Doing TIMESLAB in S

H. Joseph Newton and Jane Harvill
Department of Statistics, Texas A & M University

Ported to R by:
Bernd Johannes Wuebben
Department of Mathematics, Cornell University
<wuebben@math.cornell.edu>

1. Introduction

TIMESLAB (see Newton (1988)) is an interactive, graphical oriented, command driven language for studying and analyzing univariate and bivariate linear time series in the time and frequency domain. It is written to obtain maximum performance from the IBM PC family of computers. Thus it is very difficult to port to other computer platforms. The S language (see Becker, et. al. (1988)) is also an interactive, graphics oriented command driven language. It is designed to run on a wide variety of Unix workstations and contains a large number of input/output, mathematical, and graphics routines. Unfortunately, it only has a few time series functions. Since S is extensible (a new version can be created by the user that incorporate Fortran or C functions that they have written), one way to transport the *abilities* of TIMESLAB is to embed it in S. Thus we have created a set of S functions that correspond to the commands in TIMESLAB.

In Section 2, we describe the files that are distributed with the add-on library and discuss what must be done to make them part of S. Section 3 is a dictionary of the S functions. This dictionary is designed to be in the same format as that in Appendix 1 of the S book. Finally, Section 4 contains a table of cross-references between TIMESLAB commands and the equivalent S functions. We also list a variety of other S functions that are included in the distribution. These functions include analogs of some of the TIMESLAB macros as well as some other functions that I have developed.

2. Incorporating the New Functions into S

The new functions are distributed in two files; one contains a large number of S functions while the other contains a number of fortran subprograms. In order to use these functions, a user must 1) use the S `source` function and 2) compile and dynamically load the fortran. The table in Section 4 describes which of the S functions use fortran.

3. Dictionary of S Functions for Time Series

`acf`	Calculate Sample Autocorrelation Function	`acf`

```
acf(x,m=0)
```

ARGUMENTS

`x`	A time series.
`m`	The number of lags at which to find the acf.

VALUE

`acf` returns a list containing two elements:

`corr`	A vector containing the autocorrelations.
`var`	A scalar containing the sample variance.

`acf1`	Calculate Sample Autocorrelation Function	`acf1`

```
acf1(x,m=0)
```

ARGUMENTS

`x`	A time series.
`m`	The number of lags at which to find the autocovariance function.

VALUE

`acf1` returns a list containing two elements:

`corr`	A vector containing the autocorrelations.
`var`	A scalar containing the sample variance.

`arcorr`	Calculate AR Autocorrelation Function	`arcorr`

```
arcorr(alpha,rvar=1,m=0)
```

ARGUMENTS

`alpha`	Array containing AR coefficients $\boldsymbol{\alpha}$.
`rvar`	Real scalar containing error variance $\sigma^2 (> 0)$.
`m`	Integer containing the number of autocorrelations to calculate (≥ 0).

VALUE

`arcorr` returns a list containing the following three items:

`var`	Real scalar containing the variance of the process.
`corr`	Array of length `m` containing the autocorrelations.
`ier`	Integer variable indicating whether or not the AR process is stationary (0 means yes, anything else means no).

`ardt`	Simulate Data from an AR Process	`ardt`

```
ardt(alpha,rvar,n,seed=0)
```

ARGUMENTS

alpha Array of length p containing AR coefficients $\boldsymbol{\alpha}$.

rvar Real scalar containing error variance $\sigma^2 (> 0)$.

n Integer $(> p)$ containing the length of the realization.

seed Real scalar containing the seed for the random number generator.

VALUE

ardt returns a list containing the following two items:

ier Integer variable indicating whether or not the AR process is stationary (0 means yes, anything else means no).

x Array of length n containing the realization.

arfilt	Apply an AR Filter to a Matrix	arfilt

```
arfilt(alpha,rvar,x)
```

ARGUMENTS

alpha Array containing AR coefficients $\boldsymbol{\alpha}$.

rvar Real scalar containing error variance $\sigma^2 (> 0)$.

x Array containing the matrix to be filtered.

VALUE

arfilt returns a list containing the following two items:

w Matrix containing the filtered version of x.

ier Integer variable indicating whether or not the AR process is stationary (0 means yes, j (> 0) means jth partial outside $(-1, 1)$).

arma	Form Plots Illustrating Patterns in ARMA Processes	arma

```
arma(alpha,beta,x,iopt,p,q,rvar,n,m,seed=0)
```

ARGUMENTS

alpha Array of length p containing AR coefficients $\boldsymbol{\alpha}$.

beta Array of length q containing MA coefficients $\boldsymbol{\beta}$.

x An ARMA process.

iopt Integer indicating which part of the ARMA process is to be simulated. iopt = 0 means to use input alpha, beta and x, iopt = 1 means to use input alpha and beta and simulate x, and iopt = 2 means to simulate alpha, beta and x.

p Integer containing order p of the array $\boldsymbol{\alpha}$.

q Integer containing order q of the array $\boldsymbol{\beta}$.

rvar Real scalar containing error variance $\sigma^2 (> 0)$.

n Length of the realization.

m Number of autocorrelations to be calculated.

seed Real scalar containing the seed for the random number generator.

