TSAGAUSS

a GAUSS Library for Time Series Analysis

written / collected

by Rainer Schlittgen and Thomas Noack

April 2001

Institut for Statistics and Econometry
University of Hamburg

e-mail: schlitt@hermes1.econ.uni-hamburg.de

This library is offered to the public in the spirit of the GNU General Public License. To emphasize a part of the preamble is cited here:

Only programs are listed here which may be used directly but auxiliary programs aren't. See the file `tsagauss.lcg` for a complete listing.

Inhaltsverzeichnis

ACORE

- **Purpose**

  Empirical autocorrelation function of time series

- **Format**

  `ac = ACORE(y,nlag)`

- **Input**

`y`	(n, k)-matrix, k time series of length n
`nlag`	maximum lag

- **Output**

`ac`	$(nlag + 1, k)$-matrix, autocorrelation functions for each time series separately; $ac[i, j] = r_{Y_t^{(j)}}(i - 1)$

- **Remarks**
  Allowing for missing values.
  For crosscorrelation function see `CCORE`

- **Source**

  `tsmoment.src`

ACORPLOT

- **Purpose**

  Plot of autocorrelation function and partial correlation function
  (for up to 3 time series)

- **Format**

  ```
  ACORPLOT(y,mlag,k)
  ```

- **Input**

`y`	(n, k)-matrix, up to 3 time series, e.g. $k \in \{1, 2, 3\}$
`mlag`	integer, maximum lag
`k`	integer; if =0: no limits; if =1: with limits

- **Remarks**

  For $k = 1$: Limits are $2\times$stdev of ACF/PACF

- **Source**

  `tsplot.src`

ACORRB

- **Purpose**

  Robust estimation of autocorrelation function by Huberizing the time series

- **Format**

  `ac = ACORRB(y,nlag)`

- **Input**

`y`	$(n, 1)$-vector, the time series
`nlag`	integer, number of lags

- **Output**

`ac`	$(nlag + 1, 1)$-vector of autocorrelation coefficients

- **Remarks**

  ac is determined via `acovrb`; the correlations are transformed as if they were determined from correlated Gaussian variables.

- **Source**

  `tsrobust.src`

ACORTEPLOT

- **Purpose**

  Plot of theoretical (and empirical) correlation / partial correlation function of time series

- **Format**

  `ACORTEPLOT(a,b,x,mlag)`

- **Input**

`a`	$(p, 1)$-vector of ar-coefficients: $1 - \alpha_1 \mathrm{B} - \ldots - \alpha_p \mathrm{B}^p$
`b`	$(q, 1)$-vector of ma-coefficients: $1 - \beta_1 \mathrm{B} - \ldots - \beta_q \mathrm{B}^q$
`x`	$(n, 1)$-vector, time series **or**
	scalar$= 0$: only the theoretical functions are plotted
`mlag`	integer, maximum lag

- **Source**

  `tsplot.src`

ACOVE

- **Purpose**

  Empirical autocovariance function of time series

- **Format**

  `ac = ACOVE(y,nlag)`

- **Input**

`y`	(n, k)-matrix, k time series of length n
`nlag`	maximum lag

- **Output**

`ac`	$(nlag + 1, k)$-matrix, autocovariance functions for each time series separately; $ac[i, j] = c_{Y_t^{(j)}}(i - 1)$

- **Remarks**
  Allowing missing values. For crosscovariance function see `CCOVE`

- **Source**

  `tsmoment.src`

ACOVFPER

- **Purpose**

  Recovering ACF from periodogram

- **Format**

  `acf = ACOVFPER(per,nlag)`

- **Input**

`per`	$(m, 1)$-vector, periodogram at frequencies $0 \leq k/(2n-1) < 1/2$
`nlag`	maximum ACF-lag

- **Output**

`acf`	$(nlag + 1, 1)$-vector, ACF at lags $0, 1, \ldots, nlag$

- **Remarks**

  The length n of the series is computed from the number of frequencies, i.e. the number of rows of `per`

- **Source**

  `spectrum.src`

ACOVRB

- **Purpose**

  Robust estimate of autocovariance function

- **Format**

  `ac = ACOVRB(y,nlag)`

- **Input**

`y`	$(n, 1)$-vector, the time series
`nlag`	integer, number of lags

- **Output**

`ac`	$(nlag + 1, 1)$-vector of acf-coefficients

- **Remarks**

  ac is `1.285*acove(s*psihub((y-tsmeanrb(y,&psihub))/s)`, where s is the MAD.
  The constant ensures correct variance in case of a normal white noise process.

- **Source**

  `tsrobust.src`

ACOVTH

- **Purpose**

  Theoretical autocovariance-function for ARMA[p,q]-process

- **Format**

  `ac = ACOVTH(a,b,s,nlag)`

- **Input**

`a`	$(p,1)$-vector of ar-coefficients: $\alpha_1, \ldots, \alpha_p$
`b`	$(q,1)$-vector of ma-coefficients: $\beta_1, \ldots, \beta_q$
`s`	scalar, variance of error process
`nlag`	integer, number of lags

- **Output**

`ac`	$(m+1,1)$-vector of acf-coefficients $\gamma_0, \gamma_1, \ldots, \gamma_{nlag}$.

- **Reference**

  McLeod, A. (1975): Derivation of the Theoretical Autocovariance Function of Autoregressive-Moving Averrage Time Series; Appl. Statistics, 24, 255-256

- **Remarks**

  Use $a = 0$ / $b = 0$ for pure ma-/ar-process.

  To get the autocorrelation-function, simply compute `ac/ac[1]`.

  The model is assumed to be in the form
  $(1 - \alpha_1 \mathrm{B} - \ldots - \alpha_p \mathrm{B}^p) Y_t = (1 - \beta_1 \mathrm{B} - \ldots - \beta_q \mathrm{B}^q) \varepsilon_t$

- **Source**

  `arimafit.src`

ARESTRB

- **Purpose**

  Estimation of AR-coefficients

- **Format**

  `a = AREST(y,p,meth)`

- **Input**

`y`	$(n, 1)$-vector, time series
`p`	integer, order of AR-model
`meth`	string for selection of the method: "yw" : Yule Walker- method "burg" : Burg's estimator "orth" : Gram-Schmidt-orthogonalisation for partial correlations

- **Output**

`a`	$(p, 1)$-vector of estimated AR-coefficients

- **Reference**

  Newton,H.J. and Pagano,M. (1983): Computing for Autoregressions; in: Computer Science and Statistics: The Interface, J.E.Gentle (ed.), North Holland Publishing Company

- **Source**

  `arimafit.src`

ARESTRB

- **Purpose**

  Robust estimation of AR[p]-model via GM-estimation

- **Format**

  `{a,cov} = ARESTRB(y,p,co)`

- **Input**

`y`	$(n, 1)$-vector, time series
`p`	integer, order of AR-model
`co`	flag, if $co = 1$, the covariance matrix of the estimates are estimated via bootstrap

