Package ‘FinTS’

January 26, 2024

Type Package

Title Companion to Tsay (2005) Analysis of Financial Time Series

Version 0.4-9

Description R companion to Tsay (2005) Analysis of Financial Time Series, second edition (Wiley). Includes data sets, functions and script files required to work some of the examples. Version 0.3-x includes R objects for all data files used in the text and script files to recreate most of the analyses in chapters 1-3 and 9 plus parts of chapters 4 and 11.

License GPL (>= 2)

BugReports https://r-forge.r-project.org/tracker/?group_id=84&atid=380

URL https://geobosh.github.io/FinTSDoc/ (doc),
https://r-forge.r-project.org/projects/fints/ (devel)

Depends R (>= 2.10), zoo, graphics

Suggests moments, tseries, urca, lmtest, sandwich, psych, GPArotation, chron, polynom, e1071

LazyData yes

NeedsCompilation no

Author Spencer Graves [aut],
Georgi N. Boshnakov [cre, ctb]

Maintainer Georgi N. Boshnakov <georgi.boshnakov@manchester.ac.uk>

Repository CRAN

Date/Publication 2024-01-26 19:20:02 UTC

R topics documented:

`FinTS-package` *Companion to Tsay (2005) Analysis of Financial Time Series*

Description

R companion to Tsay (2005) Analysis of Financial Time Series, second edition (Wiley). Includes data sets, functions and script files required to work some of the examples. Version 0.3-x includes R objects for all data files used in the text and script files to recreate most of the analyses in chapters 1-3 and 9 plus parts of chapters 4 and 11.

Details

Index:

```
ARIMA                   Arima with Ljung-Box
Acf                     Autocorrelation Function
ArchTest                ARCH LM Test
AutocorTest             Box-Ljung autocorrelation test
FinTS-package           Companion to Tsay (2005) Analysis of Financial
                        Time Series
```

FinTS.stats	Financial Time Series summary statistics
TsayFiles	List of the names of files downloaded from the "Analysis of Financial Data" web site.
apca	Asymptotic PCA
as.yearmon2	Conditionally convert x to yearmon if the conversion is unique, retaining x as names.
ch01data	Financial time series for Tsay (2005, ch. 1)
ch02data	Financial time series for Tsay (2005, ch. 2)
ch03data	Financial time series for Tsay (2005, ch. 3)
ch04data	Financial time series for Tsay (2005, ch. 4)
ch05data	Financial time series for Tsay (2005, ch. 5)
ch06data	Financial time series for Tsay (2005, ch. 6)
ch07data	Financial time series for Tsay (2005, ch. 7)
ch08data	Financial time series for Tsay (2005, ch. 8)
ch09data	Financial time series for Tsay (2005, ch. 9)
ch10data	Financial time series for Tsay (2005, ch. 10)
ch11data	Financial time series for Tsay (2005, ch. 11)
ch12data	Financial time series for Tsay (2005, ch. 12)
compoundInterest	Compute compound interest
findConjugates	Find complex conjugate pairs
package.dir	Directory of a package
plot.loadings	Plot loadings
plotArmaTrueacf	Plot the theoretical ACF corresponding to an ARMA model
read.yearmon	Reading Monthly zoo Series
runscript	Run a package script
url2data	Create local copies of files read from urls

See the `scripts` subdirectory of the `FinTS` installation directory = system.file(package='FinTS').

Corrections to the script files provided and contributions to script files for other chapters will be graciously accepted.

Author(s)

Spencer Graves

Maintainer: Georgi N. Boshnakov <georgi.boshnakov@manchester.ac.uk>

References

Ruey Tsay (2005) Analysis of Financial Time Series, 2nd ed. (Wiley)

See Also

https://faculty.chicagobooth.edu/ruey-s-tsay/teaching

Examples

```
# Where is the 'FinTS' directory?
system.file(package='FinTS')
```

```
# View the script file 'ch01.R', which is in the 'scripts'
# subdirectory of the system.file(package='FinTS') directory:
runscript(1, 'view')

# SP statistics in Table 1.2 of Tsay
data(d.ibmvwewsp6203)
FinTS.stats(100*d.ibmvwewsp6203[, "SP"])
```

Acf *Autocorrelation Function*

Description

Plot the ACF without the traditional noninformation unit spike at lag 0.

Usage

```
Acf(x, lag.max = NULL, type = c("correlation", "covariance", "partial"),
    plot = TRUE, na.action = na.fail, demean = TRUE, ...)

## S3 method for class 'Acf'
plot(x, ci = 0.95, type = "h", xlab = "Lag", ylab = NULL, ylim = NULL,
     main = NULL, ci.col = "blue", ci.type = c("white", "ma"),
     max.mfrow = 6, ask = Npgs > 1 && dev.interactive(),
     mar = if (nser > 2) c(3, 2, 2, 0.8) else par("mar"),
     oma = if (nser > 2) c(1, 1.2, 1, 1) else par("oma"),
     mgp = if (nser > 2) c(1.5, 0.6, 0) else par("mgp"),
     xpd = par("xpd"), cex.main = if (nser > 2) 1 else par("cex.main"),
     verbose = getOption("verbose"), acfLag0 = FALSE, ...)
```

Arguments

`x`	for 'acf': a numeric vector or time series. for 'plot.acf': an object of class 'acf'.
`lag.max`	maximum lag at which to calculate the acf.
`ci`	coverage probability for confidence interval for 'plot.acf'.
`type`	the type of 'acf' or 'plot'
`plot`	logical. If 'TRUE' the 'acf' function will call 'plot.acf'.
`na.action`	function to be called by 'acf' to handle missing values.
`demean`	logical: Should the x be replaced by `x - mean(x)` before computing the sums of squares and lagged cross products to produce the 'acf'?
`xlab,ylab,ylim,main,ci.col,ci.type,max.mfrow,ask,mar,oma,mgp,xpd,cex.main,verbose`	see the help page of `acf`: `help('acf', package = 'stats')`.
`acfLag0`	logical: TRUE to plot the traditional noninformation unit spike at lag 0. FALSE to omit that spike, consistent with the style in Tsay (2005).
`...`	further arguments passed to 'plot.acf'.

Details

These functions are provided to make it easy to plot an autocorrelation function without the noninformative unit spike at lag 0. This is done by calling plot(x, acfLag0 = FALSE, ...). Apart from the 'acfLag0' argument, the rest of the arguments are identical to those for 'acf' and 'plot.acf'.

Value

for `acf`, an object of class 'Acf', which inherits from class 'acf', as described with `help('acf', package='stats')`.

for `plot.Acf`, NULL

Author(s)

Spencer Graves for the FinTS modification of 'plot.acf'.

References

Ruey Tsay (2005) Analysis of Financial Time Series, 2nd ed. (Wiley)

See Also

`acf` `plot.acf` `Box.test` `AutocorTest`

Examples

```
data(m.ibm2697)
Acf(m.ibm2697)
Acf(m.ibm2697, lag.max=100)
Acf(m.ibm2697, lag.max=100, main='Monthly IBM returns, 1926-1997')
```

`apca` *Asymptotic PCA*

Description

Asymptotic Principal Components Analysis for a fixed number of factors

Usage

```
apca(x,nf)
```

Arguments

`x`	a numeric matrix or other object for which 'as.matrix' will produce a numeric matrix.
`nf`	number of factors desired.

Details

NOTE: This is a preliminary version of this function, and it may be modified in the future.