VALUE

`arma` returns plots illustrating patterns in ARMA processes and a list containing the following three items:

- `alpha` Array containing the AR coefficients.
- `beta` Array containing the MA coefficients.
- `x` The realization.

`armacorr`	Calculate ARMA Autocorrelation Function	`armacorr`

```
armacorr(alpha,beta,rvar=1,m)
```

ARGUMENTS

- `alpha` Array of length p containing AR coefficients $\boldsymbol{\alpha}$.
- `beta` Array of length q containing MA coefficients $\boldsymbol{\beta}$.
- `rvar` Real scalar containing error variance $\sigma^2 (> 0)$.
- `m` Integer ($\geq \max(p, q)$) containing the number of autocorrelations to calculate.

VALUE

`armacorr` returns a list containing the following three items:

- `var` Real scalar containing variance of process.
- `corr` Array of length `m` containing autocorrelations $\rho(1), \ldots, \rho(m)$
- `ier` Integer variable indicating whether or not the ARMA process is stationary (0 means yes, anything else means no).

`armadt`	Simulate Data from an ARMA Process	`armadt`

```
armadt(alpha,beta,rvar,n,seed=0)
```

ARGUMENTS

- `alpha` Array of length p containing AR coefficients $\boldsymbol{\alpha}$.
- `beta` Array of length q containing MA coefficients $\boldsymbol{\beta}$.
- `rvar` Real scalar containing error variance $\sigma^2 (> 0)$.
- `n` Integer ($> \max(p, q)$) containing the length of the realization.
- `seed` Real scalar containing the seed for the random number generator.

VALUE

`armadt` returns a list containing the two elements:

- `x` Array of length `n` containing the realization.
- `ier` Integer variable indicating whether or not the ARMA process is stationary (0 means yes, anything else means no).

`armapart`	Calculate ARMA Partial Autocorrelation Function	`armapart`

```
armapart(alpha,beta,rvar,m)
```

ARGUMENTS

`alpha`	Array containing AR coefficients $\boldsymbol{\alpha}$.
`beta`	Array containing MA coefficients $\boldsymbol{\beta}$.
`rvar`	Real scalar containing error variance $\sigma^2(> 0)$.
`m`	Integer indicating the number of partial autocorrelations to calculate.

VALUE

`armapart` returns a list containing the following two elements:

`theta`	Array of length `m` containing the partial autocorrelations.
`ier`	Integer variable indicating whether or not the ARMA process is stationary (0 means yes, anything else means no)..

`armapred`	Calculate Exact Predictions for an ARMA Process	`armapred`

```
armapred(x,alpha,beta,rvar,t1,t2,h1,h2)
```

ARGUMENTS

`x`	Array of length n containing the realization to be used in the prediction.
`alpha`	Array containing AR coefficients $\boldsymbol{\alpha}$.
`beta`	Array containing MA coefficients $\boldsymbol{\beta}$.
`rvar`	Real scalar containing error variance $\sigma^2(> 0)$.
`t1,t2`	Integers ($1 \leq \texttt{t1} \leq \texttt{t2} \leq n$) specifying the range of memories to be used.
`h1,h2`	Integers ($1 \leq \texttt{h1} \leq \texttt{h2}$) specifying the horizons to be used.

VALUE

`armapred` returns a list containing the following elements:

`xp`	Array of length $(\texttt{t2} - \texttt{t1} + 1)(\texttt{h2} - \texttt{h1} + 1)$ containing predictors.
`se`	Real scalar containing the prediction standard errors for the predictors in the array `xp`.

`armasp`	Calculate ARMA Spectral Density Function	`armasp`

```
armasp(alpha,beta,rvar=1,Q=256)
```

ARGUMENTS

`alpha`	Array of length p containing AR coefficients $\boldsymbol{\alpha}$.
`beta`	Array of length q containing MA coefficients $\boldsymbol{\beta}$.
`rvar`	Real scalar containing error variance $\sigma^2(> 0)$.
`Q`	Integer ($\geq \max(p, q)$) containing the number of frequencies between 0 and 1 at which to calculate the spectral density.

VALUE

`armasp` returns an array `f` of length $m = [\mathtt{Q}/2] + 1$ containing the values of the spectral density at the frequencies $(j-1)/\mathtt{Q}, j = 1, \ldots, m$.

`arpart`	Calculate AR Partial Autocorrelation Function	`arpart`

`arpart(alpha)`

ARGUMENTS

`alpha` Array of length p containing AR coefficients $\boldsymbol{\alpha}$.

VALUE

`arpart` returns a list containing the following two elements:

`theta` Array of length p containing partial autocorrelations.

`ier` Integer variable indicating whether the zeros of the characteristic polynomial corresponding to `alpha` are all outside the unit circle (0 means they are, anything else means they are not.)

`arsp`	Calculate AR Spectral Density	`arsp`

`arsp(alpha,rvar=1,Q=256)`

ARGUMENTS

`alpha` Array of length p containing AR coefficients $\boldsymbol{\alpha}$.

`rvar` Real scalar containing error variance $\sigma^2(>0)$.

`Q` Integer ($>p$) containing the number of frequencies between 0 and 1 at which to calculate the spectral density.