- **Output**

`a`	$(p, 1)$-vector of estimated AR-coefficients
`cov`	the estimated covarianc matrix of `a` ($co = 1$) or a missing value ($co \neq 1$)

- **Remarks**

  Start values from YW-fit based on robust acf

- **Source**

  `tsrobust.src`

ARFIT

- **Purpose**

  Fitting an AR[p]-model to a time series by Yule Walker method

- **Format**

  {a,cov,aic,bic,res} = ARFIT(y,pmax,ic,c)

- **Input**

y	$(n, 1)$-vector, time series
pmax	integer ≥ 1, maximal order of AR-model
ic	string, "aic": order selection by minimum aic "bic": order selection by minimum bic
c	string, "asymp": asymptotic covariance matrix for coefficients "boot": bootstrap covariance matrix for coefficients

- **Output**

a	$(p, 1)$-vector of estimated AR-coefficients
cov	(p, p)-matrix, covariance matrix of coefficients
aic	$(pmax, 1)$ vector, AIC-values for all orders $\leq pmax$
bic	$(pmax, 1)$ vector, BIC-values for all orders $\leq pmax$
res	$(n - p, 1)$-vector, residuals of fitted AR-model

- **Source**

  arimafit.src

ARIMAPRED

- **Purpose**

  Forecasting seasonal ARIMA[p,q]x[ps,qs]-model

- **Format**

  `pred = ARIMAPRED(y,d,par,pind,psind,qind,qsind,s,lead,a)`

- **Input**

`y`	$(n, 1)$-vector, (undifferenced) time series
`d`	$(k, 1)$-vector, the differences which were applied to reach stationarity
`par`	$(p1 + p2 + q1 + q2, 1)$-vector the estimated parameters of the ARIMA-model, output of armaest
`pind`	$(p1, 1)$-vector, lags for the non seasonal ar polynom
`psind`	$(p2, 1)$-vector, lags for the seasonal ar polynom
`qind`	$(q1, 1)$-vector, lags for the non seasonal ma polynom
`qsind`	$(q2, 1)$-vector, lags for the seasonal ma polynom
`s`	scalar, the seasonal period
`lead`	scalar, the forecasting horizon
`a`	scalar, the probability α to get $(1 - \alpha)$ prediction intervals

- **Output**

`pred`	$(lead, 3)$-matrix, `pred[.,1]`: predictions `pred[.,2]`: lower bounds of prediction intervals `pred[.,3]`: upper bounds of prediction intervals

- **Remarks**

  If an ar- or ma-polynom does not existed for estimation, the input value of the corresponding polynom must be 0.

- **Source**

  `predict.src`

ARMADHR

- **Purpose**

  Estimates subset ARMA[p,q]-model by the method of Durbin-Hannan-Rissanen.

- **Format**

  `{par,stdpar,cor,e,s} = ARMADHR(x,arsub,masub)`

- **Input**

`x`	$(n, 1)$-vector, time series
`arsub`	$(p, 1)$-vector, lags which are to be used in the autoregressive part of the model. or $= 0$ for no ar-part
`masub`	$(q, 1)$-vector, lags which are to be used in the moving average part of the model. or $= 0$ for no ma-part

- **Output**

`par`	$([p + q], 1)$-vector, the estimated parameters
`stdpar`	$([p + q], 1)$-vector, estimated standard error of estimated parameters
`cor`	$([p + q], [p + q])$-matrix, estimated correlation matrix of estimated parameters
`e`	$(n, 1)$-vector, the estimated residuals.
`s`	scalar, the estimated residual variance

- **Reference**

  Durbin (1960) Revue Inst. Int. De Statist.
  Hannan, E. J. and Rissanen, J. (1982): Recursive estimation of mixed autoregressive-moving average order; Biometrika 69, 81-94

- **Source**

  `arimafit.src`

ARMAEST

- **Purpose**

  Maximum likelihood estimation of seasonal subset ARMA[p,q]x[ps,qs]-model

- **Format**

  `{par,stdpar,corpar,sigma2,e,ll} = ARMAEST(y,p,ps,q,qs,s,par0)`

- **Input**

`y`	$(n, 1)$-vector, time series
`p`	$(p1, 1)$-vector, lags which are to be used in the non-seasonal ar part of the model
`ps`	$(p2, 1)$-vector, lags which are to be used in the seasonal ar part of the model
`q`	$(q1, 1)$-vector, lags which are to be used in the non-seasonal ma part of the model
`qs`	$(q2, 1)$-vector, lags which are to be used in the seasonal ma part of the model
`s`	scalar, the seasonal period
`par0`	$(p1+p2+q1+q2, 1)$-vector, starting values for the parameters. the order of the parameters is ascending and all parameters of all polynoms are stacked together or $= 0$ when the program should determine starting values

- **Output**

`par`	$([p1 + p2 + q1 + q2], 1)$-vector, the estimated parameters
`stdpar`	$([p1 + p2 + q1 + q2], 1)$-vector, estimated standard error of estimated parameters
`cor`	$([p1 + p2 + q1 + q2], [p1 + p2 + q1 + q2])$-matrix, estimated correlation matrix of estimated parameters
`sigma2`	scalar, the estimated residual variance
`e`	$(n, 1)$-vector, the estimated residuals.
`ll`	scalar, the value of the loglikelihood function at the estimated parameter values.

- **Reference**

  Jones, R.H. (1980): Maximum Likelihood fitting of ARMA models to time series with missing observations; Technometrics, 22, 389-395

- **Remarks**

If an ar or an ma- polynom is not desired, set the lags equal to 0.
The Kalman filter is used to compute the likeliood and Nelder-Mead's simplex algorizhm is used as optimizer. The covariance matrix is obtaind with GAUSS' procedure HESSP.

- **Source**

  `armaest.src`

ARMAMLCV

- **Purpose**

  Computes asymptotic covariance matrix for
  ML estimator of ARMA[p,q]-model.

- **Format**

  `v = ARMAMLCV(ar,ma)`

- **Input**

`ar`	$(p,1)$-vector, estimates of coefficients of ar-part 0 if no ar-part
`ma`	$(q,1)$-vector, estimates of coefficients of ma-part 0 if no ma-part

- **Output**

`v`	$p{+}q, p{+}q)$-matrix, n times the asymptotic variance-covariance matrix of estimates

- **Reference**

  Godolphin, E. J. and Unwin, J.M. (1983): Evaluation of the covariance matrix for the maximum likelihood estimator of a Gaussian autoregressive-moving average process; Biometrika 70, 279-84

- **Remarks**

  v is determined as if the data were complete

- **Source**

  `arimafit.src`

ARMARESID

- **Purpose**

  Residuals fom an ARMA[p,q]-model

- **Format**

  `e = ARMARESID(x,ar,ma)`

- **Input**

`x`	$(n, 1)$-vector, time series
`ar`	$(p, 1)$-vector the autoregressive parameters of the model
`ma`	$(q, 1)$-vector, the moving average parameters of the model

- **Output**

`e`	$(n - max(p, q), 1)$-vector, the estimated residuals.

