Package 'ftsa'

October 21, 2026

Type Package

Title Functional Time Series Analysis

Version 6.8

Date 2026-10-21

Depends R (>= 3.5.0), forecast, rainbow, fds, sde

Suggests R2jags, meboot

Imports colorspace, MASS, pcaPP, fda, pdfCluster, ecp, strucchange, e1071, psych, fGarch, KernSmooth, vars, boot, fdapace, LaplacesDemon, evgam, ROOPSD, glue, methods, longmemo

LazyLoad yes

LazyData yes

LazyDataCompression xz

ByteCompile TRUE

Maintainer Han Lin Shang <hanlin.shang@mq.edu.au>

Author Rob Hyndman [aut] (ORCID: <https://orcid.org/0000-0002-2140-5352>), Han Lin Shang [aut, cre, cph] (ORCID: <https://orcid.org/0000-0003-1769-6430>)

Description
Functions for visualizing, modeling, forecasting and hypothesis testing of functional time series.

License GPL-3

NeedsCompilation no

Repository CRAN

Date/Publication 2026-10-21 05:40:02 UTC

Contents

ftsa-package 3
centre 7
CoDa_BayesNW 8
CoDa_FPCA 10

diff.fts . . . 11
DJI_return . . . 12
dynamic_FLR . . . 12
dynupdate . . . 14
error . . . 16
ER_GR . . . 18
extract . . . 19
facf . . . 20
farforecast . . . 21
fbootstrap . . . 22
forecast.ftsm . . . 24
forecastfplsr . . . 26
fplsr . . . 27
ftsm . . . 31
ftsmiterativeforecasts . . . 33
ftsmweightselect . . . 35
GAEVforecast . . . 36
Horta_Ziegelmann_FPCA . . . 38
H_estimate_LW_NFTS . . . 40
H_estimate_RS . . . 41
is.fts . . . 43
isfe.fts . . . 43
long_run_covariance_estimation . . . 45
LQDT_FPCA . . . 46
MAF_multivariate . . . 47
mean.fts . . . 48
median.fts . . . 50
MFDM . . . 52
pcscorebootstrapdata . . . 54
plot.fm . . . 56
plot.fmres . . . 58
plot.ftsf . . . 59
plot.ftsm . . . 60
plotfplsr . . . 62
pm_10_GR . . . 63
quantile . . . 64
quantile.fts . . . 65
residuals.fm . . . 65
sd . . . 66
sd.fts . . . 67
sim_FARMA . . . 69
skew_t_fun . . . 70
stop_time_detect . . . 71
stop_time_sim_data . . . 72
summary.fm . . . 73
T_stationary . . . 74
var . . . 75
var.fts . . . 76

Index **78**

`ftsa-package` *Functional Time Series Analysis*

Description

This package presents descriptive statistics of functional data; implements principal component regression and partial least squares regression to provide point and distributional forecasts for functional data; utilizes functional linear regression, ordinary least squares, penalized least squares, ridge regression, and moving block approaches to dynamically update point and distributional forecasts when partial data points in the most recent curve are observed; performs a stationarity test for a functional time series; estimates a long-run covariance function by kernel sandwich estimator.

Author(s)

Rob J Hyndman and Han Lin Shang

Maintainer: Han Lin Shang <hanlin.shang@mq.edu.au>

References

############################ # References in Statistics ############################

R. J. Hyndman and H. L. Shang (2009) "Forecasting functional time series (with discussion)", *Journal of the Korean Statistical Society*, **38**(3), 199-221.

R. J. Hyndman and H. L. Shang (2010) "Rainbow plots, bagplots, and boxplots for functional data", *Journal of Computational and Graphical Statistics*, **19**(1), 29-45.

H. L. Shang and R. J. Hyndman (2011) "Nonparametric time series forecasting with dynamic updating", *Mathematics and Computers in Simulation*, **81**(7), 1310-1324.

H. L. Shang (2011) "rainbow: an R package for visualizing functional time series", *The R Journal*, **3**(2), 54-59.

H. L. Shang (2013) "Functional time series approach for forecasting very short-term electricity demand", *Journal of Applied Statistics*, **40**(1), 152-168.

H. L. Shang (2013) "ftsa: An R package for analyzing functional time series", *The R Journal*, **5**(1), 64-72.

H. L. Shang (2014) "A survey of functional principal component analysis", *Advances in Statistical Analysis*, **98**(2), 121-142.

H. L. Shang (2014) "Bayesian bandwidth estimation for a functional nonparametric regression model with mixed types of regressors and unknown error density", *Journal of Nonparametric Statistics*, **26**(3), 599-615.

H. L. Shang (2014) "Bayesian bandwidth estimation for a semi-functional partial linear regression model with unknown error density", *Computational Statistics*, **29**(3-4), 829-848.

H. L. Shang (2015) "Resampling techniques for estimating the distribution of descriptive statistics of functional data", *Communications in Statistics - Simulation and Computation*, **44**(3), 614- 635.

H. L. Shang (2016) "Mortality and life expectancy forecasting for a group of populations in developed countries: A robust multilevel functional data method", in C. Agostinelli, A. Basu, P. Filzmoser, D. Mukherjee (ed.), Recent Advances in Robust Statistics: Theory and Applications, Springer, India, pp. 169-184.

H. L. Shang (2016) "Mortality and life expectancy forecasting for a group of populations in developed countries: A multilevel functional data method", *Annals of Applied Statistics*, **10**(3), 1639-1672.

H. L. Shang (2016) "A Bayesian approach for determining the optimal semi-metric and bandwidth in scalar-on-function quantile regression with unknown error density and dependent functional data", *Journal of Multivariate Analysis*, **146**, 95-104.

H. L. Shang (2017) "Functional time series forecasting with dynamic updating: An application to intraday particulate matter concentration", *Econometrics and Statistics*, **1**, 184-200.

H. L. Shang (2017) "Forecasting Intraday S&P 500 Index Returns: A Functional Time Series Approach", *Journal of Forecasting*, **36**(7), 741-755.

H. L. Shang and R. J. Hyndman (2017) "Grouped functional time series forecasting: An application to age-specific mortality rates", *Journal of Computational and Graphical Statistics*, **26**(2), 330-343.

G. Rice and H. L. Shang (2017) "A plug-in bandwidth selection procedure for long-run covariance estimation with stationary functional time series", *Journal of Time Series Analysis*, **38**(4), 591-609.

P. Reiss, J. Goldsmith, H. L. Shang and R. T. Ogden (2017) "Methods for scalar-on-function regression", *International Statistical Review*, **85**(2), 228-249.

P. Kokoszka, G. Rice and H. L. Shang (2017) "Inference for the autocovariance of a functional time series under conditional heteroscedasticity", *Journal of Multivariate Analysis*, **162**, 32-50.

Y. Gao, H. L. Shang and Y. Yang (2017) "High-dimensional functional time series forecasting", in G. Aneiros, E. Bongiorno, R. Cao and P. Vieu (ed.), Functional Statistics and Related Fields, Springer, Cham, pp. 131-136.

Y. Gao and H. L. Shang (2017) "Multivariate functional time series forecasting: An application to age-specific mortality rates", *Risks*, **5**(2), Article 21.

H. L. Shang (2019) "Visualizing rate of change: An application to age-specific fertility rates", *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, **182**(1), 249-262.

H. L. Shang (2018) "Bootstrap methods for stationary functional time series", *Statistics and Computing*, **28**(1), 1-10.

Y. Gao, H. L. Shang and Y. Yang (2019) "High-dimensional functional time series forecasting: An application to age-specific mortality rates", *Journal of Multivariate Analysis*, **170**, 232-243.

D. Li, P. M. Robinson and H. L. Shang (2020) "Long-range dependent curve time series", *Journal of the American Statistical Association: Theory and Methods*, **115**(530), 957-971.

H. L. Shang (2020) "A comparison of Hurst exponent estimators in long-range dependent curve time series", *Journal of Time Series Econometrics*, **12**(1).

D. Li, P. M. Robinson and H. L. Shang (2021) "Local Whittle estimation of long range dependence for functional time series", *Journal of Time Series Analysis*, **42**(5-6), 685-695.

H. L. Shang and R. Xu (2021) "Functional time series forecasting of extreme values", *Communications in Statistics: Case Studies, Data Analysis and Applications*, **7**(2), 182-199.

U. Beyaztas, H. L. Shang and Z. Yaseen (2021) "Development of functional autoregressive model based exogenous hydrometeorological variables for river flow prediction", *Journal of Hydrology*, **598**, 126380.

U. Beyaztas and H. L. Shang (2022) "Machine learning-based functional time series forecasting: Application to age-specific mortality rates", *Forecasting*, **4**(1), 394-408.

Y. Yang, Y. Yang and H. L. Shang (2022) "Feature extraction for functional time series: Theory and application to NIR spectroscopy data", *Journal of Multivariate Analysis*, **189**, 104863.

A. E. Fernandez, R. Jimenez and H. L. Shang (2022) "On projection methods for functional time series forecasting", *Journal of Multivariate Analysis*, **189**, 104890.

H. L. Shang (2022) "Not all long-memory estimators are born equal: A case of non-stationary curve time series", *The Canadian Journal of Statistics*, **50**(1), 357-380.

X. Huang, H. L. Shang and D. Pitt (2022) "Permutation entropy and its variants for measuring temporal dependence", *Australian and New Zealand Journal of Statistics*, **64**(4), 442-477.

H. L. Shang, J. Cao and P. Sang (2022) "Stopping time detection of wood panel compression: A functional time series approach", *Journal of the Royal Statistical Society: Series C*, **71**(5), 1205-1224.

C. Tang, H. L. Shang and Y. Yang (2022) "Clustering and forecasting multiple functional time series", *The Annals of Applied Statistics*, **16**(4), 2523-2553.

J. Trinka, H. Haghbin, M. Maadooliat and H. L. Shang (2023) "Functional time series forecasting: Functional singular spectrum analysis approaches", *Stat*, **12**(1), e621.

D. Li, P. M. Robinson and H. L. Shang (2023) "Nonstationary fractionally integrated functional time series", *Bernoulli*, **29**(2), 1505-1526.

X. Huang and H. L. Shang (2023) "Nonlinear autocorrelation function of functional time series", *Nonlinear Dynamics: An International Journal of Nonlinear Dynamics and Chaos in Engineering Systems*, **111**, 2537-2554.

H. L. Shang (2023) "Sieve bootstrapping memory parameter in long-range dependent stationary functional time series", *AStA Advances in Statistical Analysis*, **107**, 421-441.

E. Paparoditis and H. L. Shang (2023) "Bootstrap prediction bands for functional time series", *Journal of the American Statistical Association: Theory and Methods*, **118**(542), 972-986.

Y. Gao, H. L. Shang and Y. Yang (2024) "Factor-augmented smoothing model for functional data", *Statistica Sinica*, **34**(1), 1-26.

M. Mutis, U. Beyaztas, G. G. Simsek, H. L. Shang and Z. M. Yaseen (2024) Development of functional quantile autoregressive model for river flow curve forecasting, *Earth and Space Science*, **11**(8), e2024EA003564.

W-K. Seo and H. L. Shang (2024) Fractional curve time series with cointegration, *Electronic Journal of Statistics*, **18**(2), 3858-3902.

H. L. Shang (2026) Change-point detection in functional time series: Applications to age-specific mortality and fertility, *Annals of Operations Research*, **363**, 1113-1131.

Y. Yang and H. L. Shang (2025) Nonstationary functional time series forecasting, *Journal of Forecasting*, **44**(4), 1347-1362.

H. L. Shang and I. Martinez Hernandez (2025) "Forecasting intraday particle number size distribution: A functional time series approach", *Environmental and Ecological Statistics*, **32**, 1313-1338.

A. Lopez-Oriona, Y. Sun and H. L. Shang (2025) Dependence-based clustering of functional time series, *Journal of Computational and Graphical Statistics*, **34**(4), 1729-1741.

W-K. Seo and H. L. Shang (2026) "Testing for integer integration of functional time series", *Journal of the American Statistical Association: Theory and Methods*, **in press**.

################################## # References in Population Studies ##################################

H. L. Shang, H. Booth and R. J. Hyndman (2011) "Point and interval forecasts of mortality rates and life expectancy: a comparison of ten principal component methods", *Demographic Research*, **25**(5), 173-214.

H. L. Shang (2012) "Point and interval forecasts of age-specific fertility rates: a comparison of functional principal component methods", *Journal of Population Research*, **29**(3), 249-267.

H. L. Shang (2012) "Point and interval forecasts of age-specific life expectancies: a model averaging", *Demographic Research*, **27**, 593-644.

H. L. Shang, A. Wisniowski, J. Bijak, P. W. F. Smith and J. Raymer (2014) "Bayesian functional models for population forecasting", in M. Marsili and G. Capacci (eds), Proceedings of the Sixth Eurostat/UNECE Work Session on Demographic Projections, Istituto nazionale di statistica, Rome, pp. 313-325.

H. L. Shang (2015) "Selection of the optimal Box-Cox transformation parameter for modelling and forecasting age-specific fertility", *Journal of Population Research*, **32**(1), 69-79.

H. L. Shang (2015) "Forecast accuracy comparison of age-specific mortality and life expectancy: Statistical tests of the results", *Population Studies*, 69(3), 317-335.

H. L. Shang, P. W. F. Smith, J. Bijak, A. Wisniowski (2016) "A multilevel functional data method for forecasting population, with an application to the United Kingdom", *International Journal of Forecasting*, 32(3), 629-649.

H. L. Shang (2017) "Reconciling forecasts of infant mortality rates at national and sub-national levels: Grouped time-series method", *Population Research and Policy Review*, 36(1), 55-84.

R. J. Hyndman, Y. Zeng and H. L. Shang (2021) "Forecasting the old-age dependency ratio to determine the best pension age", *Australian and New Zealand Journal of Statistics*, **63**(2), 241-256.

Y. Yang and H. L. Shang (2022) "Is the group structure important in grouped functional time series?", *Journal of Data Science*, **20**(3), 303-324.

H. L. Shang and Y. Yang (2022) "Forecasting Australian subnational age-specific mortality rates", *Journal of Population Research*, **38**, 1-24.

Y. Yang, H. L. Shang and J. Raymer (2024) "Forecasting Australian fertility by age, region, and birthplace", *International Journal of Forecasting*, **40**(2), 532-548.

H. L. Shang (2025) "Bootstrap prediction intervals for the age distribution of life-table deaths", *Mathematical Population Studies*, **32**(3), 166-181.

################################## # References in Actuarial Studies ##################################

H. L. Shang and S. Haberman (2017) "Grouped multivariate and functional time series forecasting: An application to annuity pricing", Presented at the Living to 100 Symposium, Orlando, Florida, January 4-6, 2017.