VALUE

`arsp` returns the array `f` of length $m = [\mathtt{Q}/2] + 1$ containing the values of the spectral density at the frequencies $(j-1)/\mathtt{Q}, j = 1, \ldots, m$.

`arsppeak`	Find Peak Frequencies in AR Spectra	`arsppeak`

`arsppeak(alpha,rvar,n,start=0)`

ARGUMENTS

VALUE

`alpha` Array of length `p` containing AR coefficients $\boldsymbol{\alpha}$.

`rvar` Real scalar containing error variance $\sigma^2(>0)$.

`n` If `ARSPPEAK` is being used for estimation purposes, `n` is an integer containing the length of the realization that was used to estimate the parameters of the process. If the parameters are the true values, let $n = 1$.

`start` An optional argument that is a real scalar containing a starting value ($0 <$ `start` $< .5$) for the maximum finding procedure.

VALUE

`arsppeak` returns a list containing the following three items:

`ier` An integer indicating whether or not ARSPPEAK was successful in finding a peak. The possible values of `ier` are:

- 0 ARSPPEAK was successful in finding a peak.
- 1 ARSPPEAK judged that the AR spectral density has no relative maxima.
- 2 A zero second derivative was encountered.
- 3 The maximum finder converged to frequency 0 or 0.5.
- 4 The maximum finder didn't converge.

`peakf` If `ier` $= 0$, the peak frequency.

`se` If `ier` $= 0$, the standard error of the peak frequency estimator.

`clip`	Clip a Vector Above and Below	`clip`

```
clip(x,low=-1.e20,up=1e.20)
```

ARGUMENTS

`x` A vector.

`low` The value that will replace any element of `x` that is less than `low`.

`up` The value that will replace any element of `x` that is greater than `up`.

VALUE

`x` The original `x` with any element less that `low` replaced by `low` and any element greater than `up` replaced by `up`.

`coeffcsd`	Calculate Asymptotic Standard Errors of ARMA MLE's	`coeffcsd`

```
coeffcsd(alpha,beta,n)
```

ARGUMENTS

`alpha` Array of length p containing AR coefficients $\boldsymbol{\alpha}$.

`beta` Array of length q containing MA coefficients $\boldsymbol{\beta}$.

`n` Integer containing the sample size of process.

VALUE

`coeffcsd` returns the array containing the asymptotic standard deviations of the coefficients if `ier`, the error indicator, returns a value of 0, indicating a nonsingular matrix.

`corrar`	Calculate AR Parameters from Autocorrelations	`corrar`

```
corrar(rho,R0,p)
```

ARGUMENTS

`rho`	Array of length `p` containing autocorrelations.
`R0`	Real scalar containing sample variance (> 0).
`p`	Integer containing the AR order (> 0).

VALUE

`corrar` returns a list containing the following two elements:

`rvar`	Real scalar variable containing error variance.
`alpha`	Array of length `p` containing AR coefficients.

`corrarma` Calculate ARMA Parameters from Autocorrelations `corrarma`

```
corrarma(rho,r0,p,q,maxit=100,del=1.e-5)
```

ARGUMENTS

`rho`	Array of length `p` + `q` containing the autocorrelations of the process.
`r0`	Real scalar containing the variance of the process (> 0).
`p`	Integer containing AR order $p (> 0)$.
`q`	Integer containing MA order $q (> 0)$.
`maxit`	Integer containing the maximum number of iterations to allow in Wilson's algorithm.
`del`	Real scalar containing convergence criterion (> 0).

VALUE

`corrarma` returns a list containing the following items:

`alpha`	Array of length `p` containing AR coefficients.
`beta`	Array of length `q` containing MA coefficients.
`rvar`	Real scalar containing the error variance σ^2.
`ier`	Integer variable containing an error/convergence indicator. The following values are possible: 0 `CORRARMA` successfully found the ARMA parameters. 1 A singular matrix was encountered trying to find AR parameters. 2 Wilson's algorithm for finding the MA parameters didn't converge.

`corrdt` Simulate Data Having Specified Autocorrelations `corrdt`

```
corrdt(rho,r0,n,seed=0)
```

ARGUMENTS

`rho`	Array containing autocorrelations..
`r0`	Real scalar containing the variance of the process (> 0).
`n`	Length of the desired realization.

seed Real scalar containing the seed for the random number generator.

VALUE

`corrdt` returns a realization of length `n` from a Gaussion process having variance `ro` and autocorrelations `rho`.

corrma	Calculate MA Parameters from Autocorrelations	corrma

```
corrma(rho,r0,q,maxit=100,del=1.e-5)
```

ARGUMENTS

- `rho` Array of length `q` containing autocorrelations of lags `1`, ..., `q`.
- `r0` Real scalar containing the variance of the MA process.
- `q` Integer containing order $q (> 0)$.
- `maxit` Integer containing the maximum number of iterations to use in Wilson's algorithm (> 0).
- `del` Real scalar containing the convergence criterion to use in Wilson's algorithm (> 0).