- **Source**

  `regarma.src`

ARSUBFIT

- **Purpose**

  Fit of a subset-ar-model for time series (choose the lags and estimate the corresponding parameters)

- **Format**

  `{arset,a,cov,res} = ARSUBFIT(y,subset,set2null,pmax)`

- **Input**

`y`	$(n, 1)$-vector, time series
`subset`	$(h, 1)$-vector of lags of a given subset
	0, no given subset
`set2null`	$(j, 1)$-vector of lags to be set to null
	0, no lag is set to null
`pmax`	integer $ge1$, maximal order of ar-coefficient

- **Output**

`lagset`	$(p, 1)$-vector, indices of relevant lags
`a`	$(p, 1)$-vector, estimated ar-coefficients
`cov`	(p, p)-matrix, variance matrix of estimated coefficients
`res`	$(m, 1)$-vector, the estimated residuals.

- **Source**

  `ariamfit.src`

BANDFILT

- **Purpose**

  bandpass filtering of a time series

- **Format**

  `yf = BANDFILT(y,q,l0,lc)`

- **Input**

`y`	$(n, 1)$-vector, time series to be filtered
`q`	scalar, gives half of length of symmetric weights
`l0`	scalar, half the desired broadnes of the filter (in frequency terms)
`l1`	scalar, the frequency at which the filter is centered

- **Output**

`yf`	$(n, 1)$-vector, the filtered time series

- **Remarks**

  q values at the begining and at the end are set to missing.

- **Source**

  `linfilt.src`

BCPLOT

- **Purpose**

  Box-Cox-transformation plot for time series

- **Format**

  `BCPLOT(y)`

- **Input**

  `y` $(n, 1)$-vector, time series

- **Output**

  Plot

- **Remarks**

  The series is divided into 10 segments. For each segment i the mean and the standard deviation is determined. The plot shows the points $(ln(mean(y_i)), ln(std(y_i))$ together with a regression line; its slope b is given, too. The transformation $z = y^{1-b}$ stabilizes variances.
  Allowing for missing values.

- **Source**

  `tsplot.src`

BPLTEST

- **Purpose**

  Computing the Box-Pierce-Lijung test statistic and p-values

- **Format**

  `LBPTEST(e,n,par)`

- **Input**

`e`	$(n', 2)$-vector, the series of residuals
`n`	scalar, length of series from which e is obtained
`par`	scalar, number of parameters estimated to obtain e

- **Source**

  `arimafit.src`

BISPECES

- **Purpose**

  Indirect bivariate spectral estimation

- **Format**

  `{f,coh,ph} = BISPECES(y,q,&win)`

- **Input**

`y`	$(n, 2)$-matrix, 2 time series of length n
`q`	scalar, number of covariances used for averaging
`win`	procedure, the lagwindow

- **Output**

`f`	$((n + 1)/2, 1)$-vector of fourierfrequencies 0, 1/n, 2/n, ..
`coh`	$((n + 1)/2, 1)$-vector, the estimated coherency at frequencies f
`ph`	$((n + 1)/2, 1)$-vector, the estimated phase at frequencies f

- **Remarks**
  Possible lagwindows:
  Bartlett's Lag-window: `lagwinba`
  Tukey's Lag-window: `lagwintu`
  Parzen's Lag-window: `lagwinpa`

- **Source**

  `bivariat.src`

CCORE

- **Purpose**

  Computes crosscorrelation function of 2 time series

- **Format**

  `{lag,cc} = CCORE(x,nlag)`

- **Input**

`x`	$(n, 2)$-matrix, 2 time series of length n
`nlag`	$(2, 1)$-vector, number of lags $(nlag1, nlag2)$

- **Output**

`lag`	$(nlag2 - nlag2 + 1, 1)$-vector, of lags from $lag1$ to $lag2$
`cc`	$(nlag1 - nlag2 + 1, 1)$-vector of crosscorrelations $c_{12}(\tau)/\sqrt{c_1(0) \cdot c_2(0)}$, where $c12(\tau)$ is the crosscovariance of the two series

- **Remarks**

  Allowing missing values.

- **Source**

  `bivariat.src`

CCORPLOT

- **Purpose**

  Plot of crosscorrelation function for 2 time series

- **Format**

  `CCORPLOT(y,mlag,lim)`

- **Input**

`y`	$(n,2)$-matrix, 2 time series
`mlag`	integer, maximum (absolut) lag
`lim`	flag, if =1, confidence limits are shown

- **Remarks**
  Plotted are crosscorrelations from -mlag to +mlag.
  Limits are 2×stdev of CCF for uncorrelated series, where one is white noise.

- **Source**

  `bivariat.src`

CCOVE

- **Purpose**

  Computes crosscovariance function of 2 time series

- **Format**

  `{lag,cc} = CCOVE(x,nlag)`

- **Input**

`x`	$(n, 2)$-matrix, 2 time series of length n
`nlag`	$(2, 1)$-vector, range of lags $(nlag1, nlag2)$ $(nlag1 \leq nlag2)$

- **Output**

`lag`	$(nlag2 - nlag1 + 1, 1)$-vector, of lags from $lag1$ to $lag2$
`cc`	$(nlag1 - nlag1 + 1, 1)$-vector of crosscovariances $\sum_{1 \leq t, t+\tau \leq n} (x_{1,t} - \bar{x}_1)(x_{2,t+\tau} - \bar{x}_2)$

- **Remarks**

  Allowing for missing values.

- **Source**

  `bivariat.src`

DIFF

- **Purpose**

  Computing difference-filter of time series

- **Format**

  `yd = DIFF(y,d)`

- **Input**

`y`	$(n, 1)$-vector, time series to be differenced
`d`	$(j, 1)$-vector, containing the lags for differencing

- **Output**

`yd`	$(n - \texttt{sumc}(d), 1)$-vector, differenced series

- **Remarks**

  For repeated differencing use the same lag more often.
  Example: $d = \{1, 1\}$ for simple differencing twice

- **Source**

  `auxilary.src`

EXPOSMOOTH

- **Purpose**

  Exponential smoothing of a time series, local constant model
  $y_{t+u} = m_t + \varepsilon_u, \ -q \leq u \leq q$

- **Format**

  `{yhat,b} = EXPOSMOOTH(y,b,start,lead)`

- **Input**

`y`	$(n, 1)$-vector, time series
`b`	scalar, smoothing constant if 0¡b¡1 if $b \notin (0, 1)$ then the smoothing constant is determined to minimize the one step prediction error
`start`	integer, time index from which on the one-step-predictions are given
`lead`	integer, forecast horizont

- **Output**

`yhat`	$(n - start + lead, 2)$-matrix; the first column gives the time index, and the second the one-step-forecasts
`b`	scalar, smoothing constant

- **Remarks**

  A plot is shown also. Use `CALL exposmooth` to see the plot only.