Value

A list with four components:

`eig`	eigenvalues
`factors`	estimated factor scores
`loadings`	estimated factor loadings
`rsq`	R-squared from the regression of each variable on the factor space

Author(s)

Ruey Tsay

References

Ruey Tsay (2005) Analysis of Financial Time Series, 2nd ed. (Wiley, sec. 9.6, pp. 436-440)

See Also

princomp

Examples

```
# Consider the monthly simple returns of 40 stocks on NYSE and NASDAQ
# from 2001 to 2003 with 36 observations.
data(m.apca0103)
dim(m.apca0103)
M.apca0103 <- with(m.apca0103,
    array(return, dim = c(36, 40),
          dimnames = list(as.character(date[1:36]),
                          paste("Co", CompanyID[seq(1, 1440, 36)], sep = ""))))

# The traditional PCA is not applicable to estimate the factor model
# because of the singularity of the covariance matrix.  The asymptotic
# PCA provides an approach to estimate factor model based on asymptotic
# properties. For the simple example considered, the sample size is
# $T$ = 36 and the dimension is $k$ = 40. If the number of factor is
# assumed to be 1, the APCA gives a summary of the factor loadings as
# below:
#
apca40 <- apca(M.apca0103, 1)
#
# (min, 1st Quartile, median, mean, 3rd quartile, max) =
# (0.069, 0.432, 0.629, 0.688, 1.071, 1.612).
#
# Note that the sign of any loading vector is not uniquely determined
# in the same way as the sign of an eigenvector is not uniquely
```

```
# determined. The output also contains the summary statistics of the
# R-squares of individual returns, i.e. the R-squares measuring the
# total variation of individual return explained by the factors. For
# the simple case considered, the summary of R-squares is (min, 1st
# Quartile, median, mean, 3rd quartile, max) =
# (0.090,0.287,0.487,0.456,0.574,0.831).
```

ArchTest *ARCH LM Test*

Description

Lagrange Multiplier (LM) test for autoregressive conditional heteroscedasticity (ARCH)

Usage

```
ArchTest (x, lags=12, demean = FALSE)
```

Arguments

x	numeric vector
lags	positive integer number of lags
demean	logical: If TRUE, remove the mean before computing the test statistic.

Details

Computes the Lagrange multiplier test for conditional heteroscedasticity of Engle (1982), as described by Tsay (2005, pp. 101-102).

This is provided for compatibility with 'archTest' in the S-Plus script in Tsay (p. 102).

Value

an object of class 'htest'

Author(s)

Bernhard Pfaff

See Also

AutocorTest

Examples

```
data(m.intc7303)
intcLM <- ArchTest(log(1+as.numeric(m.intc7303)), lag=12)
# Matches answer on Tsay (p. 102)
```

ARIMA *Arima with Ljung-Box*

Description

Fit an ARIMA model and test residuals with the Ljung-Box statistic

Usage

```
ARIMA(x, order = c(0, 0, 0),
     seasonal = list(order = c(0, 0, 0), period = NA),
     xreg = NULL, include.mean = TRUE, transform.pars = TRUE,
     fixed = NULL, init = NULL, method = c("CSS-ML", "ML", "CSS"),
     n.cond, optim.control = list(), kappa = 1e6, Box.test.lag=NULL,
     Box.test.df = c("net.lag", "lag"),
     type = c("Ljung-Box", "Box-Pierce", "rank"))
```

Arguments

`x`	a univariate time series
`order`	A specification of the non-seasonal part of the ARIMA model: the three components (p, d, q) are the AR order, the degree of differencing, and the MA order.
`seasonal`	A specification of the seasonal part of the ARIMA model, plus the period (which defaults to 'frequency(x)'). This should be a list with components 'order' and 'period', but a specification of just a numeric vector of length 3 will be turned into a suitable list with the specification as the 'order'.
`xreg`	Optionally, a vector or matrix of external regressors, which must have the same number of rows as 'x'.
`include.mean`	Should the ARMA model include a mean/intercept term? The default is 'TRUE' for undifferenced series, and it is ignored for ARIMA models with differencing.
`transform.pars`	Logical. If true, the AR parameters are transformed to ensure that they remain in the region of stationarity. Not used for 'method = "CSS"'.
`fixed`	optional numeric vector of the same length as the total number of parameters. If supplied, only 'NA' entries in 'fixed' will be varied. 'transform.pars = TRUE' will be overridden (with a warning) if any AR parameters are fixed. It may be wise to set 'transform.pars = FALSE' when fixing MA parameters, especially near non-invertibility.
`init`	optional numeric vector of initial parameter values. Missing values will be filled in, by zeroes except for regression coefficients. Values already specified in 'fixed' will be ignored.
`method`	Fitting method: maximum likelihood or minimize conditional sum-of-squares. The default (unless there are missing values) is to use conditional-sum-of-squares to find starting values, then maximum likelihood.

`n.cond` Only used if fitting by conditional-sum-of-squares: the number of initial observations to ignore. It will be ignored if less than the maximum lag of an AR term.

`optim.control` List of control parameters for 'optim'.

`kappa` the prior variance (as a multiple of the innovations variance) for the past observations in a differenced model. Do not reduce this.

`Box.test.lag` the Box.test statistic will be based on 'Box.test.lag' autocorrelation coefficients of the whitened residuals. The default is the maximum of the following:

`round(log(sum(!is.na(x))))`, recommended by Tsay (p. 27).

One more than the number of parameters estimated, not counting any 'intercept' in the model.

`Box.test.df` numeric or character variable indicating the degrees of freedom for the ch-square approximation to the distribution of the Box.test statistic. The default 'net.lag' is 'Box.test.lag' minus the number of relevant parameters estimated. The primary alternative 'lag' is the number of lags included in the computation of the statistic.

A positive number can also be provided.

`type` which Box.test 'type' should be used? Partial matching is used.

The 'rank' alternative computes 'Ljung-Box' on rank(x); see Burns (2002) and references therein.

NOTE: The default 'Ljung-Box' type generally seems to be more accurate and popular than the earlier 'Box-Pierce', which is however the default for 'Box.test'.

Details

1. Fit the desired model using 'arima'.
2. Compute the desired number of lags for Box.test
3. Apply 'AutocorTest' to the whitened residuals.

NOTE: Some software does not adjust the degrees of freedom for the number of parameters estimated. Tsay (2005) and Enders (2004) do. The need to adjust the degrees of freedom discussed by Brockwell and Davis (1990), who provide a proof describing the circumstances under which this is appropriate.

This is, however, an asymptotic result, and it would help to have simulation studies of the distribution of the Ljung-Box statistic, estimating degrees of freedom and evaluating goodness of fit. Burns recommends a rank version of the Ljung-Box test, but does not estimate degrees of freedom. If you have done such a simulation or know of a reference describing such, would you please notify the maintainer of this package?

4. If 'xreg' is supplied, compute r.squared.

Value

an 'arima' object with an additional 'Box.test' component and if 'xreg' is not null, an 'r.squared' component.

NOTE: The 'Box.test' help page in R 2.6.1 says, 'Missing values are not handled.' However, if 'x' contains NAs, 'ARIMA' still returns a numeric answer that seems plausible, at least in some examples. Therefore, either this comment on the help page is wrong (or obsolete) or the answer can not be trusted with NAs.

Author(s)

Spencer Graves for the ARIMA{FinTS} wrapper for arima, written by the R Core Team, and Box.test, written by A. Trapletti. John Frain provided the citation to a proof in Brockwell and Davis (1990) that the degrees of freedom for the approximating chi-square distribution of the Ljung-Box statistic should be adjusted for the number of parameters estimated. Michal Miklovic provided the citation to Enders (2004).

References

Brockwell and Davis (1990) Time Series: Theory and Methods, 2nd Edition (Springer, page 310).

Walter Enders (2004) Applied Econometric Time Series (Wiley, pp. 68-69)

Greta Ljung and George E. P. Box (1978) 'On a measure of lack of fit in time series models', Biometrika, vol. 66, pp. 67-72.

Ruey Tsay (2005) Analysis of Financial Time Series, 2nd ed. (Wiley, ch. 2)

Patrick Burns (2002) 'Robustness of the Ljung-Box Test and its Rank Equivalent', https://www.burns-stat.com/pages/Working/ljungbox.pdf, accessed 2007.12.29.

