Package ‘tframe’

May 11, 2007

Title Time Frame coding kernel

Description Functions for writing code that is independent of the way time is represented. See ?tframe.Intro for more details.

Depends R (>= 2.5.0)

Imports stats

Version 2007.5-2

Date 2007-05-11

LazyLoad yes

License Free. See the LICENCE file for details.

Author Paul Gilbert <pgilbert@bank-banque-canada.ca>

Maintainer Paul Gilbert <pgilbert@bank-banque-canada.ca>

URL http://www.bank-banque-canada.ca/pgilbert

R topics documented:

`00tframe.Intro` *Tframe - Generic Approach to Handling Time*

Description

Programs for implementing an object oriented approach to handling different time representations.

Details

The **tframe** package provides a kernel of functions for programming time series methods in a way that makes them relatively independently of the representation of time. pkgtframe is intended to make it easier to write code which can use any new/better time representations when they appear. It also provides plotting, time windowing, and some other utility functions which are specifically intended for time series. The main functions in this package that might be called directly by users are `tfplot`, `diffLog`, `percentChange`, `trimNA` and `tsScan`. See the help for more details and examples.

tframe provides generic methods by which code can be developed without too much dependence on the representation of time (i.e. specific time series objects). This can make most code very robust with respect to other (and future) improved/different representations of time. However, details like putting the time axis label on a plot may require a method for the the specific time representation.

This package does not try to replace classes and methods associated with time representations such as `ts`, `zoo`, and `its`. Rather, it attempts to provide generic programming "wrappers" so that other programs do not need to look after the details of these different representations. For many time series programs the availability of a *window* method provided by those classes is the main method which is needed. However, the time attributes of calculated objects are often lost and programmers must re-assign time attributes to the resulting object if they are to be retained. Historically this was done with `tsp`, but this relies on a particular time representation and would not work for other representations. In order to address this, the **tframe** methods attempt to separate the time representation from the data and allow a statement

```
tframe(x) <- tframe(y)
```

to make the time frame of `x` the same as that of `y`, without the need to worry about what time representation is used in `y`. In this assignment `x` and `y` need not be too similar (one might be a univariate series while the other is a matrix or an array or list of spatial or panel time series data), as long as they are similar in the time dimension. For the case where `tsp(x) <- tsp(y)` would make sense, that is effectively what the above tframe assignment will do. For some existing code, most of the conversion to these more robust methods is accomplished simply by changing "tsp" to "tframe" and nrow() for a time series matrix to periods().

The tframe assignment example above is accomplished by switching the dispatch so that it follows the classes of the tframe of `y`, rather than the classes of `x`, as would normally be done for the above kind of assignment. Doing this in a generic way allows for the possibility of future classes of time representation. This is different from the way that zoo, its and ts are implemented, in the sense that it is the tframe of the data which is assigned a class indicating the time representation, not the data object itself.

The most general (last) class of the tframe should be `tframe`. The method `is.tframe` checks if an object is a tframe, and the method `is.tframed` checks if an object has a tframe. In general, tframe methods act on the time frame (tframe) and tframed methods act on data which is tframed.

More specific methods can be defined for any special time representation. Methods are defined in this package for ts, zoo, and its, and defaults work for old style tsp, and may also work in sme other cases. The tframe's specific classes are called tstframe, zootframe, and itstframe, to prevent confusion using inherit().

The main programing utilities are `tframe` and `tframe<-`. For additinal details see the help for these and `tframe-package`.

The method tfwindow is used in this library and is typically just the same as window, but the new name has been used because of historical changes and bugs in window, and in order to support the argument "warn" to suppress messages (when objects are windowed unnecessarily).

One implication of a statement like tframe(x) <- tframe(y) is that the tframe should not indicate which dim of the data is the time dimension. In general this will have to be another attribute of the data. For older representations, the convention of using the first dimension for matrix data and the length for vector data, makes it unnecessary to specify.

The attribute "seriesNames" is also supported as a way to indicate the names of series in an object. This overlaps with the use of "names" and "dimnames[[2]]" used previously for series names in S, but seems necessary in order to have a more complete generic decompostion of the time dimension from the other dimensions.

Many of the functions in the library are not yet individually documented, however, the functions are all very short and can be examined easily. The code in the tests subdirectory provides a short set of tests and may serve as an example.

To implementing a new time representation, suppose it is called zzz, then the tframe attibute of an object using this time frame should have class c("zzz", "tframe"). (Note zzz should be different from the class of the object itself.) The most important methods which need to be implemented are tframe.zzz(), start.zzz(), end.zzz(), and periods.zzz(). While frequency.zzz() should not in theory be necessary, it makes porting code much easier. Other methods which may be needed are time.zzz(), checktframeConsistent.zzz(), tfTruncate.zzz(), tfExpand.zzz(), earliestStartIndex.zzz(), earliestEndIndex.zzz(), latestStartIndex.zzz(), and latestEndIndex.zzz().