H. L. Shang and S. Haberman (2017) "Grouped multivariate and functional time series forecasting: An application to annuity pricing", *Insurance: Mathematics and Economics*, **75**, 166-179.

H. L. Shang and S. Haberman (2018) "Model confidence sets and forecast combination: An application to age-specific mortality", *Genus - Journal of Population Sciences*, **74**, Article number: 19.

H. L. Shang and S. Haberman (2020) "Forecasting multiple functional time series in a group structure: an application to mortality", *ASTIN Bulletin*, **50**(2), 357-379.

H. L. Shang (2020) "Dynamic principal component regression for forecasting functional time series in a group structure", *Scandinavian Actuarial Journal*, **2020**(4), 307-322.

H. L. Shang and S. Haberman (2020) "Forecasting age distribution of death counts: An application to annuity pricing", *Annals of Actuarial Science*, **14**(1), 150-169.

H. L. Shang, S. Haberman and R. Xu (2022) "Multi-population modelling and forecasting age-specific life-table death counts", *Insurance: Mathematics and Economics*, **106**, 239-253.

H. L. Shang and S. Haberman (2024) Weighted compositional data analysis for modeling and forecasting life-table death counts, *Journal of Forecasting*, **43**(8), 3051-3071.

S. Wang, H. L. Shang, L. Tickle and H. Li (2024) Forecasting age- and sex-specific survival functions: Application to annuity pricing, *Risks*, **12**(7), 117.

H. L. Shang and S. Haberman (2025) Forecasting age distribution of life-table death counts via alpha-transformation, *Scandinavian Actuarial Journal*, **2025**(4), 387-403.

H. L. Shang and S. Haberman (2025) Forecasting age distribution of deaths: Cumulative distribution function transformation, *Insurance: Mathematics and Economics*, **122**, 249-261.

H. L. Shang and S. Haberman (2026) Constructing prediction intervals for the age distribution of deaths, *Scandinavian Actuarial Journal*, **2026**(5), 469-486.

######################## # References in Finance ########################

F. Kearney and H. L. Shang (2020) "Uncovering predictability in the evolution of the WTI oil futures curve", *European Financial Management*, **26**(1), 238-257.

H. L. Shang, K. Ji and U. Beyaztas (2021) "Granger causality of bivariate stationary curve time series", *Journal of Forecasting*, **40**(4), 626-635.

S. Butler, P. Kokoszka, H. Miao and H. L. Shang (2021) "Neural network prediction of crude oil futures using B-splines", *Energy Economics*, **94**, 105080.

H. L. Shang and F. Kearney (2022) "Dynamic functional time series forecasts of foreign exchange implied volatility surfaces", *International Journal of Forecasting*, **38**(3), 1025-1049.

H. L. Shang and K. Ji (2023) "Forecasting intraday financial time series with sieve bootstrapping and dynamic updating", *Journal of Forecasting*, **42**(8), 1973-1988.

X. Huang, H. L. Shang and K. Siu (2025) An IID test for functional time series with applications to high-frequency VIX index data, *Risks*, **13**(2), 25.

U. Beyaztas, K. Ji, H. L. Shang and E. Wu (2025) "Stock return prediction based on a functional capital asset pricing model", *Journal of Forecasting*, **44**(6), 2017-2036.

`centre` *Mean function, variance function, median function, trimmed mean function of functional data*

Description

Mean function, variance function, median function, trimmed mean function of functional data

Usage

```
centre(x, type)
```

Arguments

x	An object of class matrix.
type	Mean, variance, median, or trimmed mean?

Value

Return mean function, variance function, median function, or trimmed mean function.

Author(s)

Han Lin Shang

See Also

pcscorebootstrapdata, mean.fts, median.fts, sd.fts, var.fts

Examples

```
# The mean function is often removed in functional principal component analysis.
# The trimmed mean function is sometimes employed for robustness in the presence of outliers.
# In calculating the trimmed mean function, several functional depth measures were employed.
centre(x = ElNino_ERSST_region_1and2$y, type = "mean")
centre(x = ElNino_ERSST_region_1and2$y, type = "var")
centre(x = ElNino_ERSST_region_1and2$y, type = "median")
centre(x = ElNino_ERSST_region_1and2$y, type = "trimmed")
```

CoDa_BayesNW *Compositional data analytic approach and nonparametric function-on-function regression for forecasting density*

Description

Log-ratio transformation from constrained space to unconstrained space, where a standard nonparametric function-on-function regression can be applied.

Usage

```
CoDa_BayesNW(data, normalization, m = 5001,
 band_choice = c("Silverman", "DPI"),
 kernel = c("gaussian", "epanechnikov"))
```

Arguments

data	Densities or raw data matrix of dimension N by p, where N denotes sample size and p denotes dimensionality
normalization	Logical. If TRUE, the closure operation is applied so that each transformed curve sums to one. Default is TRUE.
m	Grid points within the data range
band_choice	Selection of optimal bandwidth
kernel	Type of kernel function

Details

1) Compute the geometric mean function 2) Apply the centered log-ratio transformation 3) Apply a nonparametric function-on-function regression to the transformed data 4) Transform forecasts back to the compositional data 5) Add back the geometric means to obtain the forecasts of the density function

Value

Out-of-sample density forecasts

Author(s)

Han Lin Shang

References

Egozcue, J. J., Diaz-Barrero, J. L. and Pawlowsky-Glahn, V. (2006) 'Hilbert space of probability density functions based on Aitchison geometry', *Acta Mathematica Sinica*, **22**, 1175-1182.

Ferraty, F. and Shang, H. L. (2025) 'Nonparametric density-on-density regression', arXiv, https://arxiv.org/abs/2503.22904.

See Also

CoDa_FPCA

Examples

```
## Not run:
CoDa_BayesNW(data = DJI_return, normalization = TRUE,
  band_choice = "DPI", kernel = "epanechnikov")

## End(Not run)
```

CoDa_FPCA *Compositional data analytic approach and functional principal component analysis for forecasting density*

Description

Log-ratio transformation from constrained space to unconstrained space, where a standard functional principal component analysis can be applied.

Usage

```
CoDa_FPCA(data, normalization, h_scale = 1, m = 5001,
 band_choice = c("Silverman", "DPI"),
 kernel = c("gaussian", "epanechnikov"),
 varprop = 0.99, fmethod)
```

Arguments

data	Densities or raw data matrix of dimension n by p, where n denotes sample size and p denotes dimensionality
normalization	Should standardization be performed?
h_scale	Scaling parameter in the kernel density estimator
m	Grid point within the data range
band_choice	Selection of optimal bandwidth
kernel	Type of kernel functions
varprop	Proportion of variance explained
fmethod	Univariate time series forecasting method

Details

1) Compute the geometric mean function 2) Apply the centered log-ratio transformation 3) Apply FPCA to the transformed data 4) Forecast principal component scores 5) Transform forecasts back to the compositional data 6) Add back the geometric means to obtain the forecasts of the density function

Value

Out-of-sample forecast densities

Author(s)

Han Lin Shang

References

Bergeron-Boucher, M.-P., Canudas-Romo, V., Oeppen, J. and Vaupel, J. W. (2017) 'Coherent forecasts of mortality with compositional data analysis', *Demographic Research*, **37**, 527-566.

Egozcue, J. J., Diaz-Barrero, J. L. and Pawlowsky-Glahn, V. (2006) 'Hilbert space of probability density functions based on Aitchison geometry', *Acta Mathematica Sinica*, **22**, 1175-1182.

See Also

Horta_Ziegelmann_FPCA, LQDT_FPCA, skew_t_fun

Examples

```
## Not run:
CoDa_FPCA(data = DJI_return, normalization = "TRUE", band_choice = "DPI",
 kernel = "epanechnikov", varprop = 0.9, fmethod = "ETS")

## End(Not run)
```

---

diff.fts *Differences of a functional time series*

Description

Computes differences of an `fts` object for each variable.

Usage

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

Arguments

`x`	An object of class `fts`.
`lag`	An integer indicating which lag to use.
`differences`	An integer indicating the order of the difference.
`...`	Other arguments.

Value

An object of class `fts`.

Author(s)

Rob J Hyndman

Examples

```
# ElNino is an object of sliced functional time series.
# Differencing is sometimes used to achieve stationarity.
diff(x = ElNino_ERSST_region_1and2)
```

`DJI_return` *Dow Jones Industrial Average (DJIA)*

Description

Dow Jones Industrial Average (DJIA) is a stock market index that shows how 30 large publicly owned companies based in the United States have traded during a standard NYSE trading session. We consider monthly cross-sectional returns from April 2004 to December 2017. The data were obtained from the CRSP (Center for Research in Security Prices) database.

Usage

```
data("DJI_return")
```

Format

A data matrix

References

Kokoszka, P., Miao, H., Petersen, A. and Shang, H. L. (2019) 'Forecasting of density functions with an application to cross-sectional and intraday returns', *International Journal of Forecasting*, **35**(4), 1304-1317.

Examples

```
data(DJI_return)
```

`dynamic_FLR` *Dynamic updates via functional linear regression*

Description

A functional linear regression is used to address the problem of dynamic updating when partial data in the most recent curve are observed.

Usage

```
dynamic_FLR(dat, newdata, holdoutdata, order_k_percent = 0.9, order_m_percent = 0.9,
    pcd_method = c("classical", "M"), robust_lambda = 2.33, bootrep = 100,
     pointfore, level = 80)
```

Arguments

dat	An object of class sfts.
newdata	A data vector of newly arrived observations.
holdoutdata	A data vector of holdout sample to evaluate point forecast accuracy.
order_k_percent	Select the number of components that explain at least 90 percent of the total variation.
order_m_percent	Select the number of components that explain at least 90 percent of the total variation.
pcd_method	Method to use for principal components decomposition. Possibilities are "classical" and "M".
robust_lambda	Tuning parameter in the two-step robust functional principal component analysis, when pcd_method = "M".
bootrep	Number of bootstrap samples.
pointfore	If pointfore = TRUE, point forecasts are produced.
level	Nominal coverage probability.

Details

This function is designed to dynamically update point and interval forecasts when partial data in the most recent curve are observed.

Value

update_forecast	Updated forecasts.
holdoutdata	Holdout sample.
err	Forecast errors.
order_k	Number of principal components in the first block of functions.
order_m	Number of principal components in the second block of functions.
update_comb	Bootstrapped forecasts for the dynamically updating time period.
update_comb_lb_ub	By taking corresponding quantiles, obtain lower and upper prediction bounds.
err_boot	Bootstrapped in-sample forecast error for the dynamically updating time period.

Author(s)

Han Lin Shang

References

H. Shen and J. Z. Huang (2008) "Interday forecasting and intraday updating of call center arrivals", *Manufacturing and Service Operations Management*, **10**(3), 391-410.

H. Shen (2009) "On modeling and forecasting time series of curves", *Technometrics*, **51**(3), 227-238.

H. L. Shang and R. J. Hyndman (2011) "Nonparametric time series forecasting with dynamic updating", Mathematics and Computers in Simulation, **81**(7), 1310-1324.

J-M. Chiou (2012) "Dynamical functional prediction and classification with application to traffic flow prediction", *Annals of Applied Statistics*, **6**(4), 1588-1614.

H. L. Shang (2013) "Functional time series approach for forecasting very short-term electricity demand", *Journal of Applied Statistics*, **40**(1), 152-168.

H. L. Shang (2017) "Forecasting Intraday S&P 500 Index Returns: A Functional Time Series Approach", *Journal of Forecasting*, **36**(7), 741-755.

H. L. Shang (2017) "Functional time series forecasting with dynamic updating: An application to intraday particulate matter concentration", *Econometrics and Statistics*, **1**, 184-200.

See Also

dynupdate

Examples

```
dynamic_FLR_point = dynamic_FLR(dat = ElNino_ERSST_region_1and2$y[,1:68],
 newdata = ElNino_ERSST_region_1and2$y[1:4,69],
 holdoutdata = ElNino_ERSST_region_1and2$y[5:12,69], pointfore = TRUE)

dynamic_FLR_interval = dynamic_FLR(dat = ElNino_ERSST_region_1and2$y[,1:68],
 newdata = ElNino_ERSST_region_1and2$y[1:4,69],
 holdoutdata = ElNino_ERSST_region_1and2$y[5:12,69], pointfore = FALSE)
```

---

dynupdate *Dynamic updates via BM, OLS, RR and PLS methods*

Description

Four methods, namely block moving (BM), ordinary least squares (OLS) regression, ridge regression (RR), and penalized least squares (PLS) regression were proposed to address the problem of dynamic updating, when partial data in the most recent curve are observed.

Usage

```
dynupdate(data, newdata = NULL, holdoutdata, method = c("ts", "block",
 "ols", "pls", "ridge"), fmethod = c("arima", "ar", "ets", "ets.na",
  "rwdrift", "rw"), pcdmethod = c("classical", "M", "rapca"),
   ngrid = max(1000, ncol(data$y)), order = 6,
```

```
robust_lambda = 2.33, lambda = 0.01, value = FALSE,
 interval = FALSE, level = 80,
  pimethod = c("parametric", "nonparametric"), B = 1000)
```

Arguments

data	An object of class sfts.
newdata	A data vector of newly arrived observations.
holdoutdata	A data vector of holdout sample to evaluate point forecast accuracy.
method	Forecasting methods. The latter four can dynamically update point forecasts.
fmethod	Univariate time series forecasting methods used in method = "ts" or method = "block".
pcdmethod	Method to use for principal components decomposition. Possibilities are "M", "rapca" and "classical".
ngrid	Number of grid points to use in calculations. Set to maximum of 1000 and ncol(data$y).
order	Number of principal components to fit.
robust_lambda	Tuning parameter in the two-step robust functional principal component analysis, when pcdmethod = "M".
lambda	Penalty parameter used in method = "pls" or method = "ridge".
value	When value = TRUE, returns forecasts or when value = FALSE, returns forecast errors.
interval	When interval = TRUE, produces distributional forecasts.
level	Nominal coverage probability.
pimethod	Parametric or nonparametric method to construct prediction intervals.
B	Number of bootstrap samples.

Details

This function is designed to dynamically update point and interval forecasts when partial data in the most recent curve are observed.

If pcdmethod = "classical", then standard functional principal component decomposition is used, as described by Ramsay and Dalzell (1991).

If pcdmethod = "rapca", then the robust principal component algorithm of Hubert, Rousseeuw and Verboven (2002) is used.

If pcdmethod = "M", then the hybrid algorithm of Hyndman and Ullah (2005) is used.