See Also

arima Box.test tsdiag

Examples

```
##
## Examples from 'arima'
##
lh100 <- ARIMA(lh, order = c(1,0,0))
lh100$Box.test
# df = 3 = round(log(lh)) - 1
# 2 parameters are estimated, but 1 is 'intercept',
# so it doesn't count in the 'df' computation

lh500 <- ARIMA(lh, order = c(5,0,0))
lh500$Box.test
# round(log(length(lh))) = 4
# Default Box.test.lag = min(5+1, 4) = 6,
# so df = 1;  without the min(5+1, ...), it would be -1.
lh500$Box.test$method # lag = 6

lh101 <- ARIMA(lh, order = c(1,0,1))
lh101$Box.test
# works with mixed ARMA

USAccD011 <- ARIMA(USAccDeaths, order = c(0,1,1),
                   seasonal = list(order=c(0,1,1)))
USAccD011$Box.test
# df = round(log(length(USAccDeaths))) - 2:
# correct 'df' with nonstationary 'seasonal' as well

LakeH200 <- ARIMA(LakeHuron, order = c(2,0,0),
```

```
                  xreg = time(LakeHuron)-1920)
LakeH200$Box.test
# correct 'df' with 'xreg'
LakeH200$r.squared

## presidents contains NAs
## graphs in example(acf) suggest order 1 or 3
require(graphics)
(fit1 <- ARIMA(presidents, c(1, 0, 0)))
fit1$Box.test
tsdiag(fit1)
##
## Example with multiple 'xreg' variables
##
tLH <- as.numeric(time(LakeHuron)-1920)
tLH2 <- cbind(timeLH.1920 = tLH, time.sq = tLH*tLH)

LakeH200. <- ARIMA(LakeHuron, order=c(2,0,0), xreg=tLH2)
LakeH200.$r.squared
```

`as.yearmon2` *Conditionally convert x to yearmon if the conversion is unique, retaining x as names.*

Description

Convert `x` to class `"yearmon"`. If duplicate months are found, return `x`. Otherwise, return the conversion with names = `x`.

Usage

```
as.yearmon2(x, ...)
```

Arguments

`x`	object suitable for `as.yearmon`
`...`	additional argument(s) (e.g., a format) passed to `as.yearmon`.

Details

Dates for some monthly data include the day of the month on which the data were published. For many purposes, one would like to have the data as a `zoo` object with a `yearmon` index, while still retaining the full date for other purposes.

If the `yearmon` form of the input is not unique, `as.yearmon2` returns the input unchanged with a warning. Otherwise, it returns the `yearmon` conversion with the input as names.

Value

Returns either its argument or its argument converted to class `yearmon` with names.

See Also

yearmon

Examples

```
x1 <- as.Date(c("2000-01-01", "2000-01-01"))
as.yearmon2(x1)
#Warning message:
#In as.yearmon2(x1) :
#  1 duplicate months found in 'x';  returning 'x' unchanged

x2 <- as.Date(c("2000-01-01", "2000-02-01"))
as.yearmon2(x2)
# month of x2 with names x2
```

AutocorTest *Box-Ljung autocorrelation test*

Description

Ljung-Box test for autocorrelation

Usage

```
AutocorTest(x, lag = ceiling(log(length(x))),
            type = c("Ljung-Box", "Box-Pierce", "rank"),
            df = lag )
```

Arguments

x	a numeric vector or a univariate time series
lag	the statistic will be based on 'lag' autocorrelation coefficients. Tsay (p. 27-28) says, 'Simulation studies suggest that the choice of [lag = log(length(x))] provides better power performance. This general rule needs modification in analysis of seasonal time series for which autocorrelations with lags at multiples of the seasonality are more important.'
type	which Box.test 'type' should be used? Partial matching is used. The 'rank' alternative computes 'Ljung-Box' on rank(x); see Burns (2002) and references therein. NOTE: The default 'Ljung-Box' type generally seems to be more accurate and popular than the earlier 'Box-Pierce', which is however the default for 'Box.test'.
df	a positive number giving the degrees of freedom for the reference chi-squre distribution used to compute the p-value for the statistic. This makes it easy to call AutocorTest with the residuals from a fit and have the p-value computed with reference to a chi-square with degrees of freedom different from "lag". See the discussion of degrees of freedom for 'Box.test in ARIMA.

Details

This is provided for compatibility with 'autocorTest' in the S-Plus script in Tsay (p. 30). It is a wrapper for the R function `Box.test`.

Value

a list of class 'htest' containing the following components:

`statistic`	a number giving the value of the test statistic.
`paramter`	a number giving the degrees of freedom of the approximate chi-squared distribution of the test statistic used to compute the p.value.
`p.value`	the p-value of the test.
`method`	a character string indicating which type of test was performed. If(df != lag), this character string ends with paste("(lag = ", lag, ")", sep="").

Source

https://faculty.chicagobooth.edu/ruey-s-tsay/teaching

References

Ruey Tsay (2005) Analysis of Financial Time Series, 2nd ed. (Wiley)

Patrick Burns (2002) 'Robustness of the Ljung-Box Test and its Rank Equivalent', https://www.burns-stat.com/pages/Working/ljungbox.pdf, accessed 2007.12.29.

See Also

`Box.test` `ARIMA`

Examples

```
data(m.ibm2697)
AutocorTest(m.ibm2697, 5)

AT4 <- AutocorTest(m.ibm2697, 5, df=4)
str(AT4) # $method = "Box-Ljung test (lag = 5)"
```

`ch01data` *Financial time series for Tsay (2005, ch. 1)*

Description

Financial time series used in examples in chapter 1.

Usage

```
data(d.ibmvwewsp6203)
data(d.intc7303)
data(d.3m6203)
data(d.msft8603)
data(d.c8603)
data(m.ibmvwewsp2603)
data(m.intc7303)
data(m.3m4603)
data(m.msft8603)
data(m.c8603)
data(m.gs10)
data(m.gs1)
data(d.fxjp00)
data(m.fama.bond5203)
data(m.gs3)
data(m.gs5)
data(w.tb3ms)
data(w.tb6ms)
```

Format

Objects of class zoo giving simple returns for each trading period (day, week or month) for different periods, with different start dates but typically running to the end of 2003.

d.ibmvwewsp6203, m.ibmvwewsp2603 Zoo objects with 4 columns (IBM, VW, EW, and SP). Daily data starts with 1962-07-03. Monthly data starts with 1926-01-30.

d.intc7303, m.intc7303 Matrices of class zoo with a single column "Intel" starting from January 1973.

d.3m6203, m.3m6203 Matrices of class zoo with a single column "MMM". Daily data starts with 1962-07-03. Monthly data starts with 1946-02-28.

d.msft8603, m.msft8603 Matrices of class zoo with a single column "MSFT" starting from 1906-03-14.

d.c8603, m.c8603 Matrix of class zoo with a single column "C" starting from 1986-10-30.

m.gs10, m.gs1 Monthly 10-yr and 1-yr Treasury constant maturity rates (4/53-3/04)

d.fxjp00 Daily exchange rate between U.S. dollar and Japanese yen

m.fama.bond5203 Monthly bond returns as follows:

- **m1.12** 1-12m
- **m24.36** 24-36m
- **m48.60** 48-60m
- **m61.120** 61-120m

m.gs3, m.gs5 Monthly 3-yr and 5-yr Treasury constant maturity rates

w.tb3ms, w.tb6ms Weekly Treasury Bill rates

Details

The first 16 of these objects contain daily and monthly simple returns for 8 financial time series analyzed Tsay (2005, Table1.2). These 8 are SP (Standard & Poors), EW, IBM, Intel, Microsoft, and Citi-Group, beginning at different times and running to the end of 2003.

The others are used elsewhere in chapter 1.