Package:	tframe
Depends:	R
License:	free, see LICENSE file for details.
URL:	http://www.bank-banque-canada.ca/pgilbert

Author(s)

Paul Gilbert <pgilbert@bank-banque-canada.ca>

Maintainer: Paul Gilbert <pgilbert@bank-banque-canada.ca>

See Also

`tframe`, `tframed`, `tfwindow`, `tfplot`

addDate *Add Periods to a Date*

Description

Add periods to two element start date of given frequency to give a new date. NULL periods is treated as 0.

Usage

```
addDate(date, periods, freq)
```

Arguments

`date`	A two element date as used by tsp i.e c(year, period).
`periods`	A number of periods.
`freq`	The number of periods in a year.

Value

A two element date.

Note

A useful utility not strictly part of tframe.

See Also

`tfExpand`

Examples

```
addDate(c(1998,1), 20, 12)
```

checktframeConsistent *Check for a Consistent tframe*

Description

Check if tframe and a time series are consistent with one another.

Usage

```
checktframeConsistent(tf, x)
## Default S3 method:
checktframeConsistent(tf, x)
```

Arguments

`tf`	A tframe)
`x`	An object)

Details

Check if the number of periods in the tframe corresponds to the number of observations in the time series.

Value

A logical scalar.

See Also

`is.tframe` `periods`

Examples

```
z <- ts(rnorm(100), start=c(1982,1), frequency=12)
checktframeConsistent(tframe(z), rnorm(100))
```

`classed` *Tframe Library Utilities*

Description

Some utilities used by other functions in the tframe library.

Usage

```
classed(x, cls)
```

Arguments

`x`	An object.
`cls`	A vector of strings.

diffLog — *Calculate the difference of log data*

Description

Calculate the difference from lag periods prior for log of data.

Usage

```
diffLog(obj, lag=1, base = exp(1),
     names=paste("diff of log of ", seriesNames(obj)))
## Default S3 method:
diffLog(obj, lag=1, base = exp(1),
     names=paste("diff of log of ", seriesNames(obj)))
```

Arguments

`obj`	A time series object.
`lag`	The difference is calculated relative to lag periods prior.
`base`	Base to use when calculating logrithms.
`names`	names for the new series (but is details).

Details

The result is a time series of the difference relative to lag periods prior for the log of the data. lag data points are lost from the beginning of the series. Negative values will result in NAs.

The `names` are not applied to the new series if the global option ModSeriesNames is FALSE. This can be set with `options(ModSeriesNames=FALSE)`. This provides a convenient mechanism to prevent changing series labels on plot axis, when the title may indicate that data is in year-to-year percent change so the axis label does not need this.

Value

A time series vector or matrix.

Examples

```
z <- matrix(100 + rnorm(200),100,2)
z[z <= 0] <- 1 # not to likely, but it can happen
z <- diffLog(z)
```

earliestEnd *Start and End for Objects with Multiple Time Series*

Description

Return start or end date (or index of the object) from multiple time series objects.

Usage

```
earliestEnd(x, ...)
earliestEndIndex(x, ...)
## Default S3 method:
earliestEndIndex(x, ...)
## S3 method for class 'tframe':
earliestEndIndex(x, ...)

earliestStart(x, ...)
earliestStartIndex(x, ...)
## Default S3 method:
earliestStartIndex(x, ...)
## S3 method for class 'tframe':
earliestStartIndex(x, ...)

latestEnd(x, ...)
latestEndIndex(x, ...)
## Default S3 method:
latestEndIndex(x, ...)
## S3 method for class 'tframe':
latestEndIndex(x, ...)

latestStart(x, ...)
latestStartIndex(x, ...)
## Default S3 method:
latestStartIndex(x, ...)
## S3 method for class 'tframe':
latestStartIndex(x, ...)
```

Arguments

`x`	A tframe or tframed object.
`...`	Additional tframe or tframed objects.

Details

These functions calculate the start and end of each object in the argument and return a result by comparing across objects. Thus, latestStart returns the start date of the object which starts latest and latestStartIndex returns the corresponding index of the object in the argument list.

Value

A date or index.

See Also

tframe tfwindow tfTruncate trimNA

Examples

```
t1<-ts(c(1,2,3,4,5), start=c(1991,1))
t2<-ts(c(2,3,4,5,6,7,8), start=c(1992,1))
t3<-ts(c(NA,2,3,4,5), start=c(1991,1))

latestStart(t1,t2,t3)  # 1992 1 corresponding to the starting date of
                        # the object which starts latest (t2)
latestStart(t1,t3)     # both start in 1991 1 (NAs count as data)
latestStart(tbind(t1,t2,t3)) # tbind gives a single object starting in 1991 1
latestStart(t2, tbind(t1,t2,t3))

latestStartIndex(t1,t2,t3)  # position of t2 in the argument list
```

---

nseries *Number of Series*

Description

Return the number of series.