Value

forecasts	An object of class fts containing the dynamic updated point forecasts.
bootsamp	An object of class fts containing the bootstrapped point forecasts, which are updated by the PLS method.
low	An object of class fts containing the lower bound of prediction intervals.
up	An object of class fts containing the upper bound of prediction intervals.

Author(s)

Han Lin Shang

References

J. O. Ramsay and C. J. Dalzell (1991) "Some tools for functional data analysis (with discussion)", *Journal of the Royal Statistical Society: Series B*, **53**(3), 539-572.

M. Hubert and P. J. Rousseeuw and S. Verboven (2002) "A fast robust method for principal components with applications to chemometrics", *Chemometrics and Intelligent Laboratory Systems*, **60**(1-2), 101-111.

R. J. Hyndman and M. S. Ullah (2007) "Robust forecasting of mortality and fertility rates: A functional data approach", *Computational Statistics and Data Analysis*, **51**(10), 4942-4956.

H. Shen and J. Z. Huang (2008) "Interday forecasting and intraday updating of call center arrivals", *Manufacturing and Service Operations Management*, **10**(3), 391-410.

H. Shen (2009) "On modeling and forecasting time series of curves", *Technometrics*, **51**(3), 227-238.

H. L. Shang and R. J. Hyndman (2011) "Nonparametric time series forecasting with dynamic updating", Mathematics and Computers in Simulation, **81**(7), 1310-1324.

H. L. Shang (2013) "Functional time series approach for forecasting very short-term electricity demand", *Journal of Applied Statistics*, **40**(1), 152-168.

H. L. Shang (2017) "Forecasting Intraday S&P 500 Index Returns: A Functional Time Series Approach", *Journal of Forecasting*, **36**(7), 741-755.

H. L. Shang (2017) "Functional time series forecasting with dynamic updating: An application to intraday particulate matter concentration", *Econometrics and Statistics*, **1**, 184-200.

See Also

`ftsm`, `forecast.ftsm`, `plot.fm`, `residuals.fm`, `summary.fm`

Examples

```
# ElNino is an object of sliced functional time series, constructed from a univariate time series.
# When we observe some newly arrived information in the most recent time period, this function
# allows us to update the point and interval forecasts for the remaining time period.
dynupdate(data = ElNino_ERSST_region_1and2, newdata = ElNino_ERSST_region_1and2$y[1:4,69],
 holdoutdata = ElNino_ERSST_region_1and2$y[5:12,57], method = "block", interval = FALSE)
```

`error` *Forecast error measure*

Description

Computes the forecast error measure.

Usage

```
error(forecast, forecastbench, true, insampletrue, method = c("me", "mpe", "mae",
 "mse", "sse", "rmse", "mdae", "mdse", "mape", "mdape", "smape",
  "smdape", "rmspe", "rmdspe", "mrae", "mdrae", "gmrae",
   "relmae", "relmse", "mase", "mdase", "rmsse"), giveall = FALSE)
```

Arguments

forecast	Out-of-sample forecasted values.
forecastbench	Forecasted values using a benchmark method, such as random walk.
true	Out-of-sample holdout values.
insampletrue	In-sample values.
method	Method of forecast error measure.
giveall	If giveall = TRUE, all error measures are provided.

Details

Bias measure:

If method = "me", the forecast error measure is mean error.

If method = "mpe", the forecast error measure is mean percentage error.

Forecast accuracy error measure:

If method = "mae", the forecast error measure is mean absolute error.

If method = "mse", the forecast error measure is mean squared error.

If method = "sse", the forecast error measure is sum of squared errors.

If method = "rmse", the forecast error measure is root mean square error.

If method = "mdae", the forecast error measure is median absolute error.

If method = "mape", the forecast error measure is mean absolute percentage error.

If method = "mdape", the forecast error measure is median absolute percentage error.

If method = "rmspe", the forecast error measure is root mean square percentage error.

If method = "rmdspe", the forecast error measure is root median square percentage error.

Forecast accuracy symmetric error measure:

If method = "smape", the forecast error measure is symmetric mean absolute percentage error.

If method = "smdape", the forecast error measure is symmetric median absolute percentage error.

Forecast accuracy relative error measure:

If method = "mrae", the forecast error measure is mean relative absolute error.

If method = "mdrae", the forecast error measure is median relative absolute error.

If method = "gmrae", the forecast error measure is geometric mean relative absolute error.

If method = "relmae", the forecast error measure is relative mean absolute error.

If method = "relmse", the forecast error measure is relative mean square error.

Forecast accuracy scaled error measure:

If `method = "mase"`, the forecast error measure is mean absolute scaled error.

If `method = "mdase"`, the forecast error measure is median absolute scaled error.

If `method = "rmsse"`, the forecast error measure is root mean square scaled error.

Value

A numeric value.

Author(s)

Han Lin Shang

References

P. A. Thompson (1990) "An MSE statistic for comparing forecast accuracy across series", *International Journal of Forecasting*, **6**(2), 219-227.

C. Chatfield (1992) "A commentary on error measures", *International Journal of Forecasting*, **8**(1), 100-102.

S. Makridakis (1993) "Accuracy measures: theoretical and practical concerns", *International Journal of Forecasting*, **9**(4), 527-529.

R. J. Hyndman and A. Koehler (2006) "Another look at measures of forecast accuracy", *International Journal of Forecasting*, **22**(3), 443-473.

Examples

```
# Forecast error measures can be categorized into three groups: (1) scale-dependent,
# (2) scale-independent but with possible zero denominator,
# (3) scale-independent with non-zero denominator.
error(forecast = 1:2, true = 3:4, method = "mae")
error(forecast = 1:5, forecastbench = 6:10, true = 11:15, method = "mrae")
error(forecast = 1:5, forecastbench = 6:10, true = 11:15, insampletrue = 16:20,
 giveall = TRUE)
```

ER_GR *Selection of the number of principal components*

Description

Eigenvalue ratio and growth ratio

Usage

```
ER_GR(data)
```

Arguments

`data`	An n by p matrix, where n denotes sample size and p denotes the number of discretized data points in a curve

Value

k_ER	The number of components selected by the eigenvalue ratio
k_GR	The number of components selected by the growth ratio

Author(s)

Han Lin Shang

References

Lam, C. and Yao, Q. (2012). Factor modelling for high-dimensional time series: Inference for the number of factors. *The Annals of Statistics*, **40**, 694-726.

Ahn, S. and Horenstein, A. (2013). Eigenvalue ratio test for the number of factors. *Econometrica*, **81**, 1203-1227.

See Also

ftsm

Examples

```
ER_GR(pm_10_GR$y)
```

---

extract *Extract variables or observations*

Description

Creates subsets of an fts object.

Usage

```
extract(data, direction = c("time", "x"), timeorder, xorder)
```

Arguments

data	An object of class fts.
direction	In time direction or x variable direction?
timeorder	Indexes of time order.
xorder	Indexes of x variable order.

Value

When xorder is specified, it returns an fts object with the same arguments as data but with a subset of x variables.

When timeorder is specified, it returns an fts object with the same arguments as data but with a subset of time variables.

Author(s)

Han Lin Shang

Examples

```
# ElNino is an object of class sliced functional time series.
# This function truncates the data series rowwise or columnwise.
extract(data = ElNino_ERSST_region_1and2, direction = "time",
 timeorder = 1980:2006) # Last 27 curves
extract(data = ElNino_ERSST_region_1and2, direction = "x",
 xorder = 1:8) # First 8 x variables
```

---

facf *Functional autocorrelation function*

Description

Compute functional autocorrelation function at various lags

Usage

```
facf(fun_data, lag_value_range = seq(0, 20, by = 1))
```

Arguments

fun_data	A data matrix of dimension (n by p), where n denotes sample size; and p denotes dimensionality
lag_value_range	Lag value

Details

The autocovariance at lag i is estimated by the function $\widehat{\gamma}_i(t,s)$, a functional analog of the autocorrelation is defined as

$$\widehat{\rho}_i = \frac{\|\widehat{\gamma}_i\|}{\int \widehat{\gamma}_0(t,t)dt}.$$

Value

A vector of functional autocorrelation function at various lags

Author(s)

Han Lin Shang

References

L. Horv\'ath, G. Rice and S. Whipple (2016) Adaptive bandwidth selection in the long run covariance estimator of functional time series, *Computational Statistics and Data Analysis*, **100**, 676-693.

Examples

```
facf_value = facf(fun_data = t(ElNino_ERSST_region_1and2$y))
```

`farforecast` *Functional data forecasting through functional principal component autoregression*

Description

The coefficients from the fitted object are forecasted using a multivariate time-series forecasting method. The forecast coefficients are then multiplied by the functional principal components to obtain a forecast curve.

Usage

```
farforecast(object, h = 10, var_type = "const", Dmax_value, Pmax_value,
 level = 80, PI = FALSE)
```

Arguments

`object`	An object of `fds`.
`h`	Forecast horizon.
`var_type`	Type of multivariate time series forecasting method; see `VAR` for details.
`Dmax_value`	Maximum number of components considered.
`Pmax_value`	Maximum order of VAR model considered.
`level`	Nominal coverage probability of prediction error bands.
`PI`	When `PI = TRUE`, a prediction interval will be given along with the point forecast.

Details

1. Decompose the smooth curves via a functional principal component analysis (FPCA).

2. Fit a multivariate time-series model to the principal component score matrix.

3. Forecast the principal component scores using the fitted multivariate time-series models. The order of VAR is selected optimally via an information criterion.

4. Multiply the forecast principal component scores by estimated principal components to obtain forecasts of $f_{n+h}(x)$.

5. Prediction intervals are constructed by taking quantiles of the one-step-ahead forecast errors.

Value

`point_fore`	Point forecast
`order_select`	Selected VAR order and number of components
`PI_lb`	Lower bound of a prediction interval
`PI_ub`	Upper bound of a prediction interval

Author(s)

Han Lin Shang

References

A. Aue, D. D. Norinho and S. Hormann (2015) "On the prediction of stationary functional time series", *Journal of the American Statistical Association*, **110**(509), 378-392.

J. Klepsch, C. Kl\"uppelberg and T. Wei (2017) "Prediction of functional ARMA processes with an application to traffic data", *Econometrics and Statistics*, **1**, 128-149.

See Also

`forecast.ftsm`, `forecastfplsr`

Examples

```
sqrt_pm10 = sqrt(pm_10_GR$y)
multi_forecast_sqrt_pm10 = farforecast(object = fts(seq(0, 23.5, by = 0.5), sqrt_pm10),
 h = 1, Dmax_value = 5, Pmax_value = 3)
```

---

fbootstrap *Bootstrap independent and identically distributed functional data*

Description

Computes bootstrap or smoothed bootstrap samples based on independent and identically distributed functional data.

Usage

```
fbootstrap(data, estad = func.mean, alpha = 0.05, nb = 200, suav = 0,
 media.dist = FALSE, graph = FALSE, ...)
```

Arguments

`data`	An object of class `fds` or `fts`.
`estad`	Estimate function of interest. Default is to estimate the mean function. Other options are `func.mode` or `func.var`.
`alpha`	Significance level used in the smooth bootstrapping.
`nb`	Number of bootstrap samples.
`suav`	Smoothing parameter.
`media.dist`	Estimate mean function.
`graph`	Graphical output.
`...`	Other arguments.

Value

A list containing the following components is returned.

`estimate`	Estimate function.
`max.dist`	Max distance of bootstrap samples.
`rep.dist`	Distances of bootstrap samples.
`resamples`	Bootstrap samples.
`center`	Functional mean.

Author(s)

Han Lin Shang

References

A. Cuevas and M. Febrero and R. Fraiman (2006), "On the use of the bootstrap for estimating functions with functional data", *Computational Statistics and Data Analysis*, **51**(2), 1063-1074.

A. Cuevas and M. Febrero and R. Fraiman (2007), "Robust estimation and classification for functional data via projection-based depth notions", *Computational Statistics*, **22**(3), 481-496.

M. Febrero and P. Galeano and W. Gonzalez-Manteiga (2007) "A functional analysis of NOx levels: location and scale estimation and outlier detection", *Computational Statistics*, **22**(3), 411-427.

M. Febrero and P. Galeano and W. Gonzalez-Manteiga (2008) "Outlier detection in functional data by depth measures, with application to identify abnormal NOx levels", *Environmetrics*, **19**(4), 331-345.

M. Febrero and P. Galeano and W. Gonzalez-Manteiga (2010) "Measures of influence for the functional linear model with scalar response", *Journal of Multivariate Analysis*, **101**(2), 327-339.

J. A. Cuesta-Albertos and A. Nieto-Reyes (2010) "Functional classification and the random Tukey depth. Practical issues", Combining Soft Computing and Statistical Methods in Data Analysis, *Advances in Intelligent and Soft Computing*, **77**, 123-130.

D. Gervini (2012) "Outlier detection and trimmed estimation in general functional spaces", *Statistica Sinica*, **22**(4), 1639-1660.

H. L. Shang (2015) "Re-sampling techniques for estimating the distribution of descriptive statistics of functional data", *Communications in Statistics–Simulation and Computation*, **44**(3), 614-635.

H. L. Shang (2018) Bootstrap methods for stationary functional time series, *Statistics and Computing*, **28**(1), 1-10.

See Also

`pcscorebootstrapdata`

Examples

```
# Bootstrapping the distribution of a summary statistics of functional data.
fbootstrap(data = ElNino_ERSST_region_1and2)
```

`forecast.ftsm` *Forecast functional time series via functional principal component analysis*

Description

The coefficients from the fitted object are forecasted using either an ARIMA model (`method = "arima"`), an AR model (`method = "ar"`), an exponential smoothing method (`method = "ets"`), a linear exponential smoothing method allowing missing values (`method = "ets.na"`), or a random walk with drift model (`method = "rwdrift"`). The forecast coefficients are then multiplied by the principal components to obtain a forecast curve.

Usage

```
## S3 method for class 'ftsm'
forecast(object, h = 10, method = c("ets", "arima", "ar", "ets.na",
 "rwdrift", "rw", "struct", "arfima"), level = 80, jumpchoice = c("fit",
  "actual"), pimethod = c("parametric", "nonparametric"), B = 100,
   usedata = nrow(object$coeff), adjust = TRUE, model = NULL,
    damped = NULL, stationary = FALSE, ...)
```

Arguments

`object`	Output from `ftsm`.
`h`	Forecast horizon.
`method`	Univariate time series forecasting methods. Current possibilities are "ets", "arima", "ets.na", "rwdrift", "ar", "struct", "arfima", and "rw".
`level`	Coverage probability of prediction intervals.
`jumpchoice`	Jump-off point for forecasts. Possibilities are "actual" and "fit". If "actual", the forecasts are bias-adjusted by the difference between the fit and the last year of observed data. Otherwise, no adjustment is used. See Booth et al. (2006) for details on the jump-off point.
`pimethod`	Indicates if parametric method is used to construct prediction intervals.
`B`	Number of bootstrap samples.
`usedata`	Number of time periods to use in forecasts. Default is to use all.
`adjust`	If `adjust = TRUE`, adjusts the variance so that the one-step forecast variance matches the empirical one-step forecast variance.
`model`	If the `ets` method is used, `model` allows a model specification to be passed to `ets()`.
`damped`	If the `ets` method is used, `damped` allows the damping specification to be passed to `ets()`.
`stationary`	If `stationary = TRUE`, `method` is set to `method = "ar"` and only stationary AR models are used.
`...`	Other arguments passed to the forecast routine.