Source

https://faculty.chicagobooth.edu/ruey-s-tsay/teaching

References

Ruey Tsay (2005) Analysis of Financial Time Series, 2nd ed. (Wiley, ch. 1)

See Also

FinTS.stats

Examples

```
# First half of Table 1.2:
data(d.ibmvwewsp6203)
data(d.intc7303)
data(d.3m6203)
data(d.msft8603)
data(d.c8603)
(Daily.Simple.Returns.pct <- rbind(
    SP = FinTS.stats(100*d.ibmvwewsp6203[, "SP"]),
    VW = FinTS.stats(100*d.ibmvwewsp6203[, "VW"]),
    EW = FinTS.stats(100*d.ibmvwewsp6203[, "EW"]),
    IBM= FinTS.stats(100*d.ibmvwewsp6203[, "IBM"]),
    Intel=FinTS.stats(100*d.intc7303[, "Intel"]),
    MMM= FinTS.stats(100*d.3m6203[, "MMM"]),
    MSFT=FinTS.stats(100*d.msft8603[, 'MSFT']),
    C  = FinTS.stats(100*d.c8603[, "C"])
) )

(Daily.log.Returns.pct <- rbind(
    SP = FinTS.stats(100*log(1+d.ibmvwewsp6203[, "SP"])),
    VW = FinTS.stats(100*log(1+d.ibmvwewsp6203[, "VW"])),
    EW = FinTS.stats(100*log(1+d.ibmvwewsp6203[, "EW"])),
    IBM= FinTS.stats(100*log(1+d.ibmvwewsp6203[, "IBM"])),
    Intel=FinTS.stats(100*log(1+d.intc7303[,"Intel"])),
    MMM= FinTS.stats(100*log(1+d.3m6203[, "MMM"])),
    MSFT=FinTS.stats(100*log(1+d.msft8603[, 'MSFT'])),
    C  = FinTS.stats(100*log(1+d.c8603[, "C"]))
) )
```

ch02data *Financial time series for Tsay (2005, ch. 2)*

Description

Financial time series used in examples in chapter 2.

Usage

```
data(m.ibm2697)
data(m.vw2697)
data(q.gnp4791)
data(m.ibm3dx2603)
data(m.3m4697)
data(q.gdp4703)
data(d.sp9003lev)
data(q.jnj)
data(m.decile1510)
data(w.gs1n36299)
```

Format

Objects of class zoo giving simple returns for each trading period (day, week or month) for different periods.

m.ibm2697, m.vw2697 Monthly returns for IBM stock and the value weighted index from 1926 to 1997.

q.gnp4791 Growth rate of U.S. quarterly real gnp, from 1947Q2 to 1991Q1.

m.ibm3dx2603 Monthy returns of IBM stock, the value and equal weighted and Standard and Poors indices from 1926 through 2003.

m.3m4697 Monthly simple returns of 3M stock from Feb., 1946 through Dec. 2003.

q.gdp4703 U.S. quarterly GDP from 1947 through 2003

d.sp9003lev Daily values of S&P 500 index from 1990 through 2003.

q.jnj Quarterly earnings of Johnson & Johnson from 1960 through 1980.

m.decile1510 Monthly simple returns of Deciles 1, 5, 10. Decile 1 means the weighted returns of companies in the first 10 percent of market cap (i.e. 0 to 10). (Thus, it is not the 10th percentile.) Decile 10 means the returns of the top 10 percent of the companies (market cap). Therefore, decile 1 is the smallest listed companies, and decile 10 is for the largest companies.

The 'index' of 'm.decile1510' has class 'Date'. Since it's a monthly series, it would be better for many purposes if it had 'index' of class 'yearmon'. See the 'examples' below for how to achieve this conversion.

w.gs1n36299 zoo object with two columns, 'gs1' and 'gs3', giving weekly 1-yr & 3-yr interest rates from 1962-01-05 through 2007-11-02. These data were reextracted from the Federal Reserve Bank at St. Louis to replace data from the book's web site that had obvious data quality problems (e.g., a date of 1962-08-32).

To get data covering January 4, 1962, through September 10, 1999, use `window(w.gs1n36299, start = as.Date("1962-01-12"),end=as.Date("1999-09-10"));` see 'examples' below.

Author(s)

Spencer Graves with help from Gabor Grothendieck.

Source

https://faculty.chicagobooth.edu/ruey-s-tsay/teaching

References

Ruey Tsay (2005) Analysis of Financial Time Series, 2nd ed. (Wiley, ch. 2)

See Also

ch01data

Examples

```
## m.decile1510 has 'index' of class 'Date'
## Since it's a monthly series, for many purposes,
## it should have 'index' of class 'yearmon'.
## To get this, do the following:
##
data(m.decile1510)
mDecile1510 <- zoo(m.decile1510, as.yearmon(index(m.decile1510)))

## w.gs1n36299 covers a broader range than used in
## Tsay (2005, sec. 2.9, pp. 80ff):  subset using 'window':
##
data(w.gs1n36299)
w.gs1n3 <- window(w.gs1n36299, start = as.Date("1962-01-12"),
    end = as.Date("1999-09-10"))
```

`ch03data` *Financial time series for Tsay (2005, ch. 3)*

Description

Financial time series used in examples in chapter 3.

Usage

```
#m.intc7303
data(exch.perc)
data(sp500)
#m.ibm2697
#d.ibmvwewsp6203
data(m.ibmspln)
data(m.ibmsplnsu)
data(d.sp8099)
```

Format

Three data stes used in chapter 3 are also used in chapter 1 or 2 and are documented with 'ch01data' or 'ch02data': In particular, 'm.intc7303' and 'd.ibmvwewsp6203' are used in chapters 1 and 3 and are documented with 'ch01data'; 'm.ibm2697' is used in chapters 2 and 3 is documented with ch02data.

The other data sets used in chapter 3 are as follows:

exch.perc numeric vector of length 2497 giving percentage changes in the exchange rate between the German mark and the US dollar in 10 minute intervals, June 5-19, 1989. (The book describes analyses of 2488 observations. If these 2497 observations are plotted, it is difficult to see any differences from Figure 3.2.)

sp500 object of class 'zooreg' giving the monthly excess returns of the S&P 500 index starting from 1926. This zooreg object is labeled assuming it starts in January, though the book does not say whether it starts in January or just some time in 1926. (Many of the files included date with the data, but 'sp500.dat' did not.)

m.ibmspln object of class 'zooreg' giving the monthly log returns of IBM stock and S&P 500 index from January 1926 to December 1999 for 888 observations. NOTE: The examples in the book use only the first 864 of these observations.

m.ibmsplnsu same as 'm.ibmspln' but with a third column 'summer' that is 1 in June, July and August, and 0 otherwise.

d.sp8099 zoo object giving the average daily returns of the S&P 500 from 1980 through 1999.

Source

https://faculty.chicagobooth.edu/ruey-s-tsay/teaching

References

Ruey Tsay (2005) Analysis of Financial Time Series, 2nd ed. (Wiley, ch. 3)

See Also

ch01data, ch02data

ch04data *Financial time series for Tsay (2005, ch. 4)*

Description

Financial time series used in examples in chapter 4.

Usage

```
data(m.unrate)
#d.ibmvwewsp6203
#m.3m4697
#q.gnp4791
data(w.3mtbs7097)
data(m.ibmln2699)
data(q.unemrate)
```

Format

Three data stes used in chapter 4 are also used earlier: 'd.ibmvwewsp6203' is used in chapter 1, and 'm.3m4697' and 'q.gnp4791' are used in chapter 2; these three data objects are docuemented in 'ch01data' or 'ch02data'.

The other data sets used in chapter 4 are as follows:

m.unrate zoo object giving the monthly US civilian unemployment rate from 1948 through 2004.

w.3mtbs7097 zoo object giving the US weekly 3-month treasury bill rate in the secondary market from 1970 through 1997.

m.ibmln2699 zoo object giving the monthly log returns in percentages of IBM stock from 1926 through 1999.

q.unemrate zoo object giving the US quarterly unemployment rate seasonally adjusted from 1948 through 1993.

Source

https://faculty.chicagobooth.edu/ruey-s-tsay/teaching

References

Ruey Tsay (2005) Analysis of Financial Time Series, 2nd ed. (Wiley, ch. 4)

See Also

ch01data ch02data

ch05data *Financial time series for Tsay (2005, ch. 5)*

Description

Financial time series used in examples in chapter 5.

Usage

```
data(ibm)
data(ibm9912.tp)
data(ibmdurad)
data(ibm1to5.dur)
data(ibm91.ads)
data(ibm91.adsx)
data(day15.ori)
data(day15)
```

Format

ibm IBM transactions data (11/1/1990 - 1/31/1991)

data.frame of date.time, volume, bid, ask, and price of IBM stock transactions. date.time is of class 'chron', while volume, bid, ask, and price are all numeric. Some tranactions have the same date.time values, which is why this is a data.frame and not a zoo object.

ibm9912.tp IBM transactions data of December 1999: data.frame of date.time and price.

ibmdurad Adjusted time durations between trades of IBM stock (11/01/1990-1/31/1991).