Usage

```
nseries(x)
## Default S3 method:
nseries(x)
```

Arguments

x	A time series object.

Details

Generic method to return the number of series.

Value

An integer.

Examples

```
nseries(tbind(rnorm(100,20,5)))
```

`percentChange` *Various Time Series Calculations*

Description

Calculate various conversions of time series.

Usage

```
percentChange(obj, ...)
## Default S3 method:
percentChange(obj, base=NULL, lag=1, cumulate=FALSE, e=FALSE, ...)

ytoypc(obj, names = paste("y to y %ch", seriesNames(obj)))
## Default S3 method:
ytoypc(obj, names = paste("y to y %ch", seriesNames(obj)))

annualizedGrowth(obj, ...)
## Default S3 method:
annualizedGrowth(obj, lag=1, freqLagRatio=frequency(obj)/lag,
    names=paste("Annual Growth of", seriesNames(obj)), ...)
```

Arguments

`obj`	An object on which the calculation is to be done. The default method works for a time series vector or matrix (with columns corresponding to series, which are treated individually).
`e`	If e is TRUE the exponent of the series is used (after cumulating if cumulate is TRUE). e can be a logical vector with elements corresponding to columns of obj.
`base`	If base is provided it is treated as the first period value (that is, prior to differencing). It is prefixed to the m prior to cumulating. It should be a vector of length dim(m)[2]. (If e is TRUE then base should be log of the original data).
`lag`	integer indicating the number of periods relative to which the change should be calculated.
`cumulate`	logical indicating if the series should be cumulated before the percent change is calculated.
`freqLagRatio`	the ratio of `obj`'s frequency to the number of lags.
`names`	gives new names to be given to the calculated series.
`...`	arguments passed to other methods.

Details

`percentChange` calculate the percent change relative to the data lag periods prior. If `cumulate` is TRUE then the data is cumulated first. `cumulate` can be a logical vector with elements corresponding to columns of obj.

The result is a time series of the year over year percent change. This uses percentChange with lag=frequency(obj).

The `names` are not applied to the new series if the global option ModSeriesNames is FALSE. This can be set with `options(ModSeriesNames=FALSE)`. This provides a convenient mechanism to prevent changing series labels on plot axis, when the title may indicate that data is in year-to-year percent change so the axis label does not need this.

`annualizedGrowth` calculates the year to year percentage growth rate using `100*((obj/shift(obj, periods= -lag))^freqLagRatio - 1)`. The default gives the annualized one period growth. If `lag` is equal to the frequency of `obj` then the result is year-over-year growth.

Value

A time series or time series matrix.

See Also

`diff`

Examples

```
z <- matrix(100 + rnorm(200),100,2)
z[z == 0] <- 1 # not to likely, but it can happen
zyypc <- ytoypc(z)
zpc <- percentChange(z)
zag <- annualizedGrowth(z)
```

`selectSeries` *Extract a Subset of Series*

Description

Extract a subset of series from a tframed object.

Usage

```
    selectSeries(x, series = seqN(nseries(x)))
    ## Default S3 method:
    selectSeries(x, series = seqN(nseries(x)))
    ## S3 method for class 'ts':
    selectSeries(x, series = seqN(nseries(x)))
```

Arguments

`x`	A tframed object.
`series`	The subset of series to retain.

Details

This is like [, , drop=FALSE] but retains class, series name and tframe information. It also provides a methods which works with multivariate series which are not matrices (e.g. tfPADIdata).

Value

A tframed object.

See Also

`seriesNames`

Examples

```
z <- selectSeries(matrix(rnorm(1000), 100,10), series=c(2, 5, 6))
```

`seqN` *Tframe Library Utilities*

Description

Some utilities used by other functions in the tframe library.

Usage

```
seqN(N)
```

Arguments

`N`	NULL, 0, or a positive integer

`seriesNames` *Names of Series in a time series object*

Description

Extract or set names of series in a time series object.

Usage

```
seriesNames(x)
## Default S3 method:
seriesNames(x)

seriesNames(x) <- value
## S3 replacement method for class 'default':
seriesNames(x) <- value
## S3 replacement method for class 'ts':
seriesNames(x) <- value
```

Arguments

`x`	a time series object.
`value`	names to be given to time series.

Value

The first usage returns a vector of strings with the series names. The assignment method makes `names` (a vector of strings) the series names of data.

See Also

`tframed`, `seriesNamesInput`, `seriesNamesOutput`

Examples

```
z <- matrix(rnorm(100), 50,2)
seriesNames(z) <- c("a", "b")
seriesNames(z)
```

`splice` *Splice Time Series*

Description

Splice together (in time dimension) two time series objects. This function can also be used to overlay obj1 on obj2 (obj1 takes precedence). The time windows do not have to correspond.

Usage

```
splice(mat1,mat2, ...)
## Default S3 method:
splice(mat1,mat2, ...)
```

Arguments

`mat1`	A time series object.
`mat2`	A time series object.
`...`	arguments to be passed to other methods (not used by the default method).