VALUE

`corrma` returns a list containing the following three items:

- `beta` Array of length `q` containing MA coefficients.
- `rvar` Real scalar containing the error variance σ^2 of the MA process.
- `ier` Integer variable indicating whether or not Wilson's algorithm converged (0 means yes, 1 means no).

crlag	Apply the Circular Shift Operator to a Vector	crlag

```
crlag(x)
```

ARGUMENTS

- `x` An array.

VALUE

`crlag` returns the array that results from applying the circular shift operator to the original array.

delay	Make S Pause for a Specified Time	delay

```
delay(seconds)
```

ARGUMENTS

- `seconds` The specified amount of time for the delay.

VALUE

`delay` causes S to pause for the specified time `seconds`.

`diffeq`	Find Future Values of a Difference Equation	`diffeq`

`diffeq(alpha,p,n,e)`

ARGUMENTS

- `alpha` Array of length `p` containing AR coefficients $\boldsymbol{\alpha}$.
- `p` Integer containing order $p(> 0)$.
- `n` Integer ($>$ `p`) containing the length of the realization.
- `e` Array of length `n` containing the values for the general difference equation.

VALUE

- `x` Array of length `n` containing the realization.

`divpoly`	Divide Two Polynomials	`divpoly`

`divpoly(num,den,n)`

ARGUMENTS

- `num` Array containing coefficients of the numerator polynomial whose zeroth coefficient is one.
- `den` Array containing coefficients of the denominator polynomial with zeroth coefficient equal to one.
- `n` Integer containing the order of resulting polynomial.

VALUE

`divpoly` returns the array `ratio` of length `n` containing the coefficients of the polynomial that results from the division the original two polynomials.

`dot`	Calculate Inner Product of Two Vectors	`dot`

`dot(x,y)`

ARGUMENTS

- `x` A vector.
- `y` A vector.

VALUE

`dot` returns the inner product of the two vectors `x` and `y`.

`dtarma`	Calculate Exact ARMA MLE's	`dtarma`

C

```
dtarma(x,alpha,beta,maxit=200,eps=.0001)
```

ARGUMENTS

- `x` Array containing the data to be used in the estimation of the procedure.
- `alpha` Array containing the starting value for the AR coefficients.
- `beta` Array containing the starting value for the MA coefficients.
- `maxit` Integer ($1 \leq$ `maxit` ≤ 500) containing the maximum number of iterations in the optimization procedure.
- `eps` Real scalar containing the convergence criterion.

VALUE

`dtarma` returns a list containing the following six items:

- `ier` Integer variable containing termination information (0 means convergence, > 1 means some convergence error.)
- `alpha` Array containing the values of the AR coefficients at termination.
- `beta` Array containing the values of the MA coefficients at termination.
- `rvar` Real scalar variable containing the value of the error variance at termination.
- `m2ll` Real scalar variable containing the value of -2 times the log likelihood evaluated at the output values of the parameters.
- `var` Real scalar containing a measure of degree of convergence at termination.

`filt`	Apply a Linear Filter to a Vector	`filt`

```
filt(beta,beta0,x)
```

ARGUMENTS

- `beta` Array of length p containing coefficients of lags $1, \ldots, p$.
- `beta0` Real scalar containing the coefficient for lag 0.
- `x` Array of length n containing the data to be filtered.

VALUE

`filt` returns an array of length $n - p$ containing the result of the filter.

`freqs`	Form Vector of Natural (Nyquist) Frequencies	`freqs`

```
freqs(n)
```

ARGUMENTS

- `n` Integer indicating the number of desired frequencies.

VALUE

`freqs` returns an array of length `n` of frequencies all contained in the interval [0,0.5].

`infqnt`	Plot Informative Quantile Function of a Data Set	`infqnt`

```
infqnt(x)
```

ARGUMENTS

`x` Array of length n containing the data.

VALUE

`infqnt` returns a plot of the informative quantile function for the data set `x`.

`madt`	Simulate Data from an MA Process	`madt`

```
madt(beta,rvar,n,seed=0)
```

ARGUMENTS

`beta` Array of length q containing the MA coefficients $\boldsymbol{\beta}$.

`rvar` Real scalar containing the error variance $\sigma^2 (> 0)$.

`n` Integer $(> q)$ containing the desired number of observations.

VALUE

`madt` returns an array `x` of length `n` containing the realization.

`masmooth`	Apply Moving Average Smoother to a Vector (S Version)	`masmooth`

```
masmooth(x,k)
```

ARGUMENTS

`x` Array containing the data.

`k` Length of the moving average smoother.

VALUE

`masmooth` returns the resulting smoothed array `z`.

`macorr`	Calculate MA Autocorrelation Function	`macorr`

```
macorr(beta,rvar=1,m)
```

ARGUMENTS

`beta` Array containing MA coefficients $\boldsymbol{\beta}$.
`rvar` Real scalar containing error variance (> 0).
`m` Integer containing the number of autocorrelations to calculate (> 0).

VALUE

`macorr` returns a list containing the following four items:
`var` Real scalar containing the variance of the process.
`corr` Array of length `m` containing the autocorrelations.
`ier` Integer variable indicating whether or not the MA process is stationary (0 means yes, anything else means no).