Format: data.frame with columns date.time and adjusted.duration

ibm1to5.dur subset of 'ibmdurad' limited to positive durations in the first 5 trading days.

ibm91.ads a data.frame on the changes in the price of IBM stock transactions between November 1, 1990 and January 31, 1991. This period includes 63 trading days, during which 59,838 transactions were recorded during normal trading hours. The first transaction for each day was dropped leaving the 59,775 transactions in this data.frame.

A.priceChange 1 if a price change from the previous trade, 0 otherwise

DirectionOfChg 1 if positive, -1 if negative, 0 if no change

SizeInTicks Size of the price change in number of ticks of 1/8 of a US dollar.
NOTE: In this data.frame there are 10 anomalous records for which `A.priceChange != 0` but `SizeInTicks == 0`. These correspond to price changes of half a tick, which got rounded down to 0.

ibm91.adsx a data.frame with 6 variables the same transactions as in 'ibm91.ads':

volume.thousands thousands of shares traded

time.betw.trades seconds between the previous two trades

bid.ask.spread the bid-ask spread in USD of the current transaction.

A.priceChange 1 if the previous trade involved a price change from its predacessor, and 0 otherwise

DirectionOfChg 1 if the previous change was positive, -1 if negative, 0 if no change

SizeInTicks Size of the price change in the previous trade in number of ticks of 1/8 of a US dollar. **NOTE:** The last three columns are ibm91.ads lagged one transaction, so `ibm91.adsx[-1, 4:5] == ibm91.ads[-59775, ]`, with 24 exceptions.

day15.ori data.frame with the transaction time and the stock price for the 728 IBM stock transactions that occurred during normal trading hours on November 21, 1990.

day15 a zoo object with the following columns supposedly summarizing only the price changes in day15.ori:

timeBetwPriceChg time in seconds since the last price change

DirectionOfChange 1 if the price increased, -1 if it decreased

priceChgTicks price change in number of ticks of USD 1/8.

nTradesWoChg number of trades without a price change since the previous price change ... supposedly. These numbers do not match a manual extraction of these data from 'day15.ori'.

multTrans 1 if there were multiple transactions within the same one second interval, 0 if not.

dailyCumChg cumulative price change in USD since the start of normal trading on November 21, 1990.

Source

https://faculty.chicagobooth.edu/ruey-s-tsay/teaching

References

Ruey Tsay (2005) Analysis of Financial Time Series, 2nd ed. (Wiley, ch. 5)

See Also

`ch01data` `ch02data`

`ch06data` *Financial time series for Tsay (2005, ch. 6)*

Description

Financial time series used in examples in chapter 6.

Usage

```
data(d.ibmy98)
data(d.cscoy99)
```

Format

Objects of class zoo giving returns for each trading day for different periods

d.ibmy98 Zoo object giving daily simple returns of IBM stock for each trading day in 1998.

d.cscoy99 Zoo object giving daily log returns of Cisco stock for each trading day in 1999.

Source

https://faculty.chicagobooth.edu/ruey-s-tsay/teaching

References

Ruey Tsay (2005) Analysis of Financial Time Series, 2nd ed. (Wiley, ch. 6)

See Also

ch01data, ch02data, ch03data, ch04data, ch05data

---

ch07data *Financial time series for Tsay (2005, ch. 7)*

Description

Financial time series used in examples in chapter 7.

Usage

```
data(d.ibm6298wmx)
data(d.intc7297)
```

Format

d.ibm6298wmx a zoo object of 9190 observations on several series relating to IBM stock, 1962-07-03 to 1998-12-31:

- **dailySimpleRtns** daily simple returns in percentages of IBM stock
- **day** numbers 1:9190
- **meanCorrectedLogRtns** mean-corrected log returns
- **Q4** 1 for October, November, December, and 0 otherwise
- **drop2.5pct** an indicator variable for the behavior of the previous trading day. Specifically, this is 1 if the meanCorrectedLogRtns for the previous day was at most (-0.025).
- **nOfLast5outside2.5pct** number of the last 5 days for which the meanCorrectedLogRtns exceeded +/-2.5
- **annualTrend** an annual trend defined as (year-1961)/38.
- **GARCH1.1volatility** a volatility series based on a Gaussian GARCH(1,1) model for the mean-corrected log returns.
  The simpleDailyRtns and the zoo index are from 'd-ibm6298.txt' from the book's web site.
  The 'day' and 'meanCorrectedLogRtns' are from 'd-ibmln98wm.txt'.
  The last 5 columns are from 'd-ibml25x.txt'; they are described on p. 332 of the book.

d.intc7297 a zoo object of daily log returns of Intel stock, 1972-12-15 to 1997-12-31.

Source

https://faculty.chicagobooth.edu/ruey-s-tsay/teaching

References

Ruey Tsay (2005) Analysis of Financial Time Series, 2nd ed. (Wiley, ch. 7)

See Also

ch01data, ch02data, ch03data, ch04data, ch05data, ch06data

---

ch08data *Financial time series for Tsay (2005, ch. 8)*

Description

Financial time series used in examples in chapter 8.

Usage

```
data(m.ibmsp2699ln)
data(m.bnd)
data(m.gs1n3.5301)
data(w.tb3n6ms)
data(sp5may)
```

Format

m.ibmsp2699ln zoo object giving monthly simple and log returns of IBM stock and the Standard and Poor's 500 from 1926 through 1999. (This combines files 'm-ibmsp2699.txt' and 'm-ibmspln.txt' from the book's web site.)

m.bnd zoo object giving the monthly simple returns of 30, 20, 10, 5 and 1 year maturity bonds from 1942 through 1999.

m.gs1n3.5301 zoo object giving 1 and 3 year US Treasury constant maturity interest rates from April 1953 to January 2001 (used in Example 8.6, pp. 373ff).

w.tb3n6ms zoo object giving weekly 3 and 6 month US Treasury Bill interest rates from 1958-12-12 to 2004-08-06 (used in Sect. 8.6.5, pp. 385ff).

sp5may A data.frame of 7061 observations on 4 variables based on minute-by-minute observations of the Standard and Poor's 500 Futures and prices in May 1993.

These data are used, after some processing, in Tsay(Sect. 8.7.2, pp. 392ff). Unfortunately, it's not yet clear what these numbers are. The following is a current guess and will doubtless change in the future.

logFuture logarithms of June Futures contracts traded at the Chicago Mercantile Exchange. The first difference of this series appears to be plotted in Figure 8.16(a), after replacing '10 extreme values (5 on each side) by the simple average of their two nearest neighbors.' (p. 392)

logPrice logarithms of Standard and Poor's 500 price levels. The first differences of this series appears to be plotted in Figure 8.16(b), after adjustment similar to that for 'logFuture'.

dailyAvgSomething numbers that assume 19 distinct levels separated by 18 discrete jumps. The name of this will likely change whenever more information about it can be obtained for this documentation.

day index for the 19 distinct levels assumed by 'dailyAvgSomething'. This is probably the trading day in May 1993. However, there appear to have been 20 trading days in that month, so if these 19 levels do correspond to trading days, it's not clear which date is missing.

These data were analyzed by Forbes, Kalb, and Kofman (1999); Tsay (1998) was also referenced with the discussion of the analysis of these data.

Source

https://faculty.chicagobooth.edu/ruey-s-tsay/teaching

References

Forbes, C. S., Kalb, G. R. J., and Kofman, P. (1999) 'Bayesian Arbitrage Threshold Analysis', Journal of Business and Economic Statistics, 17: 364-372.

Ruey Tsay (1998) 'Testing and Modeling Multivariate Threshold Models', Journal of the American Statistical Association, 93: 1188-1202.

Ruey Tsay (2005) Analysis of Financial Time Series, 2nd ed. (Wiley, ch. 8)

See Also

ch01data, ch02data

---

ch09data *Financial time series for Tsay (2005, ch. 9)*

Description

Financial time series used in examples in chapter 9.