Details

Splice together two time series objects. The mat1 and mat2 objects should contain the same number of time series variables and be arranged in the same order. (e.g. - the first column of mat1 is spliced to the first column of mat2, etc.). If data is provided in both mat1 and mat2 for a given period then mat1 takes priority. The frequencies should be the same.

Value

A time series object

See Also

`tfwindow`, `trimNA`, `tbind`

Examples

```
splice(ts(matrix(rnorm(24),24,1), start=c(1980,1), frequency=4),
       ts(matrix(rnorm(6),  6,1), start=c(1986,1), frequency=4))
```

---

tbind *Bind Time Series*

Description

Bind together (in non-time dimension) two time series objects.

Usage

```
    tbind(x, ..., pad.start=TRUE, pad.end=TRUE, warn=TRUE)
    ## Default S3 method:
    tbind(x, ..., pad.start=TRUE, pad.end=TRUE, warn=TRUE)
    ## S3 method for class 'ts':
    tbind(x, ..., pad.start=TRUE, pad.end=TRUE, warn=TRUE)
```

Arguments

`x`	A time series object.
`...`	Time series objects.
`pad.start`	Logical indicating if the start should be truncated or padded with NAs to align time.
`pad.end`	Logical indicating if the end should be truncated or padded with NAs to align time.
`warn`	Logical indicating if warnings should be issued.

Details

Bind data as in cbind (or formerly tsmatrix) and align time dimension. The default action pads series with NA to time union. If pad.start and/or pad.end is FALSE and the intersection is empty then NULL is returned and a warning is issued if warn=TRUE.

Value

A time series object

See Also

`tfwindow`, `trimNA`, `splice`

Examples

```
tbind(    ts(matrix(rnorm(24),24,1), start=c(1986,1), frequency=4),
    ts(matrix(rnorm(6),  6,1), start=c(1986,1), frequency=4))
```

testEqual — *Compare Two Objects*

Description

Generic function to compare two objects. The methods return a logical value, TRUE if the objects are the same type and value and FALSE otherwise. The default compares array values but not attributes or class. Some descriptive information in the objects may be ignored.

Usage

```
testEqual(obj1, obj2, fuzz = 0)
## Default S3 method:
testEqual(obj1, obj2, fuzz = 1e-16)
## S3 method for class 'array':
testEqual(obj1, obj2, fuzz = 1e-16)
## S3 method for class 'list':
testEqual(obj1, obj2, fuzz = 1e-16)
## S3 method for class 'matrix':
testEqual(obj1, obj2, fuzz = 1e-16)
## S3 method for class 'numeric':
testEqual(obj1, obj2, fuzz = 1e-16)
```

Arguments

`obj1, obj2`	Objects of the same class.
`fuzz`	Differences less than fuzz are ignored.

Details

The functions for comparing numeric values used in the default method for this generic replacement.

Value

TRUE or FALSE.

See Also

`testEqualtframes`

Examples

```
testEqual(matrix(1:10,10,2), array(1:10, c(10,2)))
testEqual(matrix(1:10,10,1),1:10)
```

testEqualtframes *Compare Two Time Frames*

Description

Generic function to compare two time frames. The methods return a logical value, TRUE if the time frames are the same type and value and FALSE otherwise.

Usage

```
testEqualtframes(tf1,tf2)
## Default S3 method:
testEqualtframes(tf1,tf2)
## S3 method for class 'stamped':
testEqualtframes(tf1,tf2)
```

Arguments

`tf1, tf2`	Time frames of the same class.

Details

Time frames are compared. Time frames need to be of the same class (although it would be nice if they did not need to be).

Value

TRUE or FALSE

See Also

`testEqual`

Examples

```
testEqualtframes(tframe(matrix(1:10,10,2)), tframe(array(1:10, c(10,2))))
```

tfExpand *Expand a Tframe or Tframed Object.*

Description

Expand a tframe or tframed object.

Usage

```
tfExpand(x, add.start = 0, add.end = 0)
## Default S3 method:
tfExpand(x, add.start = 0, add.end = 0)
## S3 method for class 'tframe':
tfExpand(x, add.start = 0, add.end = 0)

tfTruncate(x, start=NULL, end=NULL)
## Default S3 method:
tfTruncate(x, start=NULL, end=NULL)
## S3 method for class 'tframe':
tfTruncate(x, start=NULL, end=NULL)
```

Arguments

`x`	A tframe or tframed object.
`start`	an integer indicating the position at which the new tframe is to start.
`end`	an integer indicating the position at which the new tframe is to end.
`add.start`	an integer indicating the number of periods on the beginning.
`add.end`	an integer indicating the number of periods on the end.

Details

These methods are like tfwindow but use position indicators (rather than dates) and work with a tframe or tframed data. Applied to a tframe they return an adjusted tframe. Applied to a tframed object they return an adjusted object with its adjusted tframe.They are low level utilities for other functions.

