

zoo reference card

Creation

`zoo(x, order.by)` creation of a "zoo" object from the observations `x` (a vector or a matrix) and an index `order.by` by which the observations are ordered.
For computations on arbitrary index classes, methods to the following generic functions are assumed to work: combining `c()`, querying length `length()`, subsetting `[],` ordering `ORDER()` and value matching `MATCH()`. For pretty printing an `as.character` and/or `index2char` method might be helpful.

Creation of regular series

`zoo(x, order.by, freq)` works as above but creates a "zooreg" object which inherits from "zoo" if the frequency `freq` complies with the index `order.by`. An `as.numeric` method has to be available for the index class.
`zooreg(x, start, end, freq)` creates a "zooreg" series with a numeric index as above and has (almost) the same interface as `ts()`.

Standard methods

<code>plot</code>	plotting
<code>lines</code>	adding a "zoo" series to a plot
<code>print</code>	printing
<code>summary</code>	summarizing (column-wise)
<code>str</code>	displaying structure of "zoo" objects
<code>head, tail</code>	head and tail of "zoo" objects

Coercion

<code>as.zoo</code>	coercion to "zoo" is available for objects of class "ts", "its", "irts" (plus a default method).
<code>as.class.zoo</code>	coercion from "zoo" to other classes. Currently available for <i>class</i> in "matrix", "vector", "data.frame", "list", "irts", "its" and "ts".
<code>is.zoo</code>	querying whether an object is of class "zoo"

Merging and binding

<code>merge</code>	union, intersection, left join, right join along indexes
<code>cbind</code>	column binding along the intersection of the index
<code>c, rbind</code>	combining/row binding (indexes may not overlap)
<code>aggregate</code>	compute summary statistics along a coarser grid of indexes

Mathematical operations

<code>Ops</code>	group generic functions performed along the intersection of indexes
<code>t</code>	transposing (coerces to "matrix" before)
<code>cumsum</code>	compute (columnwise) cumulative quantities: sums <code>cumsum()</code> , products <code>cumprod()</code> , maximum <code>cummax()</code> , minimum <code>cummin()</code> .

Extracting and replacing data and index

<code>index, time</code>	extract the index of a series
<code>index<-, time<-</code>	replace the index of a series
<code>coredata, coredata<-</code>	extract and replace the data associated with a "zoo" object
<code>lag</code>	lagged observations
<code>diff</code>	arithmetic and geometric differences
<code>start, end</code>	querying start and end of a series
<code>window, window<-</code>	subsetting of "zoo" objects using their index

NA handling

<code>na.omit</code>	omit NAs
<code>na.contiguous</code>	compute longest sequence of non-NA observations
<code>na.locf</code>	impute NAs by carrying forward the last observation
<code>na.approx</code>	impute NAs by interpolation
<code>na.trim</code>	remove leading and/or trailing NAs

Rolling functions

<code>rollapply</code>	apply a function to rolling margin of an array
<code>rollmean</code>	more efficient functions for computing the rolling mean, median and maximum are <code>rollmean()</code> , <code>rollmedian()</code> and <code>rollmax()</code> , respectively

Methods for regular series

<code>is.regular</code>	checks whether a series is weakly (or strictly if <code>strict = TRUE</code>) regular
<code>frequency, deltat</code>	extracts the frequency or its reciprocal value respectively from a series, for "zoo" series the functions try to determine the regularity and frequency in a data-driven way
<code>cycle</code>	gives the position in the cycle of a regular series