`masp`	Calculate MA Spectral Density Function	`masp`

```
masp(beta,rvar=1,Q=256)
```

ARGUMENTS
`beta` Array of length q containing coefficients $\boldsymbol{\beta}$.
`rvar` Real scalar containing error variance $\sigma^2 (> 0)$.
`Q` Integer ($> q$) containing the number of frequencies between 0 and 1 at which to calculate the spectral density.

VALUE

`masp` returns the array `f` of length $m = [\mathtt{Q}/2] + 1$ containing the MA spectra at frequencies $(j-1)/\mathtt{Q}, j = 1\ldots, m$.

`movave`	Apply Moving Average Smoother to a Vector (Fortran Version)	`movave`

```
movave(x,k)
```

ARGUMENTS
`x` Array containing the data.
`k` Length of the moving average smoother.

VALUE

`movave` returns the resulting smoothed array `z`.

`movbox`	Form Quantities Needed for Moving Box Plot	`movbox`

```
movbox(x,k)
```

ARGUMENTS
`x` Array of length n containing the data.
`k` Integer indicating the length of the moving average smoother.

VALUE

movbox returns a list containing the following three objects:

- summ: Matrix ($n \times 5$) with columns defined as follows:
  Column 1 is largest value in $[u4, u4 + 1.5 * IQR]$, or $u4$ if none,
  Column 2 is the upper fourth (median of largest (n+1)/2),
  Column 3 is the median,
  Column 4 is lower fourth (median of smallest (n+1)/2),
  Column 5 is smallest value in $[l4, l4 - 1, 5 * IQR]$, or $l4$ if none,
  where IQR is the interquartile range.
- inds: Integer array of indices.
- outs: Real array containing values corresponding to inds that fall outside $[l4 - 1.5 * IQR, u4 + 1.5 * IQR]$.

movord — Apply Moving Order Statistic Operator to a Vector — movord

```
movord(x,nord,k)
```

ARGUMENTS

- x: Array of length n containing the data.
- nord: Order of moving order statistic desired.
- k: Length of moving average smoother.

VALUE

movord returns the array of moving order statistics.

multpoly — Multiply Two Polynomials — multpoly

```
multpoly(alpha,beta)
```

ARGUMENTS

- alpha: Array of length p containing the coefficients of the first polynomial.
- beta: Array of length q containing the coefficients of the second polynomial.

VALUE

multpoly returns an array of length $p+q$ containing the coefficients of the product of the polynomials.

odot — Form Outer Product of Two Vectors — odot

```
odot(x,y)
```

ARGUMENTS

- x: A vector.

`y` A vector.

VALUE

`odot` returns an $n \times n$ matrix that is the outer product of the vectors `x` and `y`.

`pacf`	Calculate Sample Partial Autocorrelation Function	`pacf`

`pacf(x,m)`

ARGUMENTS

`x` Array of length n containing the data.

`m` Integer ($0 < m < n$) containing the number of partial autocorrelations to find.

VALUE

`pacf` returns an array of length `m` containing the partial autocorrelations.

`partar`	Calculate AR Coefficients from Autocorrelations	`partar`

`partar(theta)`

ARGUMENTS

`theta` Array of length p containing partial autocorrelations.

VALUE

`partar` returns an array of length p containing AR coefficients.

`perdgm`	Calculate Periodogram of a Time Series	`perdgm`

`perdgm(x)`

ARGUMENTS

`x` Array of length n containing the data.

VALUE

`perdgm` returns the periodogram of `x` at the frequencies $j/n, j = 0, 1, \ldots, [n/2]$.

`plotsp`	Form Plot of a (True or Sample) Spectral Density	`plotsp`

`plotsp(f,n,div,main=‘‘Log Std Spectra’’)`

ARGUMENTS

f Array of length $m = [\mathtt{n}/2] + 1$ containing some spectral quantity the frequencies $(j-1)/\mathtt{n}, j = 1, \ldots, m$.

n Integer containing the length of the array f.

div Real scalar containing divisor of the array f.

main Main title of the resulting plot.

VALUE

plotsp produces a plot of $\log(\mathtt{f}(j)/\mathtt{div})$ versus $(j-1)/\mathtt{n}$ for $j = 1, \ldots, [\mathtt{n}/2] + 1$ with vertical scale running from -6 to 6.

poly	Evaluate a Polynomial at a Vector of Points	poly

```
poly(coeffs,x)
```

ARGUMENTS

coeffs Array of length $p + 1$ containing the coefficients of the polynomial of degree p, (coeffs(i) is the coefficient of power $i - 1$).

x Array containing the points at which to evaluate the polynomial.

VALUE

poly returns an array containing the values of the polynomial.

polyrt	Find the Roots of a Polynomial Given its Coefficients	polyrt

```
polyrt(coeffs,m=100,eps=1.e-6)
```

ARGUMENTS

coeffs Array of length $p + 1$ containing the coefficients of the polynomial of degree p, (coeffs(i) is the coefficient of power $i - 1$).

m Integer containing the maximum number of iterations in the procedure.

eps Real scalar indicating the convergence criterion.