Value

A tframe or tframed object.

See Also

`tfwindow` `tframed`

Examples

```
z <- ts(rnorm(100), start=c(1982,1), frequency=12)
Dz <- tframed(diff(z), tfTruncate(tframe(z), start=2))
tframe(Dz)
IDz <- tframed(cumsum(c(0, Dz)), tfExpand(tframe(Dz), add.start=1))
tframe(IDz)
tframe(tfTruncate(z, start=5))
```

`tfL` *Time Series Shifting and Differencing*

Description

Lag, shift forward, or difference a tframe or tframed object.

Usage

```
tfL(x, p=1)
## Default S3 method:
tfL(x, p=1)
## S3 method for class 'tframe':
tfL(x, p=1)

## S3 method for class 'tframe':
diff(x,lag=1, differences=1, ...)
## S3 method for class 'tframed':
diff(x,lag=1, differences=1, ...)
```

Arguments

`x`	a tframed object.
`lag`	difference calculated relative to lag periods previous.
`differences`	order of differencing.
`p`	number of periods to shift or lag periods for differencing.
`...`	arguments to be passed to other methods.

Details

`tfL` methods shift the time frame, or the time frame of the object, by `p` periods. (This might also be thought of as the exponent of the lag operator.) Positive `p` means shift the time frame forward and negative `p` means shift the time frame back. Shifting the time frame forward means the data at a point in time is the data from the previous point in time in the unshifted data, so the result corresponds to what is often called the lagged data. The default `p` (+1) means the start and end for the results are one period later. When applied to a data object, the default result is one lag of the data. This convention is not the same as that used for `k` in the function `lag`.

Note that the time frame of the data is shifted, but a vector or matrix representation of the data is unchanged. This means that operations on the data, such as `+, -, *, and /`, need to be time aware as, for example, operations on ts objects are. If the operations do not recognize the time framed aspect of the objects, then the operation will be performed with default methods that will probably have unintended results (see examples).

Differencing methods create a time frame or time framed object by differencing the number of times indicated by `differences` at a lagged number of periods indicated by `lag`. (Positive values of codelag indicate number of periods back.) The default is take the difference from data one period previous. See `diff` for more details, but note that the result when applied to a time frame is a time frame, not a series.

See Also

diff, lag

Examples

```
z <- ts(rnorm(100), start=c(1982,1), frequency=12)
tfstart(z)
tfperiods(z)
z <- diff(z)
tfstart(z)
tfperiods(z)

ts(1:5) - tfL(ts(1:5))
(1:5) - tfL(1:5) # (1:5) this is not a tframed object, so minus is the default
ts(1:5) - tfL(ts(1:5), p= 2)

z <- ts(1:10, start=c(1992,1), frequency=4)
z - tfL(z)

z <- ts(matrix(1:10,5,2), start=c(1992,1), frequency=4)
seriesNames(z) <- c("One", "Two")
z - tfL(z)
```

tfplot *Plot Tframed Objects*

Description

Plot tframe or tframed objects.

Usage

```
tfplot(x, ...)

## Default S3 method:
tfplot(x, ..., tf=tfspan(x, ...), start=tfstart(tf), end=tfend(tf),
   series=seq(nseries(x)), Title=NULL,
    lty = 1:5, lwd = 1, pch = NULL, col = 1:6, cex = NULL,
   xlab=NULL, ylab=seriesNames(x), xlim = NULL, ylim = NULL,
   graphs.per.page=5, par=NULL, mar=par()$mar, reset.screen=TRUE)
tfOnePlot(x, tf=tframe(x), start=tfstart(tf), end=tfend(tf),
    lty=1:5, lwd=1, pch=NULL, col=1:6, cex=NULL,
    xlab=NULL, ylab=NULL, xlim=NULL, ylim=NULL, ...)
```

Arguments

`x`	a tframe or tframed object to plot.
`...`	any additional tframed objects for the same plot.
`start`	start of plot. (passed to tfwindow)
`end`	end of plot. (passed to tfwindow)
`tf`	a tframe or tframed object which can be used to specify start and end.

`series`	series to be plotted. (passed to selectSeries)
`Title`	string to use for plot title (but see details).
`lty`	passed to matplot. See also par.)
`lwd`	passed to matplot. See also par.)
`pch`	passed to matplot. See also par.)
`col`	passed to matplot. See also par.)
`cex`	passed to matplot. See also par.)
`xlab`	string to use for x label (passed to plot).
`ylab`	string to use for y label (passed to plot).
`xlim`	passed to matplot. See also par.)
`ylim`	passed to matplot. See also par.)
`graphs.per.page`	integer indicating number of graphs to place on a page.
`par`	a list of arguments passed to par() before plotting.)
`mar`	margins passed to plot (deprecated, use par).)
`reset.screen`	logical indicating if the plot window should be cleared before starting. If this is not TRUE then mar values will have no effect.

