4:6]
ex.accel2 = ex[,c("Accel. X", "Accel. Y", "Accel. Z")]

# Create an IMU object with both gyros and accels on axis X and Y
ex.b = ex[,c(1,2,4,5)]
ex.b2 = ex[,c("Gyro. X", "Gyro. Y", "Accel. X", "Accel. Y")]

## End(Not run)
```

# Index

## \* datasets

- australia, 13
- hydro, 49
- savingrt, 78

[.imu, 91

AIC.fitsimts, 4

AR, 5

AR1, 6

ar1\_to\_wv, 7

ARIMA, 8

ARMA, 9

ARMA11, 10

arma11\_to\_wv, 11

arma\_to\_wv, 7, 12, 12, 57

australia, 13

auto\_corr, 13, 62, 63

best\_model, 15

check, 15

compare\_acf, 16

corr\_analysis, 17

deriv\_2nd\_ar1, 19

deriv\_2nd\_arma11, 20

deriv\_2nd\_dr, 21

deriv\_2nd\_ma1, 22

deriv\_ar1, 23

deriv\_arma11, 24

deriv\_dr, 25

deriv\_ma1, 25

deriv\_qn, 26

deriv\_rw, 27

deriv\_wn, 28

derivative\_first\_matrix, 18

diag\_boxpierce, 28

diag\_ljungbox, 29

diag\_plot, 30

diag\_portmanteau\_, 30

DR, 31

dr\_to\_wv, 32

estimate, 33

evaluate, 34

FGN, 35

gen\_ar1blocks, 36

gen\_bi, 37

gen\_gts, 38

gen\_lts, 39

gen\_nswn, 41

GM, 42

gmwm, 43

gmwm\_imu, 46

gts, 42, 47

hydro, 49

imu, 49

imu\_time, 51

inherits, 52

is, 52

is.gts, 52

is.imu (is.gts), 52

is.lts (is.gts), 52

is.ts.model (is.gts), 52

lts, 53

M, 54

MA, 55

MA1, 56

ma1\_to\_wv, 57

make\_frame, 58

MAPE, 59

MAT, 60

np\_boot\_sd\_med, 61

plot.gmwm, 61

plot.PACF, 62  
plot.simtsACF, 64  
plot\_pred, 65  
PLP, 66  
predict.fitsimts, 67  
predict.gmwm, 68

QN, 69  
qn\_to\_wv, 70

read.imu, 70  
resid\_plot, 71  
rgmwm, 72  
rtruncated\_normal, 73  
RW, 73  
RW2dimension, 74  
rw\_to\_wv, 75

SARIMA, 75  
SARMA, 77  
savingrt, 78  
select, 78  
select\_ar, 15  
select\_ar (select\_arima), 79  
select\_arima, 79  
select\_arma, 15  
select\_arma (select\_arima), 79  
select\_ma, 15  
select\_ma (select\_arima), 79  
simple\_diag\_plot, 80  
simplified\_print\_SARIMA, 81  
SIN, 82  
summary.fitsimts, 83  
summary.gmwm, 83

theo\_acf, 84  
theo\_pacf, 62, 63, 85

update.gmwm, 86  
update.gts (update.lts), 87  
update.imu (update.lts), 87  
update.lts, 87

value, 88

WN, 89  
wn\_to\_wv, 90