Details

1. Obtain a smooth curve $f_t(x)$ for each t using a nonparametric smoothing technique.

2. Decompose the smooth curves via a functional principal component analysis.

3. Fit a univariate time series model to each of the principal component scores.

4. Forecast the principal component scores using the fitted time series models.

5. Multiply the forecast principal component scores by fixed principal components to obtain forecasts of $f_{n+h}(x)$.

6. The estimated variances of the error terms (smoothing error and model residual error) are used to compute prediction intervals for the forecasts.

Value

List with the following components:

`mean`	An object of class `fts` containing point forecasts.
`lower`	An object of class `fts` containing lower bound for prediction intervals.
`upper`	An object of class `fts` containing upper bound for prediction intervals.
`fitted`	An object of class `fts` of one-step-ahead forecasts for historical data.
`error`	An object of class `fts` of one-step-ahead errors for historical data.
`coeff`	List of objects of type `forecast` containing the coefficients and their forecasts.
`coeff.error`	One-step-ahead forecast errors for each of the coefficients.
`var`	List containing the various components of variance: model, error, mean, total and coeff.
`model`	Fitted `ftsm` model.
`bootsamp`	An array of $dimension = c(p, B, h)$ containing the bootstrapped point forecasts. p is the number of variables. B is the number of bootstrap samples. h is the forecast horizon.

Author(s)

Rob J Hyndman

References

H. Booth and R. J. Hyndman and L. Tickle and P. de Jong (2006) "Lee-Carter mortality forecasting: A multi-country comparison of variants and extensions", *Demographic Research*, **15**, 289-310.

B. Erbas and R. J. Hyndman and D. M. Gertig (2007) "Forecasting age-specific breast cancer mortality using functional data model", *Statistics in Medicine*, **26**(2), 458-470.

R. J. Hyndman and M. S. Ullah (2007) "Robust forecasting of mortality and fertility rates: A functional data approach", *Computational Statistics and Data Analysis*, **51**(10), 4942-4956.

R. J. Hyndman and H. Booth (2008) "Stochastic population forecasts using functional data models for mortality, fertility and migration", *International Journal of Forecasting*, **24**(3), 323-342.

R. J. Hyndman and H. L. Shang (2009) "Forecasting functional time series" (with discussion), *Journal of the Korean Statistical Society*, **38**(3), 199-221.

H. L. Shang (2013) "Functional time series approach for forecasting very short-term electricity demand", *Journal of Applied Statistics*, **40**(1), 152-168.

H. L. Shang (2013) "ftsa: An R package for analyzing functional time series", *The R Journal*, **5**(1), 64-72.

H. L. Shang, A. Wisniowski, J. Bijak, P. W. F. Smith and J. Raymer (2014) "Bayesian functional models for population forecasting", in M. Marsili and G. Capacci (eds), Proceedings of the Sixth Eurostat/UNECE Work Session on Demographic Projections, Istituto nazionale di statistica, Rome, pp. 313-325.

H. L. Shang (2015) "Selection of the optimal Box-Cox transformation parameter for modelling and forecasting age-specific fertility", *Journal of Population Research*, **32**(1), 69-79.

H. L. Shang (2015) "Forecast accuracy comparison of age-specific mortality and life expectancy: Statistical tests of the results", *Population Studies*, **69**(3), 317-335.

H. L. Shang, P. W. F. Smith, J. Bijak, A. Wisniowski (2016) "A multilevel functional data method for forecasting population, with an application to the United Kingdom", *International Journal of Forecasting*, **32**(3), 629-649.

See Also

`ftsm`, `forecastfplsr`, `plot.ftsf`, `plot.fm`, `residuals.fm`, `summary.fm`

Examples

```
# ElNino is an object of class sliced functional time series.
# Via functional principal component decomposition, the dynamic was captured
# by a few principal components and principal component scores.
# By using an exponential smoothing method,
# the principal component scores are forecasted.
# The forecasted curves are constructed by forecasted principal components
# times fixed principal components plus the mean function.
forecast(object = ftsm(ElNino_ERSST_region_1and2), h = 10, method = "ets")
forecast(object = ftsm(ElNino_ERSST_region_1and2, weight = TRUE))
```

`forecastfplsr` *Forecast functional time series via functional partial least squares regression*

Description

The decentralized response is forecasted by multiplying the estimated regression coefficient by the new decentralized predictor

Usage

```
forecastfplsr(object, components, h)
```

Arguments

object	An object of class fts.
components	Number of optimal components.
h	Forecast horizon.

Value

An fts class object, containing forecasts of responses.

Author(s)

Han Lin Shang

References

R. J. Hyndman and H. L. Shang (2009) "Forecasting functional time series" (with discussion), *Journal of the Korean Statistical Society*, **38**(3), 199-221.

See Also

forecast.ftsm, ftsm, plot.fm, plot.ftsf, residuals.fm, summary.fm

Examples

```
# A set of functions is decomposed by functional partial least squares decomposition.
# By forecasting univariate partial least squares scores, the forecasted curves are
# obtained by multiplying the forecasted scores by fixed functional partial least
# squares function plus fixed mean function.
forecastfplsr(object = ElNino_ERSST_region_1and2, components = 2, h = 5)
```

---

fplsr *Functional partial least squares regression*

Description

Fits a functional partial least squares (PLSR) model using the nonlinear partial least squares (NIPALS) algorithm or simple partial least squares (SIMPLS) algorithm.

Usage

```
fplsr(data, order = 6, type = c("simpls", "nipals"), unit.weights =
 TRUE, weight = FALSE, beta = 0.1, interval = FALSE, method =
  c("delta", "boota"), alpha = 0.05, B = 100, adjust = FALSE,
   backh = 10)
```

Arguments

data	An object of class fts.
order	Number of principal components to fit.
type	When type = "nipals", uses the NIPALS algorithm; when type = "simpls", uses the SIMPLS algorithm.
unit.weights	Constrains predictor loading weights to have unit norm.
weight	When weight = TRUE, a set of geometrically decaying weights is applied to the decentralized data.
beta	When weight = TRUE, the speed of geometric decay is governed by a weight parameter.
interval	When interval = TRUE, produces distributional forecasts.
method	Method used for computing prediction intervals.
alpha	1-alpha gives the nominal coverage probability.
B	Number of replications.
adjust	When adjust = TRUE, an adjustment is performed.
backh	When adjust = TRUE, an adjustment is performed by evaluating the difference between predicted and actual values in a testing set. backh specifies the testing set.

Details

Point forecasts:

The NIPALS function implements the orthogonal scores algorithm, as described in Martens and Naes (1989). This is one of the two classical PLSR algorithms; the other is the simple partial least squares regression in DeJong (1993). The difference between these two approaches is that the NIPALS deflates the original predictors and responses, while the SIMPLS deflates the covariance matrix of original predictors and responses. Thus, SIMPLS is more computationally efficient than NIPALS.

In a functional data set, the functional PLSR can be performed by setting the functional responses to be 1 lag ahead of the functional predictors. This idea has been adopted from the Autoregressive Hilbertian processes of order 1 (ARH(1)) of Bosq (2000).

Distributional forecasts:

Parametric method:

Influenced by the works of Denham (1997) and Phatak et al. (1993), one way of constructing prediction intervals in the PLSR is via a local linearization method (also known as the Delta method). It can be easily understood as the first two terms in a Taylor series expansion. The variance of coefficient estimators can be approximated, from which analytic-formula-based prediction intervals are constructed.

Nonparametric method:

After discretizing and decentralizing functional data $f_t(x)$ and $g_s(y)$, a PLSR model with K latent components is built. Then, the fit residuals $o_s(y_i)$ between $g_s(y_i)$ and $\hat{g}_s(y_i)$ are calculated as

$$o_s(y_i) = g_s(y_i) - \hat{g}_s(y_i), i = 1, ..., p.$$

The next step is to generate B bootstrap samples $o_s^b(y_i)$ by randomly sampling with replacement from $[o_1(y_i), ..., o_n(y_i)]$. Adding bootstrapped residuals to the original response variables in order to generate new bootstrap responses,

$$g_s^b(y_i) = g_s(y_i) + o_s^b(y_i).$$

Then, the PLSR models are constructed using the centered and discretized predictors and bootstrapped responses to obtain the bootstrapped regression coefficients and point forecasts, from which the empirical prediction intervals and kernel density plots are constructed.

Value

A list containing the following components is returned.

`B`	$(p \times m)$ matrix containing the regression coefficients. p is the number of variables in the predictors and m is the number of variables in the responses.
`P`	$(p \times order)$ matrix containing the predictor loadings.
`Q`	$(m \times order)$ matrix containing the response loadings.
`T`	`(ncol(data$y)-1)` x `order` matrix containing the predictor scores.
`R`	$(p \times order)$ matrix containing the weights used to construct the latent components of predictors.
`Yscores`	`(ncol(data$y)-1)` x `order` matrix containing the response scores.
`projection`	$(p \times order)$ projection matrix used to convert predictors to predictor scores.
`meanX`	An object of class `fts` containing the column means of predictors.
`meanY`	An object of class `fts` containing the column means of responses.
`Ypred`	An object of class `fts` containing the 1-step-ahead predicted values of the responses.
`fitted`	An object of class `fts` containing the fitted values.
`residuals`	An object of class `fts` containing the regression residuals.
`Xvar`	A vector with the amount of predictor variance explained by each component.
`Xtotvar`	Total variance in predictors.
`weight`	When `weight = TRUE`, a set of geometrically decaying weights is given. When `weight = FALSE`, weights are all equal to 1.
`x1`	Time period of an `fts` object, which can be obtained from `colnames(data$y)`.
`y1`	Variables of an `fts` object, which can be obtained from `data$x`.
`ypred`	Returns the original functional predictors.
`y`	Returns the original functional responses.
`bootsamp`	Bootstrapped point forecasts.

lb	Lower bound of prediction intervals.
ub	Upper bound of prediction intervals.
lbadj	Adjusted lower bound of prediction intervals.
ubadj	Adjusted upper bound of prediction intervals.
lbadjfactor	Adjusted lower bound factor, which lies generally between 0.9 and 1.1.
ubadjfactor	Adjusted upper bound factor, which lies generally between 0.9 and 1.1.

Author(s)

Han Lin Shang

References

S. Wold and A. Ruhe and H. Wold and W. J. Dunn (1984) "The collinearity problem in linear regression. The partial least squares (PLS) approach to generalized inverses", *SIAM Journal of Scientific and Statistical Computing*, **5**(3), 735-743.

S. de Jong (1993) "SIMPLS: an alternative approach to partial least squares regression", *Chemometrics and Intelligent Laboratory Systems*, **18**(3), 251-263.

C. J. F. ter Braak and S. de Jong (1998) "The objective function of partial least squares regression", *Journal of Chemometrics*, **12**(1), 41-54.

B. Dayal and J. MacGregor (1997) "Recursive exponentially weighted PLS and its applications to adaptive control and prediction", *Journal of Process Control*, **7**(3), 169-179.

B. D. Marx (1996) "Iteratively reweighted partial least squares estimation for generalized linear regression", *Technometrics*, **38**(4), 374-381.

L. Xu and J-H. Jiang and W-Q. Lin and Y-P. Zhou and H-L. Wu and G-L. Shen and R-Q. Yu (2007) "Optimized sample-weighted partial least squares", *Talanta*, **71**(2), 561-566.

A. Phatak and P. Reilly and A. Penlidis (1993) "An approach to interval estimation in partial least squares regression", *Analytica Chimica Acta*, **277**(2), 495-501.

M. Denham (1997) "Prediction intervals in partial least squares", *Journal of Chemometrics*, **11**(1), 39-52.

D. Bosq (2000) *Linear Processes in Function Spaces*, New York: Springer.

N. Faber (2002) "Uncertainty estimation for multivariate regression coefficients", *Chemometrics and Intelligent Laboratory Systems*, **64**(2), 169-179.

J. A. Fernandez Pierna and L. Jin and F. Wahl and N. M. Faber and D. L. Massart (2003) "Estimation of partial least squares regression prediction uncertainty when the reference values carry a sizeable measurement error", *Chemometrics and Intelligent Laboratory Systems*, **65**(2), 281-291.

P. T. Reiss and R. T. Ogden (2007), "Functional principal component regression and functional partial least squares", *Journal of the American Statistical Association*, **102**(479), 984-996.

C. Preda, G. Saporta (2005) "PLS regression on a stochastic process", *Computational Statistics and Data Analysis*, **48**(1), 149-158.

C. Preda, G. Saporta, C. Leveder (2007) "PLS classification of functional data", *Computational Statistics*, **22**, 223-235.

A. Delaigle and P. Hall (2012), "Methodology and theory for partial least squares applied to functional data", *Annals of Statistics*, **40**(1), 322-352.

M. Febrero-Bande, P. Galeano, W. Gonz\'alez-Manteiga (2017), "Functional principal component regression and functional partial least-squares regression: An overview and a comparative study", *International Statistical Review*, **85**(1), 61-83.

See Also

ftsm, forecast.ftsm, plot.fm, summary.fm, residuals.fm, plot.fmres

Examples

```
# When weight = FALSE, all observations are assigned equally.
# When weight = TRUE, all observations are assigned geometrically decaying weights.
fplsr(data = ElNino_ERSST_region_1and2, order = 6, type = "nipals")
fplsr(data = ElNino_ERSST_region_1and2, order = 6)
fplsr(data = ElNino_ERSST_region_1and2, weight = TRUE)
fplsr(data = ElNino_ERSST_region_1and2, unit.weights = FALSE)
fplsr(data = ElNino_ERSST_region_1and2, unit.weights = FALSE, weight = TRUE)

# The prediction intervals are calculated numerically.
fplsr(data = ElNino_ERSST_region_1and2, interval = TRUE, method = "delta")

# The prediction intervals are calculated by bootstrap method.
fplsr(data = ElNino_ERSST_region_1and2, interval = TRUE, method = "boota")
```

---

ftsm *Fit functional time series model*

Description

Fits a principal component model to an `fts` object. The function uses optimal orthonormal principal components obtained from a principal components decomposition.