Details

In many cases these are the same as plot methods, however, tfplot puts different series in the object `x` in different plot panels, whereas `plot` usually puts them in the same panel. For this reason, `tfplot` tends to work better when the scale of the different series are very different. If additional objects are supplied, then they should each have the same number of series as `x` and all corresponding series will be plotted in the same panel.

`tfplot` provides an alternate generic mechanism for plotting time series data. New classes of time series may define there own `tfplot` (and `plot`) methods.

The start and end arguments to tfplot determine the start and end of the plot. The argument tf is an alternate way to specify the start and end. It is ignored if start and end are specified.

If `xlim` and `ylim` are not NULL they should be a vector of two elements giving the max and min, which are applied to all graphs, or a list of length equal to the number of series to be plotted with each list element being the two element vector for the corresponding plot limits.

The `Title` is not put on the plot if the global option PlotTitles is FALSE. This can be set with `options(PlotTitles=FALSE)`. This provides a convenient mechanism to omit all titles when the title may be added separately (e.g. in Latex).

Value

None.

Side Effects

An object is plotted.

See Also

`tfprint` `tframe` `tframed` `print` `plot` `matplot` `par`

Examples

```
tfplot(ts(rnorm(100), start=c(1982,1), frequency=12))
tfplot(ts(rnorm(100), start=c(1982,1), frequency=12), start=c(1985,6))
```

`tfprint` *Print Tframed Objects*

Description

Print tframe or tframed objects.

Usage

```
tfprint(x, ...)
## Default S3 method:
tfprint(x, ...)
## S3 method for class 'tframe':
tfprint(x, ...)
## S3 method for class 'tframe':
print(x, ...)
```

Arguments

`x`	a tframe or tframed object.
`...`	arguments to be passed to other methods.

Details

`tfprint` prints data in a tframed object while `tframePrint` prints the tframe. In many cases these are the same as print methods. However, tfprint tries to provide an alternate generic mechanism that is consistent with the tframe view of the data. This may not always be the preferred print method. Also, new classes of time series may define there own print methods in ways which use a different logic from the tframe library. Thus tfprint provides a way to program functions which use methods consistent with the tframe library logic.

Value

tfprint methods return the object invisibly.

Side Effects

An object is printed.

See Also

`tfplot` `tframe` `tframed` `print` `plot`

Examples

```
tfprint(ts(rnorm(100)))
```

tframe — *Extract or Set a tframe*

Description

Extract or set the tframe of an object.

Usage

```
as.tframe(...)
as.tframed(x)

tframe(x)
## Default S3 method:
tframe(x)
## S3 method for class 'ts':
tframe(x)
## S3 method for class 'zoo':
tframe(x)
## S3 method for class 'its':
tframe(x)

tframe(x) <- value
tfSet(value, x)
## Default S3 method:
tfSet(value, x)
## S3 method for class 'list':
tfSet(value, x)
## S3 method for class 'tstframe':
tfSet(value, x)
## S3 method for class 'zootframe':
tfSet(value, x)
## S3 method for class 'itstframe':
tfSet(value, x)

tframed(x, tf=NULL, names = NULL, ...)
## Default S3 method:
tframed(x, tf = NULL, names = NULL, start=NULL, end=NULL, ...)

is.tframe(x)
is.tframed(x)
```

Arguments

`x`	an object (to which a tframe is assigned in assignment methods).
`value`	a tframe.
`tf`	a tframe object or a tframed object from which a tframe is taken.
`start`	provides simple way to specify a tframed time series similar to a `ts` object.
`end`	provides simple way to specify a tframed time series similar to a `ts` object.

`names`	optional vector of strings to specify new series names.
`...`	arguments passed to default to construct a tframe (rather than extract one from x.) `frequency` might often be used if `start` or `end` are specified.

Details

The usage `tframe(x)` returns the tframe of a tframed object. The assignment `tframe(x)<-`, `tfSet`, and `tframed` set the tframe of an object. `as.tframe(...)` constucts a tframe from .... `is.tframe` and `is.tframed` return logicals if the argument is a tframe or tframed object respectively. `as.tframed` guarantees `x` has a tframe by assigning a default if `x` does not already have a tframe.

The object of these functions is to be able to write code with `tframe(y) <- tframe(x)`, to assign the tframe attributes of `x` to `y`, without needing to handle details of the time representation and without concern for the number of series in `x` and `y`, which need not be the same. A check is made to ensure the number of periods in the data correspond with the number implied by the tframe.

There is an attempt to use the same time representation for `y` as `x` has (e.g. ts, zoo, its), but this cannot be guaranteed because `y` may not be representable using the `x` represnetation. For example, `x` might be an "mts" constructed with `ts()` whereas `y` is a list with some data structures. In this case, a "pure tframe" approach is used.