- **Source**

  `tsdecomp.src`

EXPOSMOOTH1

- **Purpose**

  Computes exponential smoothing of time series, local linear trend model $y_{t+u} = m_t + h \cdot u + \varepsilon_u, \ -q \leq u \leq q$

- **Format**

  `{yhat} = EXPOSMOOTH1(y,ab,start,lead)`

- **Input**

`y`	$(n, 1)$-vector, time series
`ab`	$(2, 1)$-vector, smoothing constants (0¡ab¡1) $ab[., 1]$ level, $ab[., 2]$ slope
`start`	integer, the index from which on the one-step-predictions are determined
`lead`	integer, forecast horizont

- **Output**

`yhat`	$(n - start + lead, 2)$-matrix; the first column gives the time index, and the second the one-step-forecasts

- **Remarks**

  $h-$step forcasts are computed as $\hat{y}_{n,h} = \hat{m}_n + h \cdot \hat{b}_n$

- **Source**

  `tsdecomp.src`

FOURIER

- **Purpose**

  Computes the fouriercoefficients of time series

- **Format**

  `fc = FOURIER(y)`

- **Input**

  `y` $(n, 1)$-vector, time series

- **Output**

  `fc` $([n/2]+1, 2)$-matrix, the fourier coefficients at frequencies k/n, $0 \leq k/n \leq 1/2$

- **Source**

  `spectrum.src`

FOURINV

- **Purpose**

  Reconstructing time series from fourier coefficients which were determined with FOURIER

- **Format**

  `x = FOURINV(fcoe,n)`

- **Input**

`fcoe`	$([n/2]+1,2)$-matrix, fourier coefficients at frequencies k/n
`n`	integer, length of time series

- **Output**

`x`	$(n,1)$-vector, time series

- **Source**

  `spectrum.src`

HOLTWINT

- **Purpose**

  Computes and plots exponential smoothing of time series using Holt-Winters-method

- **Format**

  `yhat = HOLTWINT(y,abc,d,start,lead)`

- **Input**

`y`	$(n, 1)$-vector, time series
`abc`	$(3, 1)$-vector, weights $abc[., 1]$: level $abc[., 2]$: ascent $abc[., 3]$: seasonal component
`d`	integer, seasonal period (1 when no seasonal component)
`start`	integer, time point from which on the forecasts are computed
`lead`	integer, forecasting horizont

- **Output**

`yhat`	$(n - start + lead, 2)$-matrix; the first column gives the time index, and the second the one-step-forecasts

- **Remarks**

  A plot of the selectet part of the series together with the one step $(t \leq n + 1)$ and more-step forecasts $(n + 1 < t \leq n + h)$ is shown also

- **Source**

  `tsdecomp.src`

IACORE

- **Purpose**

  Computing the inverse autocorelation function IACF using the autocorrelation function ACF

- **Format**

  `iac = IACORE(acf,m)`

- **Input**

`acf`	$(n, 1)$-vector, acf at lags $1, \ldots, n$
`m`	integer $(m < n)$, maximum IACF-lag

- **Output**

`iac`	$(m, 1)$-vector, IACF at lags $1, \ldots, m$

- **Source**

  `tsmoment.src`

INNOVREC

- **Purpose**

  Estimation of MA coefficients via innovation algorithm by Brockwell & Davis

- **Format**

  `{th,v} = INNOVREC(x,m)`

- **Input**

`x`	$(n, 1)$-vector, the time series
`m`	integer $(m < n)$, number of MA-coefficient

- **Output**

`th`	$(m, 1)$-vector, MA coefficients
`v`	scalar, estimate of the innovation variance

- **Reference**

  Brockwell & Davis (1987): Time Series: Theory and Methods; New York: Springer

- **Source**

  `arimafit.src`

LDREC

- **Purpose**

  Levinson-Durbin recursion for determing all coefficients $a(i,j)$

- **Format**

  `mat = LDREC(acf)`

- **Input**

  `acf` $(p+1,1)$-vector,
  $acov(0),\ldots,acov(p)$ or $1, acor(1),\ldots,acor(p)$

- **Output**

  `mat` $(p,p+2)$-matrix with coefficients in lower triangular,
  pacf in the last column and $Q(p)$ in column $p+1$

- **Source**

  `arimafit.src`

LINFILT

- **Purpose**

  Filtering time series

- **Format**

  {y} = LINFILT(x,f,s)

- **Input**

x	$(n, 1)$-vector, time series
f	$(k, 1)$-vector, filter weights
s	scalar, time shift of output

- **Output**

y	filtered time series

- **Remarks**

  The output is $y_{t+s} = x_t f_1 + x_{t-1} f_2 + \cdots + x_{t+1-k} f_k$.
  Suitable numbers of missings are padded at the begin or the end.
  y can be longer than x because of the time shift into the future ($s > 0$).

- **Source**

  `linfilt.src`

MACOEFF

- **Purpose**

  Computes coefficients of MA[∞] representation of ARMA[p, q] process for lags 1 to m.

- **Format**

  `c = MACOEFF(a,b,m)`

- **Input**

`a`	$(p, 1)$-vector of ar-coefficients: $1 - \alpha_1 \mathrm{B} - \ldots - \alpha_p \mathrm{B}^p$
`b`	$(q, 1)$-vector of ma-coefficients: $1 - \beta_1 \mathrm{B} - \ldots - \beta_q \mathrm{B}^q$

- **Output**

`c`	$(m, 1)$-vector, ma-coefficients: $1 - \beta_1^* \mathrm{B} - \ldots - \beta_m^* \mathrm{B}^m$

- **Reference**

  McLeod, I. (1975): Derivation of the Theoretical Autocovariance Function of Autoregressive-Moving Average Time Series, Applied Statistics, 24, 255-256

- **Source**

  `ariamfit.src`

MAD

- **Purpose**

  Computes robust scale estimator MAD for univariate samples

- **Format**

  `m = MAD(x)`

- **Input**

  `x` $(n, 1)$-vector

- **Output**

  `m` scalar, estimated MAD

- **Remarks**

  $m = 1.4826 \cdot \text{med}\{|x_j - \text{med}\{x_i : 1 \leq i \leq n\}| : 1 \leq j \leq n\}$
  The factor makes it an unbiased estimator of σ_X in a Gaussian white noise process.