Usage

```
ftsm(y, order = 6, ngrid = max(500, ncol(y$y)), method = c("classical",
 "M", "rapca"), mean = TRUE, level = FALSE, lambda = 3,
  weight = FALSE, beta = 0.1, ...)
```

Arguments

`y`	An object of class `fts`.
`order`	Number of principal components to fit.
`ngrid`	Number of grid points to use in calculations. Set to a maximum of 500 and `ncol(y$y)`.
`method`	Method to use for principal components decomposition. Possibilities are “M”, “rapca” and “classical”.

mean	If mean = TRUE, it will estimate the mean term in the model before computing basis terms. If mean = FALSE, the mean term is assumed to be zero.
level	If level = TRUE, it will include an additional (intercept) term that depends on t but not on x.
lambda	Tuning parameter for robustness when method = "M".
weight	When weight = TRUE, a set of geometrically decaying weights is applied to the decentralized data.
beta	When weight = TRUE, the speed of geometric decay is governed by a weight parameter.
...	Additional arguments controlling the fitting procedure.

Details

If method = "classical", then standard functional principal component decomposition is used, as described by Ramsay and Dalzell (1991).

If method = "rapca", then the robust principal component algorithm of Hubert, Rousseeuw and Verboven (2002) is used.

If method = "M", then the hybrid algorithm of Hyndman and Ullah (2005) is used.

Value

Object of class "ftsm" with the following components:

x1	Time period of an fts object, which can be obtained from colnames(y$y).
y1	Variables of an fts object, which can be obtained from y$x.
y	Original functional time series or sliced functional time series.
basis	Matrix of principal components evaluated at value of y$x (one column for each principal component). The first column is the fitted mean or median.
basis2	Matrix of principal components excluded from the selected model.
coeff	Matrix of coefficients (one column for each coefficient series). The first column is all ones.
coeff2	Matrix of coefficients associated with the principal components excluded from the selected model.
fitted	An object of class fts containing the fitted values.
residuals	An object of class fts containing the regression residuals (difference between observed and fitted).
varprop	Proportion of variation explained by each principal component.
wt	Weight associated with each time period.
v	Measure of variation for each time period.
mean.se	Measure of standard error associated with the mean.

Author(s)

Rob J Hyndman

References

J. O. Ramsay and C. J. Dalzell (1991) "Some tools for functional data analysis (with discussion)", *Journal of the Royal Statistical Society: Series B*, **53**(3), 539-572.

M. Hubert and P. J. Rousseeuw and S. Verboven (2002) "A fast robust method for principal components with applications to chemometrics", *Chemometrics and Intelligent Laboratory Systems*, **60**(1-2), 101-111.

B. Erbas and R. J. Hyndman and D. M. Gertig (2007) "Forecasting age-specific breast cancer mortality using functional data model", *Statistics in Medicine*, **26**(2), 458-470.

R. J. Hyndman and M. S. Ullah (2007) "Robust forecasting of mortality and fertility rates: A functional data approach", *Computational Statistics and Data Analysis*, **51**(10), 4942-4956.

R. J. Hyndman and H. Booth (2008) "Stochastic population forecasts using functional data models for mortality, fertility and migration", *International Journal of Forecasting*, **24**(3), 323-342.

R. J. Hyndman and H. L. Shang (2009) "Forecasting functional time series (with discussion)", *Journal of the Korean Statistical Society*, **38**(3), 199-221.

See Also

`ftsmweightselect`, `forecast.ftsm`, `plot.fm`, `plot.ftsf`, `residuals.fm`, `summary.fm`

Examples

```
# ElNino is an object of class sliced functional time series, constructed
# from a univariate time series.
# By default, all observations are assigned with equal weighting.
ftsm(y = ElNino_ERSST_region_1and2, order = 6, method = "classical", weight = FALSE)
# When weight = TRUE, geometrically decaying weights are used.
ftsm(y = ElNino_ERSST_region_1and2, order = 6, method = "classical", weight = TRUE)
```

`ftsmiterativeforecasts` *Forecast functional time series via iterative functional principal component analysis*

Description

The coefficients from the fitted object are forecasted using either an ARIMA model (`method = "arima"`), an AR model (`method = "ar"`), an exponential smoothing method (`method = "ets"`), a linear exponential smoothing method allowing missing values (`method = "ets.na"`), or a random walk with drift model (`method = "rwdrift"`). The forecast coefficients are then multiplied by the principal components to obtain a forecast curve.

Usage

```
ftsmiterativeforecasts(object, components, iteration = 20)
```

Arguments

`object`	An object of class `fts`.
`components`	Number of principal components.
`iteration`	Number of iterative one-step-ahead forecasts.

Details

1. Obtain a smooth curve $f_t(x)$ for each t using a nonparametric smoothing technique.

2. Decompose the smooth curves via a functional principal component analysis.

3. Fit a univariate time series model to each of the principal component scores.

4. Forecast the principal component scores using the fitted time series models.

5. Multiply the forecast principal component scores by fixed principal components to obtain forecasts of $f_{n+h}(x)$.

6. The estimated variances of the error terms (smoothing error and model residual error) are used to compute prediction intervals for the forecasts.

Value

List with the following components:

`mean`	An object of class `fts` containing point forecasts.
`lower`	An object of class `fts` containing lower bound for prediction intervals.
`upper`	An object of class `fts` containing upper bound for prediction intervals.
`fitted`	An object of class `fts` of one-step-ahead forecasts for historical data.
`error`	An object of class `fts` of one-step-ahead errors for historical data.
`coeff`	List of objects of type `forecast` containing the coefficients and their forecasts.
`coeff.error`	One-step-ahead forecast errors for each of the coefficients.
`var`	List containing the various components of variance: model, error, mean, total and coeff.
`model`	Fitted `ftsm` model.
`bootsamp`	An array of $dim = c(p, B, h)$ containing the bootstrapped point forecasts. p is the number of variables. B is the number of bootstrap samples. h is the forecast horizon.

Author(s)

Han Lin Shang

References

H. Booth and R. J. Hyndman and L. Tickle and P. D. Jong (2006) "Lee-Carter mortality forecasting: A multi-country comparison of variants and extensions", *Demographic Research*, **15**, 289-310.

B. Erbas and R. J. Hyndman and D. M. Gertig (2007) "Forecasting age-specific breast cancer mortality using functional data model", *Statistics in Medicine*, **26**(2), 458-470.

R. J. Hyndman and M. S. Ullah (2007) "Robust forecasting of mortality and fertility rates: A functional data approach", *Computational Statistics and Data Analysis*, **51**(10), 4942-4956.

R. J. Hyndman and H. Booth (2008) "Stochastic population forecasts using functional data models for mortality, fertility and migration", *International Journal of Forecasting*, **24**(3), 323-342.

R. J. Hyndman and H. L. Shang (2009) "Forecasting functional time series" (with discussion), *Journal of the Korean Statistical Society*, **38**(3), 199-221.

See Also

`ftsm`, `plot.ftsf`, `plot.fm`, `residuals.fm`, `summary.fm`

Examples

```
# Iterative one-step-ahead forecasts via functional principal component analysis.
ftsmiterativeforecasts(object = Australiasmoothfertility, components = 2, iteration = 5)
```

`ftsmweightselect` *Selection of the weight parameter used in the weighted functional time series model*

Description

The geometrically decaying weights are used to estimate the mean curve and functional principal components, where more weight is assigned to the more recent data than to data from the distant past.

Usage

```
ftsmweightselect(data, ncomp = 6, ntestyear, errorcriterion = c("mae", "mse", "mape"))
```

Arguments

`data`	An object of class `sfts`.
`ncomp`	Number of components.
`ntestyear`	Number of holdout observations used to assess the forecast accuracy.
`errorcriterion`	Error measure.

Details

The data set is split into a fitting period and forecasting period. Using the data in the fitting period, we compute the one-step-ahead forecasts and calculate the forecast error. Then, we increase the fitting period by one, and carry out the same forecasting procedure until the fitting period covers the entire data set. The forecast accuracy is determined by the averaged forecast error across the years in the forecasting period. By using an optimization algorithm, we select the optimal weight parameter that would result in the minimum forecast error.

Value

Optimal weight parameter.

Note

Can be computationally intensive, as it takes about half a minute to compute.

Author(s)

Han Lin Shang

References

R. J. Hyndman and H. L. Shang (2009) "Forecasting functional time series (with discussion)", *Journal of the Korean Statistical Society*, **38**(3), 199-221.

See Also

`ftsm`, `forecast.ftsm`

Examples

```
ftsmweightselect(data = ElNino_ERSST_region_1and2, ncomp = 2, ntestyear = 1)
```

GAEVforecast *Fit a generalized additive extreme value model to the functional data with given basis numbers*

Description

One-step-ahead forecast for any given quantile(s) of functional time series of extreme values using a generalized additive extreme value (GAEV) model.

Usage

```
GAEVforecast(data, q, d.loc.max = 10, d.logscale.max = 10)
```

Arguments

`data`	an n by p data matrix, where n denotes the number of functional objects and p denotes the number of realizations on each functional object
`q`	a required scalar or vector of GEV quantiles that are of forecasting interest
`d.loc.max`	the maximum number of basis functions considered for the location parameter
`d.logscale.max`	the maximum number of basis functions considered for the (log-)scale parameter

Details

For the functional time series $\{X_t(u), t = 1, ..., T, u \in \mathcal{I}\}$, the GAEV model is given as

$$X_t(u)\ GEV[\mu_t(u), \sigma_t(u), \xi_t],$$

where

$$\mu_t(u) = \beta_{t,0}^{(\mu)} + \sum_{i=1}^{d_1} \beta_{t,i}^{(\mu)} b_i^{(\mu)}(u),$$

$$\ln(\sigma_t(u)) = \beta_{t,0}^{(\sigma)} + \sum_{i=1}^{d_2} \beta_{t,i}^{(\sigma)} b_i^{(\sigma)}(u), \xi_t \in [0, \infty),$$

where $d_j, j = 1, 2$ are positive integers of basis numbers, $\{b_i^{(\mu)}(u), i = 1, \ldots, d_1\}$ and $\{b_i^{(\sigma)}(u), i = 1, \ldots, d_2\}$ are the cubic regression spline basis functions.

The optimal numbers of basis functions (d_1, d_2) are chosen by minimizing the Kullback-Leibler divergence on the test set using a leave-one-out cross-validation technique.

The one-step-ahead forecast of the joint coefficients $(\widehat{\beta^{(\mu)}}_{T+1,i}, \widehat{\beta^{(\sigma)}}_{T+1,j}, \widehat{\xi}_{T+1}, i = 0, ..., d_1, j = 0, ..., d_2)$ is produced using a vector autoregressive model, whose order is selected via the corrected Akaike information criterion. Then the one-step-ahead forecast of the GEV parameter $(\widehat{\mu}_{T+1}(u), \widehat{\sigma}_{T+1}(u), \widehat{\xi}_{T+1})$ can be computed accordingly.

The one-step-ahead forecast for the τ-th quantile of the extreme values $\widehat{X}_{T+1}(u)$ is computed by

$$Q_\tau(u|\widehat{\mu}_{T+1}, \widehat{\sigma}_{T+1}, \widehat{\xi}_{T+1})$$

=

$$\widehat{\mu}_{T+1}(u) + \frac{\widehat{\sigma}_{T+1}(u)\left[(-\ln(\tau))^{-\widehat{\xi}_{T+1}} - 1\right]}{\widehat{\xi}_{T+1}}, \xi > 0, \tau \in [0, 1);\ \xi < 0, \tau \in (0, 1], \widehat{\mu}_{T+1}(u) - \widehat{\sigma}_{T+1}(u) \cdot \ln[-\ln(\tau)], \xi = 0, \tau \in$$

Value

`kdf.location`	the optimal number of basis functions considered for the location parameter
`kdf.logscale`	the optimal number of basis functions considered for the (log-)scale parameter
`basis.location`	the basis functions for the location parameter
`basis.logscale`	the basis functions for the (log-)scale parameter
`para.location.pred`	the predicted location function

`para.scale.pred`
: the predicted scale function

`para.shape.pred`
: the predicted shape parameter

`density.pred`
: the predicted density function(s) for the given quantile(s)

Author(s)

Ruofan Xu and Han Lin Shang

References

Shang, H. L. and Xu, R. (2021) 'Functional time series forecasting of extreme values', *Communications in Statistics Case Studies Data Analysis and Applications*, **7**(2), 182-199.

Examples

```
## Not run:
library(evd)
data = matrix(rgev(1000),ncol=50)
GAEVforecast(data = data, q = c(0.02,0.7), d.loc.max = 5, d.logscale.max = 5)

## End(Not run)
```

Horta_Ziegelmann_FPCA *Dynamic functional principal component analysis for density forecasting*

Description

Implementation of a dynamic functional principal component analysis to forecast densities.

Usage

```
Horta_Ziegelmann_FPCA(data, gridpoints, h_scale = 1, p = 5, m = 5001,
 kernel = c("gaussian", "epanechnikov"), band_choice = c("Silverman", "DPI"),
 VAR_type = "both", lag_maximum = 6, no_boot = 1000, alpha_val = 0.1,
 ncomp_select = TRUE, D_val = 10)
```

Arguments

`data`
: Densities or raw data matrix of dimension N by p, where N denotes sample size and p denotes dimensionality

`gridpoints`
: Grid points

`h_scale`
: Scaling parameter in the kernel density estimator

`p`
: Number of backward parameters

`m`
: Number of grid points

kernel	Type of kernel function
band_choice	Selection of optimal bandwidth
VAR_type	Type of vector autoregressive process
lag_maximum	A tuning parameter in the super_fun function
no_boot	A tuning parameter in the super_fun function
alpha_val	A tuning parameter in the super_fun function
ncomp_select	Logical. If TRUE, the number of components is selected by the bootstrap test of Bathia et al. (2010), and the associated p-values are returned. Default is TRUE.
D_val	A tuning parameter in the super_fun function

Details

1) Compute a kernel covariance function 2) Via eigen-decomposition, a density can be decomposed into a set of functional principal components and their associated scores 3) Fit a vector autoregressive model to the scores with the order selected by the Akaike information criterion 4) By multiplying the estimated functional principal components with the forecast scores, obtain forecast densities 5) Since forecast densities may fail to be non-negative or to integrate to one, normalize the forecast densities accordingly

Value

Yhat.fix_den	Forecast density
u	Grid points
du	Distance between two successive grid points
Ybar_est	Mean of density functions
psihat_est	Estimated functional principal components
etahat_est	Estimated principal component scores
etahat_pred_val	Forecast principal component scores
selected_d0	Selected number of components
selected_d0_pvalues	p-values associated with the selected functional principal components
thetahat_val	Estimated eigenvalues

Author(s)

Han Lin Shang

References

Horta, E. and Ziegelmann, F. (2018) 'Dynamics of financial returns densities: A functional approach applied to the Bovespa intraday index', *International Journal of Forecasting*, **34**, 75-88.