VALUE

polyrt returns a list containing the following items:

real Array of length p containing the real parts of the roots.

imag Array of length p containing the imaginary parts of the roots.

ier Integer variable indicating whether or not the procedure converged (0 means yes, 1 means no).

rtpoly	Find the Coefficients of a Polynomial Given its Roots	rtpoly

```
rtpoly(roots)
```

ARGUMENTS

`roots` Array of length p containing the zeros of the polynomial.

VALUE

`rtpoly` returns an array of length p containing the coefficients of the polynomial with the given roots.

`rw`	Simulate Data from a Random Walk Process	`rw`

`rw(n,seed=0)`

ARGUMENTS

`n` Integer containing the length of the desired realization.

`seed` Real scalar containing the seed for the random number generator.

VALUE

`rw` returns an array of length `n` containing a realization of a Gaussian random walk.

`schur`	Form the Schur Matrix Corresponding to AR Parameters	`schur`

`schur(alpha)`

ARGUMENTS

`alpha` An array of length p containing the AR coefficients $\boldsymbol{\alpha}$.

VALUE

`schur` returns the Schur matrix for the AR coefficients.

`seasest`	Calculate Box-Jenkins Estimates for a Seasonal ARIMA Model	`seasest`

`seasest(y,ords,coeffs,lags,back,maxit=50,eps=0.000001)`

ARGUMENTS

`y` Array of length n containing the data.

`ords` An array of length 5 containing the full and subset AR orders, followed by the full and subset MA orders, followed by a 1 if a constant term is in the model or a 0 if it is not.

`coeffs` An array containing starting values for the coefficients that are included in the model in the order full AR, subset AR, full MA, subset MA and the mean of `y`.

`lags` An array containing the lags (if any) in the model. If both the subset AR and MA orders are zero, no array called `lags` need be formed, but an argument must be included.

back An integer containing the number of back forecasts to used in determining initial values in the recursion used in evaluating the sum of squares of residuals functions (≥ 0).

maxit An integer containing the number of iterations to allow in the estimation procedure. If `maxit` is negative, then –`maxit` iterations are allowed and the values of the coefficients for the successive iterations are displayed on the screen. If `maxit` is 1 then `SEASEST` only evaluates `rvar` and `sds`.

eps Real scalar containing a convergence criterion. If the maximum value of successive iterates differs by less than `eps`, then `SEASEST` judges that the algorithm has converged.

VALUE

`seasest` returns a list containing the following five elements:

coeffs Array containing the final values reached for the parameters in the iterative process. `coeffs` is not changed from input if `maxit`= 1.

e Array of length n containing the one step ahead prediction errors corresponding to the n values of `x`.

ier An integer variable indicating whether or not convergence was achieved (0 means yes, 1 means no), if a singular matrix was encountered (2), or whether the algorithm could not continue even though convergence was no reached (3 or 4). If this final alternative happens, different starting values or convergence criteria may lead to convergence.

rv Real scalar containing an estimate of the error variance.

se An array containing the standard errors of the estimates.

seaspred	Calculate Box-Jenkins Forecasts	seaspred

```
seaspred(x,ords,coeffs,lags,rvar,tf,tl,hl,conf)
```

ARGUMENTS

x Array of length n containing the data.

ords An array of length 8 containing the full and subset AR orders, followed by the full and subset MA orders, followed by a 1 if a constant term is in the model or a 0 if it is not, followed by the number of first differences in the model, the number of Sth differences in the model, and finally the value of S.

coeffs Values for full AR, subset AR, full MA and subset MA coefficients, followed by the constant if there is one and the values of m and λ for the power transform.

lags Array containing the lags (if any) in the model. If both the subset AR and MA orders are zero, no array called `lags` need be formed, but an argument must be included.

rvar Real scalar containing the value of noise variance.

tf,tl Integers containing the prediction origins to use. The values must be at least maxp+maxq+1 (maxp and maxq are the largest AR and MA lags in the expanded version of the model) and at most n, and `tf` must be less than or equal to `tl`.

hl Integer containing the maximum number of steps ahead to forecast from each origin.

conf Real scalar containing the confidence level for the probability limits to be placed on the forecasts ($0 <$ `conf` < 1).

VALUE

`seaspred` returns a list containing the following six items:

xp Array of length ($\texttt{tl} - \texttt{tf} + 1$)`hl` containing the forecasts. The first `hl` elements are from origin `tf`, the next `hl` are from origin `tf+1`, etc.

xpl Array containing the lower probability limits on the corresponding elements of `xp`.

xpu Array containing the upper probability limits on the corresponding elements of `xp`.

ier Integer variable indicating whether `SEASPRED` finished without error (0), or an illegal power transform was requested (1).

check ?*?*?*?*?*?.

xx ?*?*?*?*?*?.

simcfs	Simulate Coefficients of a Polynomial with Zeros Outside Unit Circle	simcfs

```
simcfs(p,seed=0)
```

ARGUMENTS

p Degree of desired polynomial.

seed Real scalar containing the seed for the random number generator.

VALUE

`simcfs` returns an array containing coefficients of a polynomial having all of its zeros greater than one in modulus.

stdf	Standardize a Vector	stdf

```
stdf(f,fac,a,b)
```

ARGUMENTS

f A vector.

fac A nonzero real scalar to divide `f` by.

a Real scalar indicating lower limit allowed for standardized `f/fac`.

b Real scalar indicating upper limit allowed for standardized `f/fac`.