Usage

```
data(m.fac9003)
data(m.cpice16.dp7503)
data(m.barra.9003)
data(m.5cln)
#data(m.bnd) <- documented with ch08, also used in ch09
data(m.apca0103)
```

Format

m.fac9003 a zoo object of 168 observations giving simple excess returns of 13 stocks and the Standard and Poor's 500 index over the monthly series of three-month Treasury bill rates of the secondary market as the risk-free rate from January 1990 to December 2003. (These numbers are used in Table 9.1.)

AA Alcoa

AGE A. G. Edwards

CAT Caterpillar

F Ford Motor

FDX FedEx

GM General Motors

HPQ Hewlett-Packard

KMB Kimberly-Clark

MEL Mellon Financial

NYT New York Times

PG Proctor & Gamble

TRB Chicago Tribune

TXN Texas Instruments

SP5 Standard & Poor's 500 index

m.cpice16.dp7503 a zoo object of 168 monthly on two macroeconomic variables from January 1975 through December 2002 (p. 412):

CPI consumer price index for all urban consumers: all items and with index 1982-1984 = 100

CE16 Civilian employment numbers 16 years and over: measured in thousands

m.barra.9003 a zoo object giving monthly excess returns of ten stocks from January 1990 through December 2003:

AGE A. G. Edwards

C Citigroup

MWD Morgan Stanley

MER Merrill Lynch

DELL Dell, Inc.

IBM International Business Machines

AA Alcoa

CAT Caterpillar

PG Proctor & Gamble

m.5cln a zoo object giving monthly log returns in percentages of 5 stocks from January 1990 through December 1999:

IBM International Business Machines

HPQ Hewlett-Packard

INTC Intel

MER Merrill Lynch

MWD Morgan Stanley Dean Witter

m.apca0103 data.frame of monthly simple returns of 40 stocks from January 2001 through December 2003, discussed in sect. 9.6.2, pp. 437ff.

CompanyID 5-digit company identification code

date the last workday of the month

return in percent

Source

https://faculty.chicagobooth.edu/ruey-s-tsay/teaching

References

Ruey Tsay (2005) Analysis of Financial Time Series, 2nd ed. (Wiley, ch. 7)

See Also

ch01data, ch02data, ch03data, ch04data, ch05data, ch06data

Examples

```
data(m.apca0103)
dim(m.apca0103)
# 1440 3;  1440 = 40*36
# Are the dates all the same?
sameDates <- rep(NA, 39)
for(i in 1:39)
    sameDates[i] <- with(m.apca0103,
                         all.equal(date[1:36], date[(i*36)+1:36]))
stopifnot(all(sameDates))
M.apca0103 <- with(m.apca0103, array(return, dim = c(36, 40), dimnames =
    list(NULL, paste("Co", CompanyID[seq(1, 1440, 36)], sep=""))))
```

---

ch10data *Financial time series for Tsay (2005, ch. 10)*

Description

Financial time series used in examples in chapter 10.

Usage

```
data(d.hkja)
data(m.pfe6503)
data(m.mrk6503)
#data(m.ibmsp2699)
# <- 2 of the 4 columns in m.ibmsp2699ln
# documented with ch08data
data(d.spcscointc)
```

Format

One data set used in chapter 10 is also used earlier: 'm.ibmsp2699' is the first 2 of the 4 columns of 'm.ibmsp2699ln' used in chapter 8.

The other data sets used in chapter 10 are as follows:

d.hkja zoo object giving the daily log returns of HK and Japan market indices from 1996-01-01 through 1997-05-05 (used in Example 10.1).

m.pfe6503, m.mrk6503 zoo objects giving the monthly simple returns including dividends of Pfizer and Merk stocks.

d.spcscointc data.frame giving 2275 daily log returns of three items from January 2, 1991 through December 31, 1999:

SP500 Standard & Poor's 500 index

Cisco Cisco stock

Intel Intel stock

NOTE: This date range seems to include 2280 trading days in the New York Stock Exchange. Since the file on the book's web site did not include dates and since there appear to be more trading days than observations, dates are not currently provided with these observations. This may change with a future revision of this package.

Source

https://faculty.chicagobooth.edu/ruey-s-tsay/teaching

References

Ruey Tsay (2005) Analysis of Financial Time Series, 2nd ed. (Wiley, ch. 10)

See Also

ch01data, ch02data, ch03data, ch04data, ch05data, ch06data, ch07data, ch08data, ch09data

---

ch11data *Financial time series for Tsay (2005, ch. 11)*

Description

Financial time series used in examples in chapter 11.

Usage

```
data(aa.3rv)
# m.fac9003 described in ch09data
# q.jnj described in dh02data
```

Format

The text of chapter 11 considers one data set not used in previous chapters plus two that are. Monthly excess returns of GM stock are used in Table 9.1 of Chapter 9. Quarterly earnings of Johnson and Johnson are used in Chapter 2.

The data set introduced with Chapter 11 is as follows:

aa.3rv. a zoo object of daily 5, 10, and 20 minute realized volatility of Alcoa stock from 2003-01-02 through 2004-05-07.

Source

https://faculty.chicagobooth.edu/ruey-s-tsay/teaching

References

Ruey Tsay (2005) Analysis of Financial Time Series, 2nd ed. (Wiley, ch. 11)

See Also

ch09data, ch02data

---

ch12data *Financial time series for Tsay (2005, ch. 12)*

Description

Financial time series used in examples in chapter 12.

Usage

```
data(w.gs3n1c)
data(w.gs3c)
data(m.sp6299)
data(m.ibmspln6299)
data(m.sp5.6204)
data(m.geln)
```

Format

w.gs3n1c a zoo object of the change series of weekly US interest rates (3 and 1 year maturities) from Jan. 5, 1962, to Sep. 10, 1999. This was obtained via

```
diff(window(w.gs1n36299, start=as.Date("1962-01-05"), end=as.Date("1999-09-10")))[,
2:1]
```

to get the dates with the data. Then 'all.equal' confirmed that these numbers matched those in the file read from the web site (which did not have dates).

These are used in Example 12.1, pp. 556ff.

w.gs3c a zoo object giving the change series of weekly US 3-year maturity interest rates from March 18, 1988, to Sept. 10, 1999. This was obtained via

`window(w.gs3n1c[, 1], start=as.Date("1988-03-18"), end = as.Date("1999-09-10")).`

Then 'all.equal' confirmed that these numbers matched those read from the web site.

These data are used in Example 12.2, pp. 564ff.

m.sp6299 Monthly log returns of S&P 500 index from January 1962 to December 1999. These data are used in Example 12.3, pp. 569ff.

These data are a subset of 'm.ibmspln', used in chapter 3. That series has dates, which were not provided in the file associated with this series on the book's web site. Moreover, the file with chapter 12 has only 4 significant digits where the earlier file has 6. Since the other data are otherwise identical, this 'm.sp6299' was constructed as

`window(m.ibmspln[, 2], start = yearmon(1962), end = yearmon(1999+11/12)).`

m.ibmspln6299 Monthly log returns of IBM stock and the S&P 500 index from January 1962 to December 1999. These data are used in Example 12.4, pp. 573ff.

These data are an expansion of 'm.sp6299' and were similarly obtained from 'm.ibmspln'.

m.sp5.6204 Monthly log returns of S&P 500 index from January 1962 to November 1999. These data are used in Example 12.5, pp. 586ff.

m.geln Monthly log returns of GE stock from January 1926 to December 1999. These data are used in Example 12.6, pp. 591ff.

Source

https://faculty.chicagobooth.edu/ruey-s-tsay/teaching

References

Ruey Tsay (2005) Analysis of Financial Time Series, 2nd ed. (Wiley, ch. 12)

See Also

`ch01data`, `ch02data`, `ch03data`, `ch04data`, `ch05data`, `ch06data`, `ch07data`, `ch08data`, `ch09data`, `ch10data`, `ch11data`

`compoundInterest` *Compute compound interest*

Description

Compute compound interest for a given number of periods, compounding with an indicated frequency per period.

Usage

```
compoundInterest(interest, periods = 1, frequency = 1, net.value = FALSE)

simple2logReturns(R)
```

Arguments

`interest`	rate of interest per period (usually per year).
`periods`	number of periods over which to compound.
`frequency`	number of times per period to compound; `frequency = Inf` to convert simple to log returns.
`net.value`	if TRUE, return the total value per unit invested; otherwise return net increase = (net value - 1).
`R`	simple interest to be converted to log(returns).