- **Source**

  `auxilary.src`

MISSAR

- **Purpose**

  Interpolation of missing values in a time series by substituting them with the expected values of AR-models

- **Format**

  `{a,y} = MISSAR(x,p)`

- **Input**

`x`	$(n, 1)$-vector, time series with missing values
`p`	integer, maximal order of the ar-models to be used ($p < 18$)

- **Output**

`a`	(p, p)-matrix, the estimated ar-coefficients for the models up to order p
`y`	$(n, 1)$-vector, filled time series

- **Reference**

  Miller, R.B. and Ferreiro, O.M. (1984): A strategy to complete a time series with missing observations; In: E. Parzen (ed.): Time series analysis of irregularly observed data; Berlin: Springer

- **Remarks**

  There must not be missing values at the beginning and the end of the series.
  Set `_iterout_=1` to watch iteration history.

- **Source**

  `missfill.src`

MISSLS

- **Purpose**

  Minimum mean square error interpolation of missing values

- **Format**

  `y = MISSLS(x,tol,theo,p)`

- **Input**

`x`	$(n, 1)$-vector, time series with missing values
`tol`	scalar, control stopping criterion, minimum absolut change in parameter
`theo`	theoretical IACF for interpoltation, or 0, for interpolation with estimated IACF
`p`	estimation of IACF with parameter of AR[p]-Modell 0, choosing $p = n/10$

- **Output**

`y`	$(n, 1)$-vector, filled time series

- **Reference**

  Brubacker, S. and Wilson, G. (1976): Interpolating time series with applications to the estimation of holiday effects on electricity demand; Journal of the Royal Statistical Society, C, 25, 107-116

- **Remarks**

  There must not be missing values at the beginning and the end of the series.

- **Source**

  `missfill.src`

OUTIDENTIFY

- **Purpose**

  Iterative procedure to identify impact, locations and type of outliers in ar processes

- **Format**

  `{alpha,aus,sigma2} = OUTIDENTIFY(y,p,k)`

- **Input**

`y`	$(n, 1)$-vector, time series
`p`	scalar, the order of ar model
`k`	scalar, the level of the tests for deciding which value is to be considered an outlier.

- **Output**

`alpha`	$(p, 1)$-vector, the ar coefficients
`aus`	$(k, 3)$-matrix with information about outliers: `aus[.,1]` = type of outlier 1 = ao, 2=io `aus[.,1]` = impact `aus[.,1]` = time points where outliers were detected
`sigma2`	estimated residual variance

- **Reference**

  Wei, W.W.S.(1990): Time Series Analysis, Univariate and Multivariate Methods; Redwood City: Addison Wesley see also: Chang and Tiao (1983)

- **Source**

  `tsmoment.src`

PACFE

- **Purpose**

  Partial autocorrelation function

- **Format**

  `pa = PACFE(y,m,meth)`

- **Input**

`y`	$(n, 1)$-vector, time series
`m`	scalar, maximum lag
`meth`	string, the method : "burg": Burg's algorithm "orth": Orthogonalization otherwise: Levinson Durbin recursion is used

- **Output**

`pa`	$(m, 1)$-vector, pacf

- **Reference**

  Newton,H.J. and Pagano,M. (1983): Computing for Autoregressions; in: Computer Science and Statistics: The Interface, J.E.Gentle (ed.), North Holland Publishing Company

- **Source**

  `tsmoment.src`

PERIODO1

- **Purpose**

  Determines the periodogram of centered time series at given frequencies

- **Format**

  `per = PERIODO1(y,freq,c)`

- **Input**

`y`	$(n, 1)$-vector, time series
`freq`	$(k, 1)$-vector, frequencies
`c`	scalar; if c=1, with mean correction, else no mean correction

- **Output**

`per`	$(k, 1)$-vector, periodogram

- **Source**

  `spectrum.src`

PERIODO2

- **Purpose**

  Determines the periodogram of centered time series at equally spaced frequencies

- **Format**

  `{per,f} = PERIODO2(y,nfreq)`

- **Input**

`y`	$(n, 1)$-vector, time series
`nfreq`	integer, the number of equally spaced frequencies

- **Output**

`per`	$(nfreq, 1)$-vector, periodogram at frequencies $k/(2 \cdot nfreq - 1)$
`f`	$(nfreq, 1)$-vector, frequencies

- **Source**

  `spectrum.src`

PERIODO3

- **Purpose**

  Determines the periodogram at given frequencies using the acf

- **Format**

  `per = PERIODO3(acf,freq)`

- **Input**

  `acf` $(m+1,1)$-vector, acf at lags $0,1,\dots,m$
  `freq` $(k,1)$-vector, frequencies

- **Output**

  `per` $(k,1)$-vector, periodogram

- **Remarks**

  Use `freq=seqa(0,1/(2*n-1),n` to be able to recover the autocovariancefunction by `acovfper` .

- **Source**

  `spectrum.src`

PERIODO4

- **Purpose**

  Determines the periodogram at equally spaced frequencies using the acf

- **Format**

  `{per,f} = PERIODO3(acf,nf)`

- **Input**

`acf`	$(m, 1)$-vector, acf at lags $0, 1, \ldots, m-1$
`nf`	integer, the number of frequencies

- **Output**

`per`	$(nf+1, 1)$-vector, periodogram at frequencies f
`f`	$(nf+1, 1)$-vector, frequencies $k/(2n_f)$,$k = 0, 1, \ldots, n_f$

- **Remarks**

  Using $m < n$ where n is the length of the series is equal to perform spectral estimation with Daniell window.
  Use `nf=n` with series length n to be able to recover the autocovariancefunction by `acovfper` .

- **Source**

  `spectrum.src`

PERIODORB

- **Purpose**

  Robust determination of periodogram at equally spaced frequencies using regression interpretation

- **Format**

  `{f,p} = PERIODORB(y,&psi)`

- **Input**

`y`	$(n, 1)$-vector, th time series
`psi`	psi-function

- **Output**

`f`	$([n/2] + 1, 1)$-vector, fourier frequencies
`p`	$([n/2] + 1, 1)$-vector, periodogram at fourier frequencies

- **Remarks**

  For Fourier frequencies λ the LS-approach $\sum[(y_t - \bar{y}) - b_1 cos(2\pi\lambda t) - b_2 sin(2\pi\lambda t)]^2 \stackrel{!}{=}$ min leads to solutions $\hat{b}_1, \hat{b}_2$ which satisfy $I(\lambda) = \frac{n}{4}[\hat{b}_1^2 + \hat{b}_2^2]$ where $I(\lambda)$ is the periodogram. In this procedure the mean is replaced by a m-estimate and the ls-regression by m-estimation.