Bathia, N., Yao, Q. and Ziegelmann, F. (2010) 'Identifying the finite dimensionality of curve time series', *The Annals of Statistics*, **38**, 3353-3386.

See Also

CoDa_FPCA, LQDT_FPCA, skew_t_fun

Examples

```
## Not run:
Horta_Ziegelmann_FPCA(data = DJI_return, kernel = "epanechnikov",
    band_choice = "DPI", ncomp_select = FALSE)

## End(Not run)
```

H_estimate_LW_NFTS *Estimation of the long-memory parameter in nonstationary functional time series*

Description

Estimation of the long-memory parameter via local Whittle estimator in nonstationary functional time series

Usage

```
H_estimate_LW_NFTS(fun_dat, band_const, tuning_const, delta_val, m_const,
est_method = c("diff_add", "ELW", "ELW_2S"), choice_score = c("first", "average"),
taper_choice = c("none", "Velasco", "HC"),
ELW_mean_est = c("mean", "init", "weighted", "none"))
```

Arguments

fun_dat	A functional data set
band_const	Tuning parameter, customarily set to 1
tuning_const	Tuning parameter, customarily set to 0.4
delta_val	Tuning parameter, customarily set to 0.01
m_const	Tuning parameter, customarily set to 0.65
est_method	Estimation method from a set of principal component scores
choice_score	The leading set of principal component scores or a norm
taper_choice	Tapering parameter, customarily none
ELW_mean_est	When the estimation method is ELW, it is the mean term of the ELW

Value

C	Estimated long-run covariance
score	Estimated principal component scores
H_est	Hurst parameter estimate
d_est	Long-memory parameter estimate
alpha_est	Long-memory parameter estimate
ncomp	Number of retained components
ER	Eigenvalue ratio
diff_order	Order of difference
eigen_function	Principal components

Author(s)

Han Lin Shang

References

D. Li, P. M. Robinson, H. L. Shang (2020) Long-range dependent curve time series, *Journal of the American Statistical Association*, **115**(530), 957-971.

D. Li, P. M. Robinson, H. L. Shang (2021) Local Whittle estimation of long-range dependence for functional time series, *Journal of Time Series Analysis*, **42**(5-6), 685-695.

D. Li, P. M. Robinson, H. L. Shang (2023) Nonstationary fractionally integrated functional time series, *Bernoulli*, **29**(2), 1505-1526.

See Also

H_estimate_RS

Examples

```
sim_data_set = sim_FARMA(n = 50, seed_number = 123, process = "FAR")
LW_d_est = H_estimate_LW_NFTS(fun_dat=sim_data_set, band_const = 1, tuning_const = 0.4,
 delta_val = 0.01, m_const = 0.65, est_method = "diff_add", choice_score = "first",
 taper_choice = "none")
```

H_estimate_RS *Estimation of long-memory parameter*

Description

R/S estimate of the long-memory functional time series

Usage

```
H_estimate_RS(fun_dat, band_const = 1, choice_score = c("first", "average"),
    est_method = c("classical", "modified"))
```

Arguments

fun_dat	Functional data object on a set of discretized grid points
band_const	Constant multiplier
choice_score	First set of principal component scores or the norm of multiple principal component scores
est_method	Classical or modified R/S estimator

Value

C	Estimated covariance
score	Estimated principal component score
H_est	Estimated Hurst exponent
d_est	Estimated memory parameter
alpha_est	Estimated alpha parameter
ncomp	Number of retained components
prop_eigen_value	Proportions of eigenvalues
eigen_function	Retained principal components

Author(s)

Han Lin Shang

References

D. Li, P. M. Robinson, H. L. Shang (2020) Long-range dependent curve time series. Journal of the American Statistical Association, 115(530), 957-971.

See Also

sim_FARMA

Examples

```
sim_data_set = sim_FARMA(n = 50, seed_number = 123, process = "FAR")
RS_est = H_estimate_RS(sim_data_set, choice_score = "first", est_method = "classical")
```

is.fts *Test for functional time series*

Description

Tests whether an object is of class `fts`.

Usage

```
is.fts(x)
```

Arguments

`x`	Arbitrary R object.

Value

A logical value.

Author(s)

Rob J Hyndman

Examples

```
# check if ElNino is the class of the functional time series.
is.fts(x = ElNino_ERSST_region_1and2)
```

isfe.fts *Integrated Squared Forecast Error for models of various orders*

Description

Computes integrated squared forecast error (ISFE) values for functional time series models of various orders.

Usage

```
isfe.fts(data, max.order = N - 3, N = 10, h = 5:10, method =
 c("classical", "M", "rapca"), mean = TRUE, level = FALSE,
  fmethod = c("arima", "ar", "ets", "ets.na", "struct", "rwdrift",
   "rw", "arfima"), lambda = 3, ...)
```

Arguments

data	An object of class fts.
max.order	Maximum number of principal components to fit.
N	Minimum number of functional observations to be used in fitting a model.
h	Forecast horizons over which to average.
method	Method to use for principal components decomposition. Possibilities are "M", "rapca" and "classical".
mean	Indicates if mean term should be included.
level	Indicates if level term should be included.
fmethod	Method used for forecasting. Current possibilities are "ets", "arima", "ets.na", "ar", "arfima", "struct", "rwdrift" and "rw".
lambda	Tuning parameter for robustness when method = "M".
...	Additional arguments controlling the fitting procedure.

Value

Numeric matrix with (max.order+1) rows and length(h) columns containing ISFE values for models of orders 0:(max.order).

Note

This function can be very time consuming for data with large dimensionality or large sample size. By setting max.order small, computational speed can be dramatically increased.

Author(s)

Rob J Hyndman

References

R. J. Hyndman and M. S. Ullah (2007) "Robust forecasting of mortality and fertility rates: A functional data approach", *Computational Statistics and Data Analysis*, **51**(10), 4942-4956.

See Also

ftsm, forecast.ftsm, plot.fm, plot.fmres, summary.fm, residuals.fm

Examples

```
isfe.fts(data = ElNino_ERSST_region_1and2, max.order = 2, N = 10, h = 1:3)
```

`long_run_covariance_estimation` *Estimating long-run covariance function for a functional time series*

Description

Bandwidth estimation in the long-run covariance function for a functional time series, using different types of kernel function

Usage

```
long_run_covariance_estimation(dat, C0 = 3, H = 3)
```

Arguments

`dat`	A matrix of p by n, where p denotes the number of grid points and n denotes sample size
`C0`	A tuning parameter used in the adaptive bandwidth selection algorithm of Rice
`H`	A tuning parameter used in the adaptive bandwidth selection algorithm of Rice

Value

An estimated covariance function of size (p by p)

Author(s)

Han Lin Shang

References

L. Horvath, G. Rice and S. Whipple (2016) Adaptive bandwidth selection in the long run covariance estimator of functional time series, *Computational Statistics and Data Analysis*, **100**, 676-693.

G. Rice and H. L. Shang (2017) A plug-in bandwidth selection procedure for long run covariance estimation with stationary functional time series, *Journal of Time Series Analysis*, **38**(4), 591-609.

D. Li, P. M. Robinson and H. L. Shang (2020) Long-range dependent curve time series, *Journal of the American Statistical Association: Theory and Methods*, **115**(530), 957-971.

See Also

`fts`

Examples

```
dum = long_run_covariance_estimation(dat = ElNino_OISST_region_1and2$y[,1:5])
```

LQDT_FPCA *Log quantile density transform*

Description

Probability density function, cumulative distribution function, and quantile density function are three characterizations of a distribution. Of these three, the quantile density function is the least constrained. The only constraint is non-negativity. By taking a log transformation, there is no constraint.

Usage

```
LQDT_FPCA(data, gridpoints, h_scale = 1, M = 3001, m = 5001, lag_maximum = 4,
  no_boot = 1000, alpha_val = 0.1, p = 5,
  band_choice = c("Silverman", "DPI"),
  kernel = c("gaussian", "epanechnikov"),
  forecasting_method = c("uni", "multi"),
  varprop = 0.85, fmethod, VAR_type)
```

Arguments

data	Densities or raw data matrix of dimension N by p, where N denotes sample size and p denotes dimensionality
gridpoints	Grid points
h_scale	Scaling parameter in the kernel density estimator
M	Number of grid points between 0 and 1
m	Number of grid points within the data range
lag_maximum	A tuning parameter in the super_fun function
no_boot	A tuning parameter in the super_fun function
alpha_val	A tuning parameter in the super_fun function
p	Number of backward parameters
band_choice	Selection of optimal bandwidth
kernel	Type of kernel function
forecasting_method	Univariate or multivariate time series forecasting method
varprop	Proportion of variance explained
fmethod	If forecasting_method = "uni", specify a particular forecasting method
VAR_type	If forecasting_method = "multi", specify a particular type of vector autoregressive model

Details

1) Transform the densities f into log quantile densities Y and c, specifying the value of the cdf at 0 for the target density f. 2) Compute the predictions for future log quantile density and c value. 3) Transform the forecasts in Step 2) into the predicted density f.

Value

L2Diff	L2 norm difference between reconstructed and actual densities
unifDiff	Uniform Metric excluding missing boundary values (due to boundary cutoff)
density_reconstruct	Reconstructed densities
density_original	Actual densities
dens_fore	Forecast densities
totalMass	Assess loss of mass incurred by boundary cutoff
u	m number of grid points

Author(s)

Han Lin Shang

References

Petersen, A. and M\"uller, H.-G. (2016) 'Functional data analysis for density functions by transformation to a Hilbert space', *The Annals of Statistics*, **44**, 183-218.

Jones, M. C. (1992) 'Estimating densities, quantiles, quantile densities and density quantiles', *Annals of the Institute of Statistical Mathematics*, **44**, 721-727.

See Also

CoDa_FPCA, Horta_Ziegelmann_FPCA, skew_t_fun

Examples

```
## Not run:
LQDT_FPCA(data = DJI_return, band_choice = "DPI", kernel = "epanechnikov",
   forecasting_method = "uni", fmethod = "ets")

## End(Not run)
```

MAF_multivariate *Maximum autocorrelation factors*

Description

Dimension reduction via maximum autocorrelation factors

Usage

```
MAF_multivariate(data, threshold)
```

Arguments

data	A p by n data matrix, where p denotes the number of variables and n denotes the sample size
threshold	A threshold level for retaining the optimal number of factors

Value

MAF	Maximum autocorrelation factor scores
MAF_loading	Maximum autocorrelation factors
Z	Standardized original data
recon	Reconstruction via maximum autocorrelation factors
recon_err	Reconstruction errors between the standardized original data and reconstruction via maximum autocorrelation factors
ncomp_threshold	Number of maximum autocorrelation factors selected by explaining autocorrelation at and above a given level of threshold
ncomp_eigen_ratio	Number of maximum autocorrelation factors selected by eigenvalue ratio tests

Author(s)

Han Lin Shang

References

M. A. Haugen, B. Rajaratnam and P. Switzer (2015). Extracting common time trends from concurrent time series: Maximum autocorrelation factors with applications, arXiv paper https://arxiv.org/abs/1502.01073.

See Also

ftsm

Examples

```
MAF_multivariate(data = pm_10_GR_sqrt$y, threshold = 0.85)
```

---

mean.fts *Mean functions for functional time series*

Description

Computes mean of functional time series at each variable.

Usage

```
## S3 method for class 'fts'
mean(x, method = c("coordinate", "FM", "mode", "RP", "RPD", "radius"),
 na.rm = TRUE, alpha, beta, weight, ...)
```

Arguments

x	An object of class fts.
method	Method for computing the mean function.
na.rm	A logical value indicating whether NA values should be stripped before the computation proceeds.
alpha	Tuning parameter when method="radius".
beta	Trimming percentage, by default it is 0.25, when method="radius".
weight	Hard thresholding or soft thresholding.
...	Other arguments.

Details

If method = "coordinate", it computes the coordinate-wise functional mean.

If method = "FM", it computes the mean of trimmed functional data ordered by the functional depth of Fraiman and Muniz (2001).

If method = "mode", it computes the mean of trimmed functional data ordered by h-modal functional depth.

If method = "RP", it computes the mean of trimmed functional data ordered by random projection depth.

If method = "RPD", it computes the mean of trimmed functional data ordered by random projection derivative depth.

If method = "radius", it computes the mean of trimmed functional data ordered by the notion of alpha-radius.

Value

A list containing x = variables and y = mean rates.

Author(s)

Rob J Hyndman, Han Lin Shang

References

R. Fraiman and G. Muniz (2001) "Trimmed means for functional data", *Test*, **10**, 419-440.

O. H\"ossjer and C. Croux (1995) "Generalized univariate signed rank statistics for testing and estimating a multivariate location parameter", *Journal of Nonparametric Statistics*, **4**(3), 293-308.

A. Cuevas, M. Febrero and R. Fraiman (2006) "On the use of bootstrap for estimating functions with functional data", *Computational Statistics and Data Analysis*, **51**(2), 1063-1074.

A. Cuevas and M. Febrero and R. Fraiman (2007), "Robust estimation and classification for functional data via projection-based depth notions", *Computational Statistics*, **22**(3), 481-496.

M. Febrero and P. Galeano and W. Gonzalez-Manteiga (2007) "A functional analysis of NOx levels: location and scale estimation and outlier detection", *Computational Statistics*, **22**(3), 411-427.

M. Febrero and P. Galeano and W. Gonzalez-Manteiga (2008) "Outlier detection in functional data by depth measures, with application to identify abnormal NOx levels", *Environmetrics*, **19**(4), 331-345.

M. Febrero and P. Galeano and W. Gonzalez-Manteiga (2010) "Measures of influence for the functional linear model with scalar response", *Journal of Multivariate Analysis*, **101**(2), 327-339.

J. A. Cuesta-Albertos and A. Nieto-Reyes (2010) "Functional classification and the random Tukey depth. Practical issues", Combining Soft Computing and Statistical Methods in Data Analysis, *Advances in Intelligent and Soft Computing*, **77**, 123-130.

D. Gervini (2012) "Outlier detection and trimmed estimation in general functional spaces", *Statistica Sinica*, **22**(4), 1639-1660.