Details

These functions are vectorized for all arguments. The code uses optionally `expm1(x) = exp(x) - 1` and `log1p(x) = log(1+x)` which can preserve numerical precision for x very close to 0.

Value

vector of the length of the longest argument.

References

Ruey Tsay (2005) Analysis of Financial Time Series, 2nd ed. (Wiley, p. 6)

Examples

```
# "Net Value" column of Tsay Table 1.1, p. 4
compoundInterest(0.1, frequency = c(1, 2, 4, 12, 52, 365, Inf),
                      net.value = FALSE)
# Example 1.1, p. 6
compoundInterest(.0446, freq = Inf)
# Inverse of Example 1.1
simple2logReturns(.0456)
```

`findConjugates` *Find complex conjugate pairs*

Description

Find all complex conjugate pairs in a vector of complex numbers and return one number from each pair.

Usage

```
findConjugates(x, complex.eps = .Machine[["double.eps"]])
```

Arguments

`x`	a vector of complex numbers
`complex.eps`	a small positive number used to identify complex conjugates: `x[i]` and `x[j]` are considered conjugates if `(abs(x-Conj(x))` / `max(abs(x[i], x[j]))`) < `complex.eps` and (`abs(x[i]` - `x[j])` > `complex.eps`. The latter condition excludes repeated roots.

Details

1. Compute normalization `m2` = `outer(abs(x), abs(x), max)`.
2. Compute complex differences `c2` = `abs(outer(x, Conj(x), "-"))` / `m2`.
3. If any `abs(c2)` < `complex.eps`, make sure the numbers are not duplicate reals via

`(d2` = `abs(outer(x, x, "-")))` > `complex.eps`.

Value

a complex vector with one representative of each complex pair found

Author(s)

Spencer Graves and Ravi Varadhan

See Also

`plotArmaTrueacf`

Examples

```
# none
findConjugates(NULL)
findConjugates(numeric(0))
findConjugates(0:4)
findConjugates(rep(0:1,each=3))

# some
findConjugates(c(1+1i, 0, 1-1i, 2-2i, 3, 2+2i, NA))

# Testing with polyroot and solve(polynomial(...))
set.seed(1234)
if(require(polynom)){
  p <- polynomial(sample(1:10, size=45, rep=TRUE)) # degree 44
  z <- solve(p)
  findConjugates(z, complex.eps=.Machine$double.eps)
  # this identifies all 21 conjugate pairs, R 2.6.0 for Windows

  z1 <- polyroot(p)
  findConjugates(z1, complex.eps=.Machine$double.eps)
  # this only identifies only 3 conjugate pairs, R 2.6.0 for Windows
}
```

FinTS.stats *Financial Time Series summary statistics*

Description

Summary statistics as in Table 1.2, Tsay (2005), including the start date, number of observations, mean, standard deviation, skewness, excess kurtosis, min and max.

Usage

```
FinTS.stats(x)
```

Arguments

x	A univariate object of class 'zoo'

References

Ruey Tsay (2005) Analysis of Financial Time Series, 2nd ed. (Wiley, p. 11)

See Also

index, sum, is.na, mean, sd, skewness, kurtosis, min, max

Examples

```
FinTS.stats(rep(1, 5))

data(d.c8603)
FinTS.stats(100*d.c8603[, "C"])

## The following generates an error, because FinTS.stats
## expects a vector of class 'zoo', and d.c8603 is a matrix
#FinTS.stats(100*d.c8603)
```

package.dir *Directory of a package*

Description

Display a partial or complete directory of a package. By default, suppress common package contents to focus on 'demo', 'doc', 'scripts', and similar subdirectories whose contents might contain examples that could make it easier to learn capabilities of the package.

Usage

```
package.dir(package = 'base', lib.loc = NULL,
      exclude = c('chtml', 'data', 'help', 'html', 'latex', 'libs',
                  'man', 'Meta', 'po', 'R', 'R-ex', 'src'),
      include = NULL, pattern = NULL, recursive = FALSE)
```

Arguments

`package`	character string naming a locally installed package. If 'package' is not locally installed, it is an error.
`lib.loc`	a character vector with path names of R libraries, or 'NULL'. The default value of 'NULL' corresponds to all libraries currently known. If the default is used, the loaded packages are searched before the libraries.
`exclude`	either NULL or a character vector naming subdirectories of 'package' to exclude from the list. If 'include' is not NULL, 'exclude' is ignored.
`include`	either NULL or a character vector naming subdirectories of 'package' to exclude from the list. If 'include' is not NULL, 'exclude is ignored.
`pattern`	an optional regular expression passed with the results of system.file to dir. Only file names which match the regular expression will be returned. This is ignored if 'recursive' is FALSE.
`recursive`	logical. Should the listing recurse into subdirectories?

Details

1. `fullPath <- system.file(package = package, lib.loc = lib.loc)`
2. `Dir <- dir(fullPath)`
3. Restrict Dir only to 'include' if provided and to all but 'exclude' otherwise.
4. If recursive, return a list produced by `dir` for each of the subdirectories of interest determined in step 3. Else, return only the list of subdirectories from step 3.

Value

If recursive, a list of the contents of the subdirectories of interest. Else, a character vector of the names of the relevant subdirectories.

Author(s)

Spencer Graves

See Also

`system.file` `dir` `file.path`

Examples

```
package.dir() # 'demo'
package.dir(recursive = TRUE) # contents of 'demo'
package.dir('nlme') # 'mlbook', 'scripts'
```

plot.loadings *Plot loadings*

Description

Plots loadings as a separate barplot for each factor.

Usage

```
## S3 method for class 'loadings'
plot(x, n = 5, k = ncol(x), mfrow = c(k, 1), ...)
```

Arguments

x	A loadings object.
n	Number of components of each factor to plot.
k	Number of factors to plot.
mfrow	Passed to par(mfrow = ...) if k > 1.
...	Other arguments passed to barplot.

Details

The top n components of each of the top k factors are displayed in a separate barplot.

Value

Return value is a list of the return values from each barplot invocation.

See Also

barplot

Examples

```
data(m.barra.9003)
rtn <- m.barra.9003
stat.fac <- factanal(rtn, factors = 3)
m.barra.loadings <- loadings(stat.fac)
plot(m.barra.loadings)
```

`plotArmaTrueacf` *Plot the theoretical ACF corresponding to an ARMA model*

Description

Compute the roots and theoretical ACF corresponding to an ARMA model

Usage

```
plotArmaTrueacf(object, lag.max = 20, pacf = FALSE, plot = TRUE,
          xlab = "lag", ylab = c("ACF", "PACF")[1+pacf],
          ylim = c(-1, 1) * max(ACF), type = "h",
          complex.eps = 1000 * .Machine[["double.neg.eps"]], ...)
```

Arguments

`object`	either a numeric vector or a list with components 'ar' and 'ma'. If 'object' is numeric, it is interpreted as a model with no 'ma' part.
`lag.max`	the maximum number of lags for which to calculate the ACF or PACF.
`pacf`	logical. Should the partial autocorrelations be returned?
`plot`	logical. Should the ACF (or PACF) be plotted?
`xlab, ylab, ylim, type`	arguments for 'plot'
`complex.eps`	a small positive number used to identify complex conjugates: Let 'roots' = the vector of p roots of the characteristic polynomial of the autoregressive part of 'object'. This is used by 'findConjugates': x[i] and x[j] are considered conjugate if their relative difference exceeds complex.eps but the relative difference of their conjugates is less than complex.eps. We use 'solve' in the 'polynom' package, because it was substially more accurate for cases we tested in R 2.6.0 than 'polyroot'.
`...`	optional arguments passed to 'plot'.

Details

1. Compute and test stationarity. An ARMA process is stationary if all the roots of its AR component lie inside the unit circle (Box and Jenkins, 1970). If the process is not stationary, a warning is issued, and no plot is produced.