- **Source**

  `tsrobust.src`

RANDAR

- **Purpose**

  Simulates a realisation of a Gaussian AR[p]-process

- **Format**

  `x = RANDAR(a,s,n)`

- **Input**

`a`	$(p, 1)$-vector of AR-coefficients: $1 - \alpha_1 \mathrm{B} - \ldots - \alpha_p \mathrm{B}^p$
`s`	scalar, variance of error process
`n`	integer, length of time series

- **Output**

`x`	$(n, 1)$-vector, time series

- **Reference**

  McLeod, I. A. and Hipel, K. W. (1978): Simulation Procedures for Box-Jenkins Models; Water Resources Research 14, 969-975

- **Source**

  `tssim.src`

RANDARMA

- **Purpose**

  Simulates a realisation of a Gaussian ARMA[p,q]-process

- **Format**

  `x = RANDARMA(a,b,s,n)`

- **Input**

`a`	$(p, 1)$-vector of ar-coefficients: $1 - \alpha_1 \mathrm{B} - \ldots - \alpha_p \mathrm{B}^p$
`b`	$(q, 1)$-vector of ma-coefficients: $1 - \beta_1 \mathrm{B} - \ldots - \beta_q \mathrm{B}^q$
`s`	scalar, variance of error process
`n`	integer, length of time series

- **Output**

`x`	$(n, 1)$-vector, time series

- **Reference**

  McLeod, I. A. and Hipel, K. W. (1978): Simulation Procedures for Box-Jenkins Models; Water Resources Research 14, 969-975

- **Source**

  `tssim.src`

REGAR

- **Purpose**

  Approximative maximum likelihood estimation of a regression with AR[p]-errors

- **Format**

  `{b,a,stdb,stda,corb,cora,s} = REGAR(y,x,p)`

- **Input**

`y`	$(n, 1)$-vector, the (dependent) time series
`x`	(n, m)-matrix of regressors or scalar =1 for constant, =0 without constant
`p`	scalar, order of ar-model

- **Output**

`b`	$(m, 1)$-vector, regression-coefficients
`a`	$(p, 1)$-vector, ar-coefficients
`stdb`	$(m, 1)$-vector, standard deviations of `b`
`stda`	$(p, 1)$-vector, standard deviations of `a`
`corb`	(m, m)-matrix, the correlation matrix of `b`
`cora`	(p, p)-matrix, the correlation matrix of `a`
`s`	scalar, error variance

- **Reference**

  Fuller, W.A. (1996): Introduction to statistical time series; New York: Wiley

  Brockwell, P.J. and Davis, R.A. (1987): Time series: theory and methods; New York: Springer

- **Remarks**

  The model is $y_t = \mathbf{x}_t'\boldsymbol{\beta} + z_t, \;\; t = 1, \ldots, n$ where $z_t = \sum_{j=1}^{p} \alpha_j z_{t-j} + \epsilon_t$

  The solution is found by iterative generalized least squares.

  Gram-Schmid orthogonalization is used to obtain GLS estimates of beta.

  b and **a** are asymptotically uncorrelated.

  Set `_iterout_=1` to watch iteration history

- **Source**

  `regar.src`

REGARMA

- **Purpose**

  Maximum likelihood estimation of a regression model with ARMA[p,q]-errors

- **Format**

  `{b,a,th,stdb,stda,stdth,corb,cora,s} = REGARMA(y,x,p,q)`

- **Input**

`y`	$(n, 1)$-vector, the (dependent) time series
`x`	(n, m)-matrix of regressors or scalar =1 for constant, =0 without constant
`p`	scalar, the order of the ar-polynom
`q`	scalar, the order of the ma-polynom

- **Output**

`b`	$(m, 1)$-vector, regression-coefficients
`a`	$(p, 1)$-vector, ar-coefficients
`th`	$(q, 1)$-vector, ma-coefficients
`stdb`	$(m, 1)$-vector, standard deviations of `b`
`stda`	$(p, 1)$-vector, standard deviations of `a`
`stdth`	$(q, 1)$-vector, standard deviations of `th`
`corb`	(m, m)-matrix, the covariance matrix of `b`
`cora`	$(p + q, p + q)$-matrix, the covariance matrix of `(a,th)`
`s`	scalar, error variance

- **Reference**

  M. A. Wincek and G.C. Reinsel (1986): An exact maximum likelihood estimation procedure for regression-ARMA time series models with possibly nonconsecutive data; J. R. Statist. Soc. B, 48, 303-313

- **Remarks**

  The model is $y_{t_i} = \mathbf{x}'_{t_i}\boldsymbol{\beta} + z_{t_i},\ i = 1, \dots, n$ where $t_1 < t_2 < \cdots < t_n$ and $z_t = \sum_{j=1}^{p} \alpha_j z_{t-j} + \epsilon_t - \sum_{j=1}^{q} \theta_j \epsilon_{t-j}$
  The initial values are computed via the method by Durbin-Hannan-Rissanen .
  The algorithm is likely to diverge when the starting values do not belong to a stationary and invertible model or when the model is grossly misspecified.
  b and $(\mathbf{a}, \mathbf{th})$ are asymptotically uncorrelated.
  Set `_iterout_=1` to watch iteration history

- **Source**

  `regarma.src`

ROOTCHECK

- **Purpose**

  Computing the norms of the roots of $1 - a_1 z - \ldots - a_p z^p = 0$

- **Format**

  `r = ROOTCHECK(a)`

- **Input**

  `a` $(p, 1)$-vector, coefficients $a_1, a_2, \ldots, a_p$

- **Output**

  `r` $(p, 1)$-vector, norms of the roots

- **Remarks**

  Non existing coefficients in a must be set to zero.

- **Source**

  `arimafit.src`

RUNMEAN

- **Purpose**

  Time series smoothing by simple moving averages

- **Format**

  `g = RUNMEAN(y,q)`

- **Input**

`y`	$(n, 1)$-vector, time series to be smoothed
`q`	integer, span of moving average

- **Output**

`g`	$(n, 1)$-vector, smooth component

- **Remarks**

  Ends are filled with missings.

- **Source**

  `tsdecomp.src`

RUNMED

- **Purpose**

  Time series smoothing by moving median

- **Format**

  `g = RUNMED(y,q)`

- **Input**

`y`	$(n, 1)$-vector, time series to be smoothed
`q`	integer, span of moving median; q must be odd.

- **Output**

`g`	$(n, 1)$-vector, smooth component

- **Remarks**

  Ends are filled with missings.

- **Source**

  `tsdecomp.src`

SEASUBPLOT

- **Purpose**

  Plot of seasonal subseries

- **Format**

  `SEASUBPLOT(y,d)`

- **Input**

`y`	$(n, 1)$-vector, the time series
`d`	integer, seasonal period

- **Remarks**

  For the seasonal subseries $y_i, y_{i+d}, y_{i+2d}, \ldots$ the means and the corresponding values are shown.
  y may also be the smooth component + seasonal component or the pur seasonal component of a time series.
  If y is the pure seasonal component the means should approimatively be equal to zero.

- **Source**

  `tsplot.src`

SIGN

- **Purpose**

  Sign of numeric variables

- **Format**

  `y = SIGN(x)`

- **Input**

`x`	(n, k)-matrix

- **Output**

`y`	(n, k)-matrix, consisting of -1's and 1's

- **Source**

  `auxilary.src`

SMOOTHLS

- **Purpose**

  Smoothing of time series by least square spline

- **Format**

  `g = SMOOTHLS(y,beta)`

- **Input**

`y`	$(n, 1)$-vector, time series
`beta`	scalar, smoothing parameter

- **Output**

`g`	$(n, 1)$-vector, smooth component

- **Remarks**

  The bigger *beta* is, the smoother g will be.
  This smoothing spline is sometimes called Hodrick-Prescott filter.