The pure tframe approach sets a "tframe" attribute in object. This attribute has a class which indicates the time framing which is used. The the time frame information is often secondary, in the sense that it does not describe the object structure, but only provides some additional information which is useful for doing time based operations on the data, plotting, and printing the object. By putting this in an attribute, the objects class can be used for indicating other information about the structure of the object. For these pure tframe objects the default `tframe` and codetframe<- will often be adequate. The generic/method approach allows for special case (like TSdata where the tframe information is not an attribute of the object, but rather an attribute of the data matrices which are elements of the object).

The generic/method approach also allows for (faking) tframe assignment and extraction with classes like zoo, its, ts, and others which may appear, that try to make the time description part of the object class. (Not a "tframe" approach.) The problem is to extract real tframes and also fake these other classes and old style tsp objects so they look like tfamed objects. Another approach would be to mutilate these objects and force them really be tframed objects (to have a tframe attribute), but that risks conflicting with other (non tframe) code which used the objects. This faking is accomplished by specific methods of the classes.

The `tframed` constructor is simply a shortcut for assigning the tframe (tframe(x) <- tf) and series names (seriesNames(x) <- names) to an object, but never assigns NULL values, so the result is guaranteed to be a codetframed object. It is like `ts` but enables the tframe library's methods for handling time. If the `tf` argument is a `tframed` object rather than a `tframe`, then the codetframe is extracted and used. If the `names` argument is not mode "character" of appropriate length, then `seriesNames(names)` is used. These make it simple to assign the time frame and names of one object to another by `z <- tframed(x, tf=y, names=y)`.

`is.tframed` returns TRUE if a `tframe()` can extract a tframe from the object. This is true for many objects which are not truly tframed (like ts objects), since `tframe()` tries fairly hard to build a tframe for the object.

Value

Depends.

See Also

`tfstart`, `tfend`, `tffrequency`, `tfperiods`, `tftime`, `tfL`

Examples

```
z <- tframe(ts(rnorm(100), start=c(1982,1), frequency=12))
z
is.tframe(z)
zz <- tframed(matrix(rnorm(200), 100,2), tf=z)
is.tframed(zz)
zzz <- tframed(matrix(rnorm(200), 100,2), tf=zz)
is.tframed(zzz)
tframe(zzz)

as.tframe(start=c(1992,1), end=c(1996,3), frequency=4)
periods(as.tframe(start=c(1992,1), end=c(1996,3), frequency=4))
end(as.tframe(start=c(1992,1), end=c(1996,3), frequency=4))

z <- tframed(rnorm(100), start=c(1982,1), frequency=12)
```

`tfspan` *Time Span*

Description

Calculate Time Span of Objects.

Usage

```
tfspan(x, ...)
```

Arguments

`x`	a tframe or a tframed object.
`...`	other tframes or tframed objects.

Details

Calculate the time frame from the earliest start to latest end of all arguments.

Value

A tframe

See Also

`tframe`, `tframed` `start` `end` `frequency` `periods` `time`

Examples

```
z  <- ts(rnorm(100), start=c(1982,1), frequency=12)
zz <- ts(rnorm(100), start=c(1992,1), frequency=12)
tfspan(z, zz)
```

`tfstart` *Extract Time Frame Information*

Description

Functions for extracting time frame information.

Usage

```
## S3 method for class 'tframed':
start(x, ...)
## S3 method for class 'tframe':
start(x, ...)
tfstart(x)
## Default S3 method:
tfstart(x)
## S3 method for class 'tstframe':
tfstart(x)
## S3 method for class 'zoo':
tfstart(x)
## S3 method for class 'its':
tfstart(x)
## S3 method for class 'zootframe':
tfstart(x)
## S3 method for class 'itstframe':
tfstart(x)

## S3 method for class 'tframed':
end(x, ...)
## S3 method for class 'tframe':
end(x, ...)
tfend(x)
## Default S3 method:
tfend(x)
## S3 method for class 'tstframe':
tfend(x)
## S3 method for class 'zoo':
tfend(x)
## S3 method for class 'its':
tfend(x)
## S3 method for class 'zootframe':
tfend(x)
## S3 method for class 'itstframe':
tfend(x)

## S3 method for class 'tframed':
frequency(x, ...)
## S3 method for class 'tframe':
frequency(x, ...)
tffrequency(x)
## Default S3 method:
```

```
tffrequency(x)

periods(x)
## Default S3 method:
periods(x)
## S3 method for class 'tframed':
periods(x)
## S3 method for class 'tframe':
periods(x)
tfperiods(x)
## Default S3 method:
tfperiods(x)
## S3 method for class 'stamped':
tfperiods(x)
## S3 method for class 'zoo':
tfperiods(x)

## S3 method for class 'tframed':
time(x, ...)
## S3 method for class 'tframe':
time(x, ...)
## S3 method for class 'its':
time(x, ...)
tftime(x)
## Default S3 method:
tftime(x)
## S3 method for class 'tframed':
time(x, ...)
```

Arguments

`x`	a tframe or a tframed object.
`...`	arguments to be passed to other methods.