VALUE

`stdf` returns the standardized vector `f`.

swp	Sweep a Matrix	swp

`swp(a,k1,k2)`

ARGUMENTS

- `a` The matrix to be swept.
- `k1,k2` `a` is swept on the diagonals `k1` through `k2`.

VALUE

`swp` returns a list containing the following two items:

- `A` Matrix that results from sweeping `a`.
- `ier` Integer variable containing an indicator of whether or not a zero diagonal was encountered during the sweeping (0 means no, 1 means yes).

toepl	Form Symmetric Toeplitz Matrix Given its First Row	toepl

`toepl(R,R0,M)`

ARGUMENTS

- `R` Array of length $n-1$ containing the second through `n`th elements of the first row of the Toeplitz matrix.
- `R0` Real scalar containing the value for the diagonal of the Toeplitz matrix.
- `M` Size of the resulting matrix.

VALUE

- `G` The desired Toeplitz matrix.

tsvar	Calculate Sample Variance of a Time Series	tsvar

`tsvar(x)`

ARGUMENTS

- `x` A time series.

VALUE

`tsvar` returns the sample variance of the time series `x`.

windowf	Calculate Nonparamteric Spectral Density Estimate	windowf

`windowf(rho,R0,Q,ioptw,M,n,alpha=0.05)`

ARGUMENTS

rho Array of length M (if ioptw is between 1 and 5) or length $\mathtt{n}-1$ if ioptw is between 6 and 8 containing autocorrelations.

R0 Real scalar containing the sample variance (> 0).

Q Integer containing the number of frequencies between 0 and 1 at which to calculate spectra.

ioptw Integer containing the number of the window to be used in the estimation procedure as indicated by the following:

1 Truncated periodogram
2 Bartlett
3 Tukey
4 Parzen
5 Bohman
6 Daniell
7 Bartlett–Priestley
8 Parzen–Cogburn–Davis

M Integer (> 0) containing scale parameter.

n (If either ioptw is between 6 and 8 or the factor for determining confidence intervals is desired.) Integer containing the length of the data set being analyzed.

alpha Real scalar ($0 <$alpha< 1) indicating the level of confidence.

VALUE

windowf returns a list containing the following two items:

f Array of length $[\mathtt{Q}/2] + 1$ containing the spectral estimator at the frequencies $(j-1)/\mathtt{Q}, j = 1, \ldots, [\mathtt{Q}/2] + 1$.

c Real scalar variable that can be used to find 95% confidence intervals for the true spectral density. The interval at the ith frequency would be from f(i)/c to f(i)*c.

wn	Simulate White Noise Data	wn

```
wn(seed,n,dist=1)
```

ARGUMENTS

seed Real scalar containing the seed for the random number generator.

n Integer containing the length of the desired realization.

dist Integer containing the number of the distribution to use based on the following values:

1 N(0,1)
2 U(0,1)
3 Unit exponential
4 Logistic
5 Standard Cauchy
6 Extreme value
7 Lognormal
8 Double exponential

VALUE

`wn` returns a realization of white noise from the specified distribution.

wntest	Form Plots for White Noise Test	wntest

```
wntest(x,m,alpha=0.05)
```

ARGUMENTS

- `x` An array containing the realization to be tested.
- `m` Integer containing the number of correlations.
- `alpha` Real scalar ($0 <$`alpha`< 1) indicating the level of confidence at which the test is to be preformed.

VALUE

`wntest` returns plots of two white noise test for time series data.

4. Tables of S Functions and Fortran Subprograms

Cross-References Between TIMESLAB Commands and S Functions

TIMESLAB	S	Comment
abortoff	options(error=NULL)	See section 6.4.3 of *The New S Language*
aborton	NA	See section 6.4.3 of *The New S Language*
abs	abs*	
arcorr	arcorr**	
arcorr2	none	
ardt	ardt**, diffeq**	diffeq is provided for difference equations.
arfilt	arfilt**	
armacorr	armacorr**	
armadt	armadt**	
armapred	armapred**	
armasel	none	
armasp	armasp***	
arpart	arpart**	
arsp	arsp***	
arsp2	none	
arspcb	none	
arsppeak	arsppeak**	
barttest	none	
batchoff	NA	See section 3.4.8 of *The New S Language*
batchon	S BATCH infile outfile	See section 3.4.8 of *The New S Langauge*
binom	none	
clean	rm*	
cls	!clear	Issue the Unix command clear.
coeffcsd	coeffcsd**	
color	NA	
corr	acf***, acf1*** perdgm***, tsvar***	
corr2	none	

corrar	corrar**	
corrar2	none	
corrarma	corrarma**	
corrma	corrma**	
cos	cos*	
crossp	none	
cum	cumsum*	
cumsp	cumsum*	May be done by using cumsum on spectral array
delay	delay***	
density	density*	
diff	diff*	
dist		For example see Gamma, page 459, *The New S Language*
divsds	none	
dos	unix*	
dot	dot***	
double	none	
dtar	dtar**	
dtarma	dtarma**	
dtfore	none	
echo	NA	
edit	NA	
eig	eigen	
end		See sections 6.2.2 and 11.2.4 of *The New S Language*
endif		See sections 6.2.2 and 11.2.4 of *The New S Language*
erase	none	
exp	exp*	
extend	none	
extract	Subscript	See page 598 of *The New S Language*
fft	fft*	
filt	filt**	
goto	NA	
grmenu	NA	
gs	gr	Impliments Householder successive reflection procedure.