- **Source**

  `tsdecomp.src`

SMOOTHRB

- **Purpose**

  Smoothing of time series by robust spline

- **Format**

  `g = SMOOTHRB(y,beta,&psi)`

- **Input**

`y`	$(n, 1)$-vector, time series
`beta`	scalar, smoothing parameter
`&psi`	function, psi-function

- **Output**

`g`	$(n, 1)$-vector, smooting component

- **Reference**
  For the psi functions see Spaeth, H. (1987): Mathematische Software zur linearen Regression, p.198, Munich, Oldenbourg

- **Remarks**
  The bigger *beta* is, the smoother g will be.

  Possible psi-functions:
  `psibiw, psihamp, psihub, psifair, psithg`

- **Source**

  `tsrobust.src`

SPECARMA

- **Purpose**

  Computes the theoretical spectrum of an ARMA[p,q]-process

- **Format**

  `{sp,fr} = SPECARMA(a,b,s,n)`

- **Input**

`a`	$(p, 1)$-vector of ar-coefficients: $1 - \alpha_1 \mathrm{B} - \ldots - \alpha_p \mathrm{B}^p$
`b`	$(q, 1)$-vector of ma-coefficients: $1 - \beta_1 \mathrm{B} - \ldots - \beta_q \mathrm{B}^q$
`s`	scalar, variance of error process
`n`	integer, the number of equally spaced frequencies

- **Output**

`sp`	$(n + 1, 1)$-vector, spectrum
`fr`	$(n + 1, 1)$-vector, frequencies

- **Source**

  `spectrum.src`

SPECESTAR

- **Purpose**

  Autoregessive spectral estimation with simultaneous confidence bands

- **Format**

  `{s,bound,f} = SPECESTAR(y,p,nf,conf)`

- **Input**

`y`	$(n, 1)$-vector, time series
`p`	scalar, order of ar-model used for the estimation
`nf`	scalar, number of eqally spaced frequencies where the spectrum is estimated
`conf`	scalar, the level for the confidence bounds

- **Output**

`s`	$(nf + 1, 1)$-vector, estimated spectrum
`bound`	$(nf + 1, 2)$-matrix, confidence bounds for the spectrum
`f`	$(nf + 1, 1)$-vector, frequencies

- **Reference**

  Newton, H.J. and Pagano, M. (1984): Simultaneous confidence bands for autoregressive spectra; Biometrika, 71, 197-202

  Newton, H.J. (1988): Timeslab, a time series laboratory; Belmont, CA. : Wadsworth & Brooks/Cole

- **Remarks**

  If no bounds are desired, set `conf` equal to 0.

  The series needs to be rahther long for the upper bounds to be finite (otherwise the upperbound is set missing).

- **Source**

  `spectrum.src`

SPECESTD

- **Purpose**

  Direct spectral estimation of time series using periodogram window

- **Format**

  `{s,f} = SPECESTD(y,e,&wind,conf)`

- **Input**

`y`	$(n, 1)$-vector, time series
`e`	scalar, equal bandwidth, e must be $0 \le e < 0.5$
`wind`	function, periodogram window
`conf`	scalar, the level for confidence intervals

- **Output**

`s`	$([n/2]+1, 3)$-matrix, estimated spectrum at fourier frequencies $0, 1/n, \ldots$ with lower and upper confidence limits
`f`	$([n/2]+1, 1)$-vector, frequencies

- **Remarks**

  Possible periodogram windows:
  Bartlett's window: `perwinba`
  Parzen's window: `perwinpa`
  Tukey's window: `perwintu`
  truncated periodogram window: `perwintp`

- **Source**

  `spectrum.src`

SPECESTI

- **Purpose**

  Indirect spectral estimation of time series using lag window

- **Format**

  `{s,f} = SPECESTI(y,q,&wind)`

- **Input**

`y`	$(n, 1)$-vector, time series
`q`	integer, number of covariances used for averaging
`wind`	function, lag window

- **Output**

`s`	$([n/2]+1, 1)$-vector, estimated spectrum at fourierfrequencies $0, 1/n, \ldots, [n/2]/n$
`f`	$([n/2]+1, 1)$-vector, frequencies

- **Remarks**
  Possible lag windows:
  Bartlett's window: `lagwinba`
  Parzen's window: `lagwinpa`
  Tukey's window: `lagwintu`

- **Source**

  `spectrum.src`

SPLINEDECOMP

- **Purpose**

  Spline decomposition of a time series

- **Format**

  {g,s} = `SPLINEDECOMP(x,d,alpha,beta)`

- **Input**

`x`	$(n, 1)$-vector, time series
`d`	integer, seasonal period
`alpha`	scalar, weight for smooth component
`beta`	scalar, weight for seasonal component

- **Output**

`g`	$(n, 1)$-vector, smooth component
`s`	$(n, 1)$-vector, seasonal component

- **Source**

  `tsdecomp.src`

SYMPLOT

- **Purpose**

  Symmetry plot of time series

- **Format**

  `SYMPLOT(x)`

- **Input**

  `x` $(n, 1)$-vector, time series

- **Source**

  `tsplot.src`

TAKE

- **Purpose**

  Selecting the first or last rows of a matrix

- **Format**

  `ys = TAKE(y,q)`

- **Input**

`y`	(n, k)-matrix
`q`	integer, number of rows to be taken out of `y` $q > 0$, the first q rows are taken $q == 0$, an empty vector is returned $q < 0$, the last q rows are taken

- **Output**

`ys`	(q, k)-matrix, consisting of the selected rows

- **Remarks**

  If $|q| \geq n$, then $ys = y$.