See Also

median.fts, var.fts, sd.fts, quantile.fts

Examples

```
# Calculate the mean function by the different depth measures.
mean(x = ElNino_ERSST_region_1and2, method = "coordinate")
mean(x = ElNino_ERSST_region_1and2, method = "FM")
mean(x = ElNino_ERSST_region_1and2, method = "mode")
mean(x = ElNino_ERSST_region_1and2, method = "RP")
mean(x = ElNino_ERSST_region_1and2, method = "RPD")
mean(x = ElNino_ERSST_region_1and2, method = "radius",
 alpha = 0.5, beta = 0.25, weight = "hard")
mean(x = ElNino_ERSST_region_1and2, method = "radius",
 alpha = 0.5, beta = 0.25, weight = "soft")
```

---

median.fts *Median functions for functional time series*

Description

Computes median of functional time series at each variable.

Usage

```
## S3 method for class 'fts'
median(x, na.rm, method = c("hossjercroux", "coordinate", "FM", "mode",
 "RP", "RPD", "radius"), alpha, beta, weight, ...)
```

Arguments

x	An object of class fts.
na.rm	Remove any missing value.
method	Method for computing median.
alpha	Tuning parameter when method="radius".
beta	Trimming percentage, by default it is 0.25, when method="radius".
weight	Hard thresholding or soft thresholding.
...	Other arguments.

Details

If method = "coordinate", it computes a coordinate-wise median.

If method = "hossjercroux", it computes the L1-median using the Hossjer-Croux algorithm.

If method = "FM", it computes the median of trimmed functional data ordered by the functional depth of Fraiman and Muniz (2001).

If method = "mode", it computes the median of trimmed functional data ordered by h-modal functional depth.

If method = "RP", it computes the median of trimmed functional data ordered by random projection depth.

If method = "RPD", it computes the median of trimmed functional data ordered by random projection derivative depth.

If method = "radius", it computes the median of trimmed functional data ordered by the notion of alpha-radius.

Value

A list containing x = variables and y = median rates.

Author(s)

Rob J Hyndman, Han Lin Shang

References

R. Fraiman and G. Muniz (2001) "Trimmed means for functional data", *Test*, **10**, 419-440.

O. H\"ossjer and C. Croux (1995) "Generalized univariate signed rank statistics for testing and estimating a multivariate location parameter", *Journal of Nonparametric Statistics*, **4**(3), 293-308.

A. Cuevas, M. Febrero and R. Fraiman (2006) "On the use of bootstrap for estimating functions with functional data", *Computational Statistics and Data Analysis*, **51**(2), 1063-1074.

A. Cuevas and M. Febrero and R. Fraiman (2007), "Robust estimation and classification for functional data via projection-based depth notions", *Computational Statistics*, **22**(3), 481-496.

M. Febrero and P. Galeano and W. Gonzalez-Manteiga (2007) "A functional analysis of NOx levels: location and scale estimation and outlier detection", *Computational Statistics*, **22**(3), 411-427.

M. Febrero and P. Galeano and W. Gonzalez-Manteiga (2008) "Outlier detection in functional data by depth measures, with application to identify abnormal NOx levels", *Environmetrics*, **19**(4), 331-345.

M. Febrero and P. Galeano and W. Gonzalez-Manteiga (2010) "Measures of influence for the functional linear model with scalar response", *Journal of Multivariate Analysis*, **101**(2), 327-339.

J. A. Cuesta-Albertos and A. Nieto-Reyes (2010) "Functional classification and the random Tukey depth. Practical issues", Combining Soft Computing and Statistical Methods in Data Analysis, *Advances in Intelligent and Soft Computing*, **77**, 123-130.

D. Gervini (2012) "Outlier detection and trimmed estimation in general functional spaces", *Statistica Sinica*, **22**(4), 1639-1660.

See Also

`mean.fts`, `var.fts`, `sd.fts`, `quantile.fts`

Examples

```
# Calculate the median function by the different depth measures.
median(x = ElNino_ERSST_region_1and2, method = "hossjercroux")
median(x = ElNino_ERSST_region_1and2, method = "coordinate")
median(x = ElNino_ERSST_region_1and2, method = "FM")
median(x = ElNino_ERSST_region_1and2, method = "mode")
median(x = ElNino_ERSST_region_1and2, method = "RP")
median(x = ElNino_ERSST_region_1and2, method = "RPD")
median(x = ElNino_ERSST_region_1and2, method = "radius",
 alpha = 0.5, beta = 0.25, weight = "hard")
median(x = ElNino_ERSST_region_1and2, method = "radius",
 alpha = 0.5, beta = 0.25, weight = "soft")
```

MFDM *Multilevel functional data method*

Description

Fit a multilevel functional principal component model. The function uses two-step functional principal component decompositions.

Usage

```
MFDM(mort_female, mort_male, mort_ave, percent_1 = 0.95, percent_2 = 0.95, fh,
      level = 80, alpha = 0.2, MCMCiter = 100, fmethod = c("auto_arima", "ets"),
     BC = FALSE, lambda)
```

Arguments

mort_female	Female mortality (p by n matrix), where p denotes the dimension and n denotes the sample size.
mort_male	Male mortality (p by n matrix).
mort_ave	Total mortality (p by n matrix).
percent_1	Cumulative percentage used for determining the number of common functional principal components.
percent_2	Cumulative percentage used for determining the number of sex-specific functional principal components.
fh	Forecast horizon.
level	Nominal coverage probability of a prediction interval.
alpha	1 - Nominal coverage probability.
MCMCiter	Number of MCMC iterations.
fmethod	Univariate time-series forecasting method.
BC	If Box-Cox transformation is performed.
lambda	If BC = TRUE, specify a Box-Cox transformation parameter.

Details

The basic idea of the multilevel functional data method is to decompose functions from different sub-populations into an aggregated average, a sex-specific deviation from the aggregated average, a common trend, a sex-specific trend, and measurement error. The common and sex-specific trends are modeled by projecting them onto the eigenvectors of covariance operators of the aggregated and sex-specific centered stochastic process, respectively.

Value

first_percent	Percentage of total variation explained by the first common functional principal component.
female_percent	Percentage of total variation explained by the first female functional principal component in the residual.
male_percent	Percentage of total variation explained by the first male functional principal component in the residual.
mort_female_fore	Forecast female mortality in the original scale.
mort_male_fore	Forecast male mortality in the original scale.

Note

It can be quite time-consuming, especially when MCMCiter is large.

Author(s)

Han Lin Shang

References

C. M. Crainiceanu and J. Goldsmith (2010) "Bayesian functional data analysis using WinBUGS", *Journal of Statistical Software*, **32**(11).

C-Z. Di and C. M. Crainiceanu and B. S. Caffo and N. M. Punjabi (2009) "Multilevel functional principal component analysis", *The Annals of Applied Statistics*, **3**(1), 458-488.

V. Zipunnikov and B. Caffo and D. M. Yousem and C. Davatzikos and B. S. Schwartz and C. Crainiceanu (2011) "Multilevel functional principal component analysis for high-dimensional data", *Journal of Computational and Graphical Statistics*, **20**(4), 852-873.

H. L. Shang, P. W. F. Smith, J. Bijak, A. Wisniowski (2016) "A multilevel functional data method for forecasting population, with an application to the United Kingdom", *International Journal of Forecasting*, **32**(3), 629-649.

See Also

`ftsm`, `forecast.ftsm`, `fplsr`, `forecastfplsr`

Examples

```
## Not run:
# MFDM requires JAGS (external software) and the R2jags package.
# mort_female, mort_male and mort_ave are (p by n) log mortality matrices.
MFDM(mort_female = female_mort, mort_male = male_mort, mort_ave = total_mort,
     fh = 10, MCMCiter = 100, fmethod = "ets", BC = FALSE)

## End(Not run)
```

---

pcscorebootstrapdata *Bootstrap independent and identically distributed functional data or functional time series*

Description

Computes bootstrap or smoothed bootstrap samples based on either independent and identically distributed functional data or functional time series.

Usage

```
pcscorebootstrapdata(dat, bootrep, statistic, bootmethod = c("st", "sm",
 "mvn", "stiefel", "meboot"), smo)
```

Arguments

`dat`	An object of class `matrix`.
`bootrep`	Number of bootstrap samples.
`statistic`	Summary statistics.

`bootmethod` Bootstrap method. When `bootmethod = "st"`, sampling with replacement is implemented. To avoid the repeated bootstrap samples, the smoothed bootstrap method can be implemented by adding multivariate Gaussian random noise. When `bootmethod = "mvn"`, the bootstrapped principal component scores are drawn from a multivariate Gaussian distribution with the mean and covariance matrices of the original principal component scores. When `bootmethod = "stiefel"`, the bootstrapped principal component scores are drawn from a Stiefel manifold with the mean and covariance matrices of the original principal component scores. When `bootmethod = "meboot"`, the bootstrapped principal component scores are drawn from a maximum entropy algorithm of Vinod (2004).

`smo` Smoothing parameter.

Details

We will presume that each curve is observed on a grid of T points with $0 \leq t_1 < t_2 \cdots < t_T \leq \tau$. Thus, the raw data set $(X_1, X_2, \ldots, X_n)$ of n observations will consist of an n by T data matrix. By applying the singular value decomposition, $X_1, X_2, \ldots, X_n$ can be decomposed into $X = ULR^\top$, where the crossproduct of U and R is the identity matrix.

Holding the mean and L and R fixed at their realized values, there are four re-sampling methods that differ mainly by the ways of re-sampling U.

(a) Obtain the re-sampled singular column matrix by randomly sampling with replacement from the original principal component scores.

(b) To avoid the appearance of repeated values in bootstrapped principal component scores, we adapt a smooth bootstrap procedure by adding a white noise component to the bootstrap.

(c) Because principal component scores follow a standard multivariate normal distribution asymptotically, we can randomly draw principal component scores from a multivariate normal distribution with the mean vector and covariance matrix of the original principal component scores.

(d) Because the cross-product of U is the identity matrix, U is considered as a point on the Stiefel manifold, which is the space of n orthogonal vectors; thus, we can randomly draw principal component scores from the Stiefel manifold.

Value

`bootdata` Bootstrap samples. If the original data matrix is p by n, then the bootstrapped data are p by n by $bootrep$.

`meanfunction` Bootstrap summary statistics. If the original data matrix is p by n, then the bootstrapped summary statistics are p by $bootrep$.

Author(s)

Han Lin Shang

References

H. D. Vinod (2004), "Ranking mutual funds using unconventional utility theory and stochastic dominance", *Journal of Empirical Finance*, **11**(3), 353-377.

A. Cuevas, M. Febrero, R. Fraiman (2006), "On the use of the bootstrap for estimating functions with functional data", *Computational Statistics and Data Analysis*, **51**(2), 1063-1074.

D. S. Poskitt and A. Sengarapillai (2013), "Description length and dimensionality reduction in functional data analysis", *Computational Statistics and Data Analysis*, **58**, 98-113.

H. L. Shang (2015), "Re-sampling techniques for estimating the distribution of descriptive statistics of functional data", *Communications in Statistics–Simulation and Computation*, **44**(3), 614-635.

H. L. Shang (2018), "Bootstrap methods for stationary functional time series", *Statistics and Computing*, **28**(1), 1-10.

See Also

fbootstrap

Examples

```
# Bootstrapping the distribution of a summary statistics of functional data.
boot1 = pcscorebootstrapdata(dat = ElNino_ERSST_region_1and2$y, bootrep = 200,
 statistic = "mean", bootmethod = "st")
boot2 = pcscorebootstrapdata(dat = ElNino_ERSST_region_1and2$y, bootrep = 200,
 statistic = "mean", bootmethod = "sm", smo = 0.05)
boot3 = pcscorebootstrapdata(dat = ElNino_ERSST_region_1and2$y, bootrep = 200,
 statistic = "mean", bootmethod = "mvn")
boot4 = pcscorebootstrapdata(dat = ElNino_ERSST_region_1and2$y, bootrep = 200,
 statistic = "mean", bootmethod = "stiefel")
boot5 = pcscorebootstrapdata(dat = ElNino_ERSST_region_1and2$y, bootrep = 200,
 statistic = "mean", bootmethod = "meboot")
```

---

plot.fm *Plot fitted model components for a functional model*

Description

When `class(x)[1]` = `ftsm`, plot showing the principal components in the top row of plots and the coefficients in the bottom row of plots.

When `class(x)[1]` = `fm`, plot showing the predictor scores in the top row of plots and the response loadings in the bottom row of plots.

Usage

```
## S3 method for class 'fm'
plot(x, order, xlab1 = x$y$xname, ylab1 = "Principal component",
 xlab2 = "Time", ylab2 = "Coefficient", mean.lab = "Mean",
  level.lab = "Level", main.title = "Main effects", interaction.title
   = "Interaction", basiscol = 1, coeffcol = 1, outlier.col = 2,
    outlier.pch = 19, outlier.cex = 0.5, ...)
```

Arguments

x	Output from ftsm or fplsr.
order	Number of principal components to plot. Default is all principal components in a model.
xlab1	x-axis label for principal components.
xlab2	x-axis label for coefficient time series.
ylab1	y-axis label for principal components.
ylab2	y-axis label for coefficient time series.
mean.lab	Label for mean component.
level.lab	Label for level component.
main.title	Title for main effects.
interaction.title	Title for interaction terms.
basiscol	Color for the principal components.
coeffcol	Color for the coefficient time series.
outlier.col	Colors for outlying years.
outlier.pch	Plotting character for outlying years.
outlier.cex	Size of plotting character for outlying years.
...	Plotting parameters.

Value

Function produces a plot.

Author(s)

Rob J Hyndman

References

R. J. Hyndman and M. S. Ullah (2007) "Robust forecasting of mortality and fertility rates: A functional data approach", *Computational Statistics and Data Analysis*, **51**(10), 4942-4956.

R. J. Hyndman and H. Booth (2008) "Stochastic population forecasts using functional data models for mortality, fertility and migration", *International Journal of Forecasting*, **24**(3), 323-342.

R. J. Hyndman and H. L. Shang (2009) "Forecasting functional time series (with discussion)", *Journal of the Korean Statistical Society*, **38**(3), 199-221.

See Also

ftsm, forecast.ftsm, residuals.fm, summary.fm, plot.fmres, plot.ftsf

Examples

```
plot(x = ftsm(y = ElNino_ERSST_region_1and2))
```

`plot.fmres` *Plot residuals from a fitted functional model*

Description

Functions to produce a plot of residuals from a fitted functional model.

Usage

```
## S3 method for class 'fmres'
plot(x, type = c("image", "fts", "contour", "filled.contour",
 "persp"), xlab = "Year", ylab = "Age", zlab = "Residual", ...)
```

Arguments

`x`	Generated by `residuals(fit)`, where fit is the output from `ftsm` or `fplsr`.
`type`	Type of plot to use. Possibilities are `image`, `fts`, `contour`, `filled.contour` and `persp`.
`xlab`	Label for x-axis.
`ylab`	Label for y-axis.
`zlab`	Label for z-axis.
`...`	Plotting parameters.