Details

The methods start and end return the start or end date of a tframe or tframed object. Periods return the number of observations (time points). frequency returns the frequency of observation, typically the number of observations in a year for economic data, but possibly something else in other contexts. The concept of frequency is not very consistently defined for time series data, and the use of the frequency method should probably be avoided where possible. In practice it seems rarely necessary, but the method makes porting of older code much easier.

Value

Depends

See Also

`tframe`, `tframed` `start` `end` `frequency` `periods` `time` `lag` `diff`

Examples

```
z <- ts(rnorm(100), start=c(1982,1), frequency=12)
tfstart(z)
z <- tframed(matrix(rnorm(200), 100,2),
        tf=list(start=c(1982,1), frequency=12))
tfend(z)
periods(z)
time(z)
```

`tfwindow` *Truncate a Time Series*

Description

Truncate a time series object to a time window.

Usage

```
tfwindow(x, tf=NULL, start=tfstart(tf), end=tfend(tf), warn=TRUE)
## Default S3 method:
tfwindow(x, tf=NULL, start=tfstart(tf), end=tfend(tf), warn=TRUE)
## S3 method for class 'ts':
tfwindow(x, tf=NULL, start=tfstart(tf), end=tfend(tf), warn=TRUE)
## S3 method for class 'tframe':
tfwindow(x, tf=NULL, start=tfstart(tf), end=tfend(tf), warn=TRUE)
## S3 method for class 'zoo':
tfwindow(x, tf=NULL, start=tfstart(tf), end=tfend(tf), warn=TRUE)
## S3 method for class 'its':
tfwindow(x, tf=NULL, start=tfstart(tf), end=tfend(tf), warn=TRUE)
```

Arguments

`x`	A time series object.
`start`	A start date of a format compatible with the time series
`end`	An end date of a format compatible with the time series
`tf`	A tframe or tframed object
`warn`	A logical indicating if warning should be produced

Details

If `start` or `end` are omitted and `tf` is specified then the start or end is taken from the `tf` object. For ts class objects this function calls window but makes more effort to preserve `seriesNames` if `x` has them. It also supports the optional argument `warn` to suppress warning messages. Frequently it is convenient to write code which always truncates to a window without first checking if the data is already within the window. Since window produces a warning in this situation, the optional argument is frequently useful when tfwindow is used by other code. In Splus `tfwindow` also corrects for some bugs in older versions of `window`.

The method for windowing a tframe is a utility to be used by other programs and would not typically be called by a user.

Value

A time series object similar to x, but typically spanning a shorter time period.

Examples

```
z <- ts(matrix(rnorm(24),24,1), start=c(1980,1), frequency=4)
zz <- tfwindow(z, start=c(1982,2))
zzz <- matrix(rnorm(24),24,1)
tframe(zzz) <- tframe(z)
tfwindow(zzz, tf=tframe(zz))
```

`trimNA` *Trim NAs from Time Series*

Description

Trim NAs from the start and end of a time series object.

Usage

```
trimNA(x, startNAs=TRUE, endNAs=TRUE)
## Default S3 method:
trimNA(x, startNAs=TRUE, endNAs=TRUE)
```

Arguments

`x`	A time series matrix or an object of class TSdata.
`startNAs`	If FALSE then beginning NAs are not trimmed.
`endNAs`	If FALSE then ending NAs are not trimmed.

Details

Trim NAs from the ends of a time series object. Observations in a given period for all series are dropped if any one contains an NA.

Value

A time series object which is windowed to the subset of data which is not NAs (usually the available data).

Examples

```
trimNA(ts(rbind(NA, matrix(1:20,10,2)), start=c(1980,1), frequency=12))
```

tsScan *Read and Write Time Series to Files*

Description

Read and write time series to files.

Usage

```
tsScan(file="", skip=1, nseries=1, sep=",",
    na.strings=c("NA", "NC", "ND"), ...)

tsWrite(x, file="data", header=TRUE, sep=",", digits=16)
```

Arguments

`file`	name of file to read or write.
`x`	A time series or time series matrix.
`skip`	number of lines to skip at start of file before reading data.
`nseries`	number of columns of series to expect.
`sep`	field separator.
`na.strings`	charaters that should be treated as NA.
`header`	a logical indicating is a header line should be written.
`digits`	number of significant digits to print.
`...`	additional arguments passed to scan.

Details

Read and write a file with time series data. By default the file is comma separated values (csv) with one header line (the series names on write, ignored on read). The year and period are the first two columns, with series in following columns. These are wrappers for `scan` and `write`.

Beware that short digits settings will result in truncated data.

Value

A time series vector or matrix.

See Also

`scan`, `scan`

Examples

```
z <- ts(matrix(100 + rnorm(200),100,2), start=c(1991,1), frequency=4)
tsWrite(z, file="tmp.test.data.csv")
zz <- tsScan("tmp.test.data.csv", nseries=2)

max(abs(z - zz))
```

Index