- **Source**

  `auxilary.src`

TAPER

- **Purpose**

  Tapermodification of a time series using a cosinus-taper

- **Format**

  `z = TAPER(y,part)`

- **Input**

`y`	$(n, 1)$-vector, time series
`part`	scalar, part of time series to be modified

- **Output**

`z`	$(n, 1)$-vector, modified time series

- **Remarks**

  Part must be $0 \leq part \leq 0.5$

- **Source**

  `linfilt.src`

TRENDEXTRAPOL

- **Purpose**

  Trendextrapolation of a time series

- **Format**

  `{pred,low,up,beta} = TRENDEXTRAPOL(y,p,alpha,lead)`

- **Input**

`y`	$(n, 1)$-vector, time series
`p`	scalar, the order of the trend polynom
`alpha`	probability to determine $(1-\alpha)$-prediction intervals
`lead`	scalar, prediction horizon

- **Output**

`pred`	$(lead, 1)$-vector, the forecaste for time points n+1,...,n+lead
`low`	$(lead, 1)$-vector, lower bounds of prediction intervals
`up`	$(lead, 1)$-vector, upper bounds of prediction intervals
`beta`	$(p+1, 1)$ vector, the estimated coefficients for $t^0, t^1, \ldots, t^p$

- **Remarks**

  The errors are assumed to be white noise

- **Source**

  `predict.src`

TSLOESS

- **Purpose**

  Robust smoothing by of time series with `LOESS`

- **Format**

  `{t,yhat} = TSLOESS(y)`

- **Input**

`y`	$(n, 1)$-vector, the time series

- **Output**

`t`	$(n, 1)$-vector of time indices $(1, 2, 3, \ldots, n)$
`yhat`	$(n, 1)$-vector, robustly estimated smooth component

- **Remarks**

  The procedure uses the GAUSS procedure `LOESS`;
  missing values are allowed

- **Source**

  `tsrobust.src`

TSMEAN

- **Purpose**

  Computes mean values of time series

- **Format**

  `m = TSMEAN(x)`

- **Input**

  `x` (n, k)-matrix, k time series of length n

- **Output**

  `m` $(k, 1)$-vector, mean values

- **Remarks**
  Allowing missing values.
  Means are determind via the formula `MEANC(y-MEANC(y))+MEANC(y)` to achive greater accuracy.

- **Source**

  `tsmoment.src`

TSMEANRB

- **Purpose**
  M-estimation of location parameter by iteratively reweighted least squares
- **Format**
  `m = TSMEANRB(x,&psi)`
- **Input**
  `x` $(n, 1)$-vector, the time series
  `psi` psi-function
- **Output**
  `m` scalar, robustly estimated level
- **Reference**
  H.Spaeth: Mathematische Software zur linearen Regression, p.198
- **Remarks**
  Possible psi-functions: `psibiw`, `psihamp`, `psihub`, `psifair`, `psithg`
- **Source**
  `tsrobust.src`

TSMEDIAN

- **Purpose**

  Computes median of time series

- **Format**

  `m = TSMEDIAN(y)`

- **Input**

`y`	(n, k)-matrix, k time series of length n

- **Output**

m	$(k, 1)$-vector, median values

- **Remarks**

  Allowing missing values.
  Missings are eliminated before `MEDIAN` is called. This gives more adequate results.

- **Source**

  `tsmoment.src`

TSPLOT

- **Purpose**

  Plot of time series

- **Format**

  `TSPLOT(y,step,start)`

- **Input**

`y`	(n, p)-matrix, k time series of length n
`step`	string, the period of the time series: "year", "quart", "month" other periodicities are displayed as label only
`start`	integer, giving the start value of the time step = "year": start must be of the form yy or yyyy step = "quart": start must be of the form qyy step = "month": start must be of the form myy or mmyy

- **Remarks**

  The first column of `y` will be used to determine the ticmarks of the abscissa

- **Source**

  `tsplot.src`

TSPLOT1

- **Purpose**

  Plot of a time serie with optimization of viewratio

- **Format**

  `TSPLOT1(x)`

- **Input**

  `x` $(n, 1)$-vector, time series

- **Reference**

  Cleveland, W.S. (1993): A model for studying display methods of statistical graphics; J. Comp. Graph. Statistics, 2, 323-343

- **Remarks**

  The plot is scaled in a way that the mean angel of ascent / decent is aproximately 45^o

- **Source**

  `tsplot.src`

TSVAR

- **Purpose**

  Computes variance of time series

- **Format**

  `s = TSVAR(y)`

- **Input**

  `y` (n, k)-matrix, k time series of length n

- **Output**

  `s` $(k, 1)$-vector, the variances

- **Remarks**
  Allowing missing values.
  Variances are determined via the formula $\sum(y_i - \bar{y})^2/(n-1)$

- **Source**

  `tsmoment.src`

VARDIFF

- **Purpose**

  computing a table of variance ratios for determing the proper orders of differencing of a time series to reach stationarity

- **Format**

  ```
  VARDIFF(y,s)
  ```

- **Input**

`y`	$(n, 1)$-vector, time series
`s`	integer, seasonal period

- **Output**

  Print of a table of variance ratios

- **Remarks**

  Allowing missing values.
  Table gives $\widehat{\text{var}}((1-B)^d(1-B^s)^D y_t) \,/ \widehat{\text{var}}(y_t)$ for $0 \leq d, D \leq 3$

- **Source**

  `tsmoment.src`

VARIOGRAM

- **Purpose**

  Variogram for unequal spaced time series

- **Format**

  `v = VARIOGRAM(y,mlag,tol)`

- **Input**

`y`	$(n, 2)$-matrix, the time points (first column) and values of time series (second column)
`mlag`	integer, maximal time distance for variogram
`tol`	scalar, window width over which the values are averaged

- **Output**

`v`	$(k, 2)$-matrix, lags (first column) and variogram (second column)

- **Reference**

  Diggle, P.J. (1990): Time Series; Oxford: Clarendon Press

- **Source**

  `tsmoment.src`

VECCORE

- **Purpose**

  Computes vectorcorrelations with bootstrap standard deviations

- **Format**

  `{vec,st} = VECCORRE(y,p,q)`

- **Input**

`y`	$(n, 1)$-vector, the time series
`p`	integer, maximal order of AR-part
`q`	integer, maximal order of MA-part

- **Output**

`vec`	$(p+1, q+1)$-matrix, vectorcorrelations
`st`	$(p+1, q+1)$-matrix, standard deviations

- **Reference**

  Paparoditis, E. (1990): Vektorautokorrelationen stochastischer Prozesse und die Spezifikation von ARMA-Modellen; Heidelberg: Physica-Verlag

  Paparoditis, E. and B.H.J. Streitberg (1991): Order identification statistics in stationary autoregressive-moving average models: vector autocorrelations and the bootstrap; Journal of Time Series Analysis 13, 415-434

- **Remarks**

  Additionally a table of the vectorcorrelations is printet; the entries are marked, where the confidence intervals $vec_{i,j} \pm 2 \cdot st_{i,j}$ do not contain zero.

- **Source**

  `tsmoment.src`

WNTEST

- **Purpose**

  Graphical white noise test for a time series or a series of regression residuals

- **Format**

  `WNTEST(e,alpha,k)`

- **Input**

`e`	$(n, 1)$-vector, the time series or residuals
`alpha`	scalar, level of significance
`k`	integer, number of regressors

- **Output**

  A plot is drawn.

- **Remarks**

  If $k = 0$, e is interpreted as time series without regressors
  The Kolmogorov test by Bartlett/Durbin with residuals accepts randomness when the cumulative periodogram remains inside the inner boundary. The test rejects this hypothesis when the peridogramm crosses the outer boundary. Otherwise it is inconclusive.

- **Source**

  `spectrum.src`