2. Compute and plot the theoretical ACF.

3. Analyze periodicity of any complex roots

Value

a list with the following components

`roots` a complex vector of the roots sorted by modulus and sign of the immaginary part.

`acf, pacf` a named numeric vector of the estimated ACF (or PACF of 'pacf = TRUE').

`periodicity` a data.frame with one row for each complex conjugate pair of roots and columns 'damping' and 'period'.

Source

https://faculty.chicagobooth.edu/ruey-s-tsay/teaching

References

George E. P. Box and Gwilym M. Jenkins (1970) Time Series Analysis, Forecasting and Control (Holden-Day, sec. 3.4.1. Stationarity and invertibility properties of Mixed autoregressive-moving average processes)

Ruey Tsay (2005) Analysis of Financial Time Series, 2nd ed. (Wiley, ch. 2)

See Also

`solve.polynomial`, `ARMAacf`

Examples

```
# Tsay, Figure 2.3
op <- par(mfcol = c(1, 2))
plotArmaTrueacf(.8, lag.max = 8)
title("(a)")
plotArmaTrueacf(-.8, lag.max = 8)
title("b")
par(op)

# Tsay, Figure 2.4
op <- par(mfrow = c(2,2))
plotArmaTrueacf(c(1.2, -.35))
title("(a)")
plotArmaTrueacf(c(.6, -.4))
title("(b)")
plotArmaTrueacf(c(.2, .35))
title("(c)")
plotArmaTrueacf(c(-.2, .35))
title("(d)")
par(op)

# Tsay, Example 2.1
data(q.gnp4791)
(fit.ar3 <- ar(q.gnp4791, aic = FALSE, order = 3))
plotArmaTrueacf(fit.ar3)
```

read.yearmon *Reading Monthly zoo Series*

Description

Read a text file containing monthly data with a date column and return a zoo object with index = a yearmon series with the dates read as names.

Usage

```
read.yearmon(file, format = "", tz = "", FUN = NULL, regular = FALSE,
             index.column = 1, ...)
```

Arguments

`file`	character giving the name of the file which the data are to be read from/written to. See `read.table` and `write.table` for more information.
`format`	date format argument passed to `as.Date.character`.
`tz`	time zone argument passed to `as.POSIXct`.
`FUN`	a function for computing the index from the first column of the data. See details.
`regular`	logical. Should the series be coerced to class `"zooreg"` (if the series is regular)?
`index.column`	integer. The column of the data frame in which the index/time is stored.
`...`	further arguments passed to `read.table` or `write.table`, respectively.

Details

The arguments are as for `read.zoo`:

`TS <- read.zoo(...)`

`zoo(coredata(TS), as.yearmon2(index(TS)))`

Value

an object of class `"zoo"` or `"zooreg"`.

See Also

`read.table`, `zoo`, `read.zoo`, `coredata`, `index`, `as.yearmon2`

Examples

```
## Not run:
## turn *numeric* first column into yearmon index
## where number is year + fraction of year represented by month
z <- read.zoo("foo.csv", sep = ",", FUN = as.yearmon2)
z2 <- read.yearmon("foo.csv", sep = ",")

## End(Not run)
```

runscript *Run a package script*

Description

Run a script associated with a particular chapter

Usage

```
runscript(x, method = c('run', 'copy', 'view', 'show', 'dir'),
          ask = TRUE, fmt = "ch%02d.R", package = "FinTS",
          subdir = "scripts", lib.loc = NULL)
```

Arguments

x	an object to identify a file in package/subdir via `sprintf(fmt, x)`. For example, the default 'fmt' translates `x = 2` into 'ch02.R'. If no 'x' is specified, a directory of options is provided. CAUTION: Under some systems like ESS (Emacs Speaks Statistics) under Windows, pop-up menus such as produced by `runscript()` may not work properly.
method	One of the following: **run** run the desired script file, similar to demo or example. **copy** make a copy if the desired script file in the working directory, similar to `Stangle(vignette(...)[["file"]])`. **view** display the desired script file on R console but do not execute it. **show** display the desired script file using file.show **dir** return the directory showing only the location of the desired script. Partial matching is allowed.
ask	logical: Should `par(ask=TRUE)` be called before graphical output happens from the script?
fmt	a format to be used with 'x' in sprintf to create the name of a file in lib.loc/package/subdir.
subdir	subdirectory of package containing a file of the name constructed via `sprintf(fmt, x)`.
package	Name of a package with subdirectory 'subdir'.
lib.loc	NULL or character string identifying the location where `system.file(subdir, package, lib.loc)` will find the folder containing the file identified via `sprintf(fmt, x)`.

Details

similar to demo or example

Value

the full path and filename, invisibly unless `method == 'dir'`

Author(s)

Gabor Grothendieck and Spencer Graves

See Also

demo, sprintf, system.file, package.dir, Stangle, vignette, example

Examples

```
## Not run:
# provide a menu
runscript()

# run R/library/FinTS/scripts/ch01.R
runscript(1)

# same as:
runscript(1, "run")

# make a copy as 'ch01.R' in the working directory
runscript(1, 'copy')

# display on console only
runscript(1, 'view')

# display using file.show
runscript(1, 'show')

# where is it?
runscript(1, 'dir')

# run R/library/nlme/scripts/afda-ch01.R
if(require(fda))
    runscript(1, fmt = "afda-ch%02d.R", package = "fda")

## End(Not run)
```

TsayFiles — *List of the names of files downloaded from the "Analysis of Financial Data" web site.*

Description

A list organized by chapter and text vs. exercises of the files downloaded from the web site associated with Tsay (2005) Analysis of Financial Time Series, 2nd ed. (Springer) and stored in "~library/FinTS/scripts/TsayFiles". These facilitate the process of creating and updating the ’FinTS’ package (and documenting the creation process).

Usage

```
TsayFiles
```

Format

TsayFiles A list with names 'ch01', 'ch02', ..., 'ch12' for components describing the files associated with the corresponding chapter.

Each chapter component is a list with 'text' and 'exercises' components, where 'text' and 'exercises' are each a character array giving names for 'data', 'file', 'url', and 'found' for the data referenced in the text or exercises of that chapter:

- **data** 'file' without the extension, e.g, 'd-ibmvwewsp6203' for daily simple returns of IBM, VW, EW, SP (7/3/62-12/31/03) = 'file' without the extension.
- **file** short file name = 'data' plus the extension, e.g., 'd-ibmvwewsp6203'.txt' for daily simple returns of IBM, VW, EW, SP (7/3/62-12/31/03)
- **url** universal resource locator for the data, e.g., "http://faculty.chicagogsb.edu/ruey.tsay/teaching/fts2/d-ibmvwewsp6203.txt"
- **found** 'TRUE' if the data were found, 'FALSE' if the attempt to access the url failed.

NOTES:

(1) 13 files are referenced twice, and 2 are referenced three times on the web page. This redundancy is retained in 'TsayFiles'.

(2) A few files (most noticably some with with '.dat' extension) are referenced in the HTML code without an apparent visible link. These 'invisible files' are retained in 'TsayFiles'.

FinTS.url A character string giving the universal resource locator (URL) associated with the Tsay (2005) book:

`FinTS.url <- "https://faculty.chicagobooth.edu/ruey-s-tsay/teaching"`

Source

Tsay (2005) Analysis of Financial Time Series, 2nd ed. (Springer)

Examples

```
data(TsayFiles)
TsayFiles$ch01$exercises
```

---

url2data *Create local copies of files read from urls*

Description

Call 'download.file' with each element of a vector of character strings assumed to be URLs, create a local copy for each, and return a character matrix summarizing what was done.

Usage

```
url2data(url.)
```

Arguments

`url.` a vector of character strings assumed to be URLs, whose names are assumed to be the names to be used for local copies of the URLs.

Details

1. `fili <- names(urls.)`
2. `dati <- fili` without its extension, i.e., the part following the last '.'
3. `for(i in 1:length(url.)try(download.file(url.[i], fili[i]));` `found[i] = TRUE` if something found and FALSE if not.
4. Return a character matrix with 4 columns: data = dati, file = fili, url = url., and found.

Examples

```
# See ~R\library\FinTS\scripts\TsayFiles.R
```

Index