help	help*	
hist	hist*	
if	if, ifelse	
info	NA	
infqnt	infqnt***	
invpoly	divpoly**	
label		See Chapters 4 and 10 of *The New S Language*
length	length*	
line	seq	$y < -m * \texttt{seq} + b$, m is the slope, b is the intercept
list	list	
listm	list	
listsp	perdgm***	
loge	log*	
macorr	macorr**	
macro	NA	
madt	madt**	
masp	masp***	
maxmin	max, min	The S commands are seperate commands.
mchol	chol	
mdel	Subscript	See page 598 of *The New S Language*
minv	solve	
mmult	%*%	In S matrix multiplication is an operation.
multpoly	multpoly**	
notes	NA	
overoff	none	
overon	NA	
page		See Chapters 4 and 10 of *The New S Language*
parcorr	pacf**	
partar	none	
pause	NA	
plot	plot	
plot2		See Chapters 4 and 10 of *The New S Language*

plotcsp	plot	Used in combination with cumsum command
plotk		See Chapters 4 and 10 of *The New S Language*
plotoff	NA	
ploton	NA	
plotsize	NA	
plotsp	plotsp	
polar	none	See page 421 of *The New S Language*
poly	poly**	
polyroots	polyrt**	
print	NA	
printer		See Section 4.1 *The New S Language*
printsel		See Chapters 4 and 10 of *The New S Language*
promptoff	NA	
prompton	browser	
psoff	NA	
pson	none	
qtest	none	
quit	q()	
read	scan	
record	sink*	
reg	lsfit	
replace	Subscript	See page 598 of *The New S Language*
rescreen	gr.display	
restart	NA	
reverse	rev*	
rootspoly	rtpoly**	
save	dput *, write*	
savesc		See section 10.2.2 of *The New S Language*
seasest	seasest**	
seaspred	seaspred**	

sin	sin*	
singleoff	NA	
singleon	none	
sort	sort*	
speaker	NA	
speakeroff	NA	
speakeron	NA	
submns	sabl	See page 574 of *The New S Language*
sweep	swp**	The S function sweep does something else.
textcolor	NA	
time	proc.time, unix.time	See Section 7.3.5 of *The New S Language*
toepl	toepl***	
trans	t*	
type2	NA	
type4	rbind, cbind	
while	while	See sections 6.2.2 and 11.2.4 of *The New S Language*
window	windowf***	The S function window does something else.
wn	wn***	

* New S function

** S function written by Newton, Uses Fortran

*** S function written by Newton, Doesn't Use Fortran

Notes

1.

Other S Functions

Name	Purpose
band	S version of BAND macro
chiplot	S version of CHIPLOT macro
fplot	S version of FPLOT macro
arma	Illustrate ARMA processes
armapart	Calcuate ARMA partial autocorrelation functions
clip	Clip a vector above and below
corrdt	Simulate data having specified autocorrelations
crlag	Apply circular shift operator
freqs	Form a vector of natural frequencies
masmooth	S version of SMOOTH macro
movave	Fortran version of SMOOTH macro
movbox	Form quantities needed for moving box plot
movord	Apply moving order statistics operator to a vector
odot	Form outer product of two vectors
rw	S version of RW macro
schur	Form Schur matrix corresponding to AR parameters
simcfs	S version of RANDCOEF macro
stdf	Standardize a vector
wntest	S version of WNTEST macro

Fortran Subprograms

Name	File	Purpose
arfilt	arfilt.f	Apply AR filter
arpart	arpart.f	Find partials from AR coefficients
arsppk	arsppk.f	Find peak frequency of AR process and its standard error
corrar	corrar.f	Find AR parameters from correlations
cvmx1	crarma.f	Find ARMA parameters from covariances
diffeq	diffeq.f	Perform a difference equation
dtarma	dtarma.f	Find ARMA MLE's
filt	filt.f	Filter an array
marq	seasest.f	Calculate Box-Jenkins estimates for Seasonal ARIMA
median	median.f	Calculating medians
mmult	mmult.f	Multiply an (nxm) matrix times an (mxk) matrix
movave	movord.f	Find moving averages
mxcsd	coeffsd.f	Find standard deviation of estimated coefficients of ARMA process
mxpd	armapred.f	Find ARMA predictors
pacf	arsp.f	Find AR spectral density
pacf	pacf.f	Find sample partial autocorrelations
partar	partar.f	Find AR coefficients from partial autocorrelations
poly	poly.f	Evaluate a polynomial at given values
rtpoly	roots1.f	Find coefficients of a polynomial given its roots
schur	schur.f	Form Schur matrix for AR coefficients
sspr	seaspred.f	Calculate Box-Jenkins forecasts
swpk12	swp.f	Sweep (nxn) matrix on diagonals k1 through k2
wilson	wilson.f	Find mx covariances