Value

Produces a plot.

Author(s)

Rob J Hyndman

See Also

`ftsm`, `forecast.ftsm`, `plot.fm`, `residuals.fm`, `summary.fm`

Examples

```
# colorspace package was used to provide a more coherent color option.
plot(residuals(ftsm(y = ElNino_ERSST_region_1and2)), type = "filled.contour", xlab = "Month",
     ylab = "Residual sea surface temperature")
```

plot.ftsf *Plot fitted model components for a functional time series model*

Description

Plot fitted model components for an fts object.

Usage

```
## S3 method for class 'ftsf'
plot(x, plot.type = c("function", "components", "variance"),
 components, xlab1 = fit$y$xname, ylab1 = "Basis function",
  xlab2 = "Time", ylab2 = "Coefficient", mean.lab = "Mean",
   level.lab = "Level", main.title = "Main effects",
    interaction.title = "Interaction", vcol = 1:3, shadecols = 7,
     fcol = 4, basiscol = 1, coeffcol = 1, outlier.col = 2,
      outlier.pch = 19, outlier.cex = 0.5,...)
```

Arguments

x	Output from forecast.ftsm.
plot.type	Type of plot.
components	Number of principal components.
xlab1	x-axis label for principal components.
xlab2	x-axis label for coefficient time series.
ylab1	y-axis label for principal components.
ylab2	y-axis label for coefficient time series.
mean.lab	Label for mean component.
level.lab	Label for level component.
main.title	Title for main effects.
interaction.title	Title for interaction terms.
vcol	Colors to use if plot.type = "variance".
shadecols	Color for shading of prediction intervals when plot.type = "components".
fcol	Color of point forecasts when plot.type = "components".
basiscol	Colors for principal components if plot.type = "components".
coeffcol	Colors for time series coefficients if plot.type = "components".
outlier.col	Colors for outlying years.
outlier.pch	Plotting character for outlying years.
outlier.cex	Size of plotting character for outlying years.
...	Plotting parameters.

Details

When `plot.type = "function"`, it produces a plot of the forecast functions;

When `plot.type = "components"`, it produces a plot of the principal components and coefficients with forecasts and prediction intervals for each coefficient;

When `plot.type = "variance"`, it produces a plot of the variance components.

Value

Function produces a plot.

Author(s)

Rob J Hyndman

References

R. J. Hyndman and M. S. Ullah (2007) "Robust forecasting of mortality and fertility rates: A functional data approach", *Computational Statistics and Data Analysis*, **51**(10), 4942-4956.

R. J. Hyndman and H. Booth (2008) "Stochastic population forecasts using functional data models for mortality, fertility and migration", *International Journal of Forecasting*, **24**(3), 323-342.

R. J. Hyndman and H. L. Shang (2009) "Forecasting functional time series (with discussion)", *Journal of the Korean Statistical Society*, **38**(3), 199-221.

H. L. Shang, H. Booth and R. J. Hyndman (2011) "Point and interval forecasts of mortality rates and life expectancy: A comparison of ten principal component methods", *Demographic Research*, **25**(5), 173-214.

See Also

`ftsm`, `plot.fm`, `plot.fmres`, `residuals.fm`, `summary.fm`

Examples

```
plot(x = forecast(object = ftsm(y = ElNino_ERSST_region_1and2)))
```

`plot.ftsm` *Plot fitted model components for a functional time series model*

Description

Plot showing the basis functions in the top row of plots and the coefficients in the bottom row of plots.

Usage

```
## S3 method for class 'ftsm'
plot(x, components, components.start = 0, xlab1 = x$y$xname, ylab1 = "Basis function",
 xlab2 = "Time", ylab2 = "Coefficient", mean.lab = "Mean",
  level.lab = "Level", main.title = "Main effects",
   interaction.title = "Interaction", basiscol = 1, coeffcol = 1,
    outlier.col = 2, outlier.pch = 19, outlier.cex = 0.5, ...)
```

Arguments

x	Output from ftsm.
components	Index of the last principal component to plot.
components.start	Index of the first component to plot. The default 0 includes the mean and level panels before the first component.
xlab1	x-axis label for basis functions.
xlab2	x-axis label for coefficient time series.
ylab1	y-axis label for basis functions.
ylab2	y-axis label for coefficient time series.
mean.lab	Label for mean component.
level.lab	Label for level component.
main.title	Title for main effects.
interaction.title	Title for interaction terms.
basiscol	Color for the basis functions.
coeffcol	Color for the coefficient time series.
outlier.col	Color for outlying years, used only when x was fitted with ftsm(..., weight = TRUE).
outlier.pch	Plotting character for outlying years.
outlier.cex	Size of plotting character for outlying years.
...	Plotting parameters.

Value

None. Function produces a plot.

Author(s)

Rob J Hyndman

References

R. J. Hyndman and M. S. Ullah (2007) "Robust forecasting of mortality and fertility rates: A functional data approach", *Computational Statistics and Data Analysis*, **51**(10), 4942-4956.

R. J. Hyndman and H. L. Shang (2009) "Forecasting functional time series" (with discussion), *Journal of the Korean Statistical Society*, **38**(3), 199-221.

See Also

forecast.ftsm, ftsm, plot.fm, plot.ftsf, residuals.fm, summary.fm

Examples

```
# plot different principal components.
plot.ftsm(ftsm(y = ElNino_ERSST_region_1and2, order = 2), components = 2)
```

---

plotfplsr *Plot fitted model components for a functional time series model*

Description

Plot showing the basis functions of the predictors in the top row, followed by the basis functions of the responses in the second row, then the coefficients in the bottom row of plots.

Usage

```
plotfplsr(x, xlab1 = x$ypred$xname, ylab1 = "Basis function", xlab2 = "Time",
 ylab2 = "Coefficient", mean.lab = "Mean", interaction.title = "Interaction")
```

Arguments

x	Output from fplsr.
xlab1	x-axis label for basis functions.
ylab1	y-axis label for basis functions.
xlab2	x-axis label for coefficient time series.
ylab2	y-axis label for coefficient time series.
mean.lab	Label for mean component.
interaction.title	Title for interaction terms.

Value

None. Function produces a plot.

Author(s)

Han Lin Shang

References

R. J. Hyndman and M. S. Ullah (2007) "Robust forecasting of mortality and fertility rates: A functional data approach", *Computational Statistics and Data Analysis*, **51**(10), 4942-4956.

R. J. Hyndman and H. L. Shang (2009) "Forecasting functional time series" (with discussion), *Journal of the Korean Statistical Society*, **38**(3), 199-221.

See Also

forecast.ftsm, ftsm, plot.fm, plot.ftsf, residuals.fm, summary.fm

Examples

```
# Fit the data by the functional partial least squares.
ausfplsr = fplsr(data = ElNino_ERSST_region_1and2, order = 2)
plotfplsr(x = ausfplsr)
```

---

pm_10_GR *Particulate Matter Concentrations (pm10)*

Description

This data set consists of half-hourly measurements of the concentrations (measured in ug/m3) of particulate matter with an aerodynamic diameter of less than 10 μm, abbreviated PM10, in ambient air taken in Graz-Mitte, Austria from October 1, 2010 until March 31, 2011. To stabilize the variance, a square-root transformation can be applied to the data.

Usage

```
data(pm_10_GR)
```

Format

A data set.

Details

As epidemiological and toxicological studies have pointed to negative health effects, European Union (EU) regulation sets pollution standards for the level of the concentration. Policy makers have to ensure compliance with these EU rules and need reliable statistical tools to determine appropriate measures, such as partial traffic regulation, and to justify them to the public (see Stadlober, H\"ormann and Pfeiler, 2008).

Source

Thanks to Professor Siegfried H\"ormann for providing this data set. The original data source is https://zenodo.org/records/7959116

References

A. Aue, D. D. Norinho, S. H\"ormann (2015) "On the prediction of stationary functional time series", *Journal of the American Statistical Association*, **110**(509), 378-392.

E. Stadlober, S. H\"ormann, B. Pfeiler (2008) "Quality and performance of a PM10 daily forecasting model", *Atmospheric Environment*, **42**, 1098-1109.

S. H\"ormann, B. Pfeiler, E. Stadlober (2005) "Analysis and prediction of particulate matter PM10 for the winter season in Graz", *Austrian Journal of Statistics*, **34**(4), 307-326.

H. L. Shang (2017) "Functional time series forecasting with dynamic updating: An application to intraday particulate matter concentration", *Econometrics and Statistics*, **1**, 184-200.

Examples

```
plot(pm_10_GR)
```

`quantile` *Quantile*

Description

Generic functions for quantile.

Usage

```
quantile(x, ...)
```

Arguments

x	Numeric vector whose sample quantiles are wanted, or an object of a class for which a method has been defined.
...	Arguments passed to specific methods.

Value

Refer to specific methods. For numeric vectors, see the `quantile` functions in the `stats` package.

Author(s)

Han Lin Shang

See Also

`quantile.fts`

Examples

```
quantile(1:10)
quantile(x = ElNino_ERSST_region_1and2, probs = c(0.25, 0.5, 0.75))
```

quantile.fts *Quantile functions for functional time series*

Description

Computes quantiles of functional time series at each variable.

Usage

```
## S3 method for class 'fts'
quantile(x, probs, ...)
```

Arguments

x	An object of class fts.
probs	Quantile percentages.
...	Other arguments.

Value

Return quantiles for each variable.

Author(s)

Han Lin Shang

See Also

mean.fts, median.fts, var.fts, sd.fts

Examples

```
quantile(x = ElNino_ERSST_region_1and2)
```

residuals.fm *Compute residuals from a functional model*

Description

After fitting a functional model, it is useful to inspect the residuals. This function extracts the relevant information from the fit object and puts it in a form suitable for plotting with image, persp, contour, filled.contour, etc.

Usage

```
## S3 method for class 'fm'
residuals(object, ...)
```

Arguments

object	Output from ftsm or fplsr.
...	Other arguments.

Value

Produces an object of class "fmres" containing the residuals from the model.

Author(s)

Rob J Hyndman

References

B. Erbas and R. J. Hyndman and D. M. Gertig (2007) "Forecasting age-specific breast cancer mortality using functional data model", *Statistics in Medicine*, **26**(2), 458-470.

R. J. Hyndman and M. S. Ullah (2007) "Robust forecasting of mortality and fertility rates: A functional data approach", *Computational Statistics and Data Analysis*, **51**(10), 4942-4956.

R. J. Hyndman and H. Booth (2008) "Stochastic population forecasts using functional data models for mortality, fertility and migration", *International Journal of Forecasting*, **24**(3), 323-342.

H. L. Shang (2012) "Point and interval forecasts of age-specific fertility rates: a comparison of functional principal component methods", *Journal of Population Research*, **29**(3), 249-267.

H. L. Shang (2012) "Point and interval forecasts of age-specific life expectancies: a model averaging", *Demographic Research*, **27**, 593-644.

See Also

ftsm, forecast.ftsm, summary.fm, plot.fm, plot.fmres

Examples

```
plot(residuals(object = ftsm(y = ElNino_ERSST_region_1and2)),
 xlab = "Year", ylab = "Month")
```

---

sd *Standard deviation*

Description

Generic functions for standard deviation.

Usage

```
sd(...)
```

Arguments

...	Arguments passed to specific methods.

Details

The sd functions in the stats package are replaced by sd.default.

Value

Refer to specific methods. For numeric vectors, see the sd functions in the stats package.

Author(s)

Han Lin Shang

See Also

sd.fts

Examples

```
# the generic dispatches to sd.default for a numeric vector
sd(1:10)
# and to sd.fts for a functional time series
sd(x = ElNino_ERSST_region_1and2, method = "coordinate")
```

---

sd.fts *Standard deviation functions for functional time series*

Description

Computes standard deviation of functional time series at each variable.

Usage

```
## S3 method for class 'fts'
sd(x, method = c("coordinate", "FM", "mode", "RP", "RPD", "radius"),
 trim = 0.25, alpha, weight,...)
```

Arguments

x	An object of class fts.
method	Method for computing standard deviation function.
trim	Percentage of trimming.
alpha	Tuning parameter when method="radius".
weight	Hard thresholding or soft thresholding.
...	Other arguments.

Details

If `method = "coordinate"`, it computes coordinate-wise standard deviation functions.

If `method = "FM"`, it computes the standard deviation functions of trimmed functional data ordered by the functional depth of Fraiman and Muniz (2001).

If `method = "mode"`, it computes the standard deviation functions of trimmed functional data ordered by h-modal functional depth.

If `method = "RP"`, it computes the standard deviation functions of trimmed functional data ordered by random projection depth.

If `method = "RPD"`, it computes the standard deviation functions of trimmed functional data ordered by random projection with derivative depth.

If `method = "radius"`, it computes the standard deviation function of trimmed functional data ordered by the notion of alpha-radius.

Value

A list containing `x` = variables and `y` = standard deviation rates.

Author(s)

Han Lin Shang

References

R. Fraiman and G. Muniz (2001) "Trimmed means for functional data", *Test*, **10**, 419-440.

O. H\"ossjer and C. Croux (1995) "Generalized univariate signed rank statistics for testing and estimating a multivariate location parameter", *Journal of Nonparametric Statistics*, **4**(3), 293-308.

A. Cuevas, M. Febrero and R. Fraiman (2006) "On the use of bootstrap for estimating functions with functional data", *Computational Statistics and Data Analysis*, **51**(2), 1063-1074.

A. Cuevas and M. Febrero and R. Fraiman (2007), "Robust estimation and classification for functional data via projection-based depth notions", *Computational Statistics*, **22**(3), 481-496.

M. Febrero and P. Galeano and W. Gonzalez-Manteiga (2007) "A functional analysis of NOx levels: location and scale estimation and outlier detection", *Computational Statistics*, **22**(3), 411-427.

M. Febrero and P. Galeano and W. Gonzalez-Manteiga (2008) "Outlier detection in functional data by depth measures, with application to identify abnormal NOx levels", *Environmetrics*, **19**(4), 331-345.

M. Febrero and P. Galeano and W. Gonzalez-Manteiga (2010) "Measures of influence for the functional linear model with scalar response", *Journal of Multivariate Analysis*, **101**(2), 327-339.

J. A. Cuesta-Albertos and A. Nieto-Reyes (2010) "Functional classification and the random Tukey depth. Practical issues", Combining Soft Computing and Statistical Methods in Data Analysis, *Advances in Intelligent and Soft Computing*, **77**, 123-130.

D. Gervini (2012) "Outlier detection and trimmed estimation in general functional spaces", *Statistica Sinica*, **22**(4), 1639-1660.

See Also

mean.fts, median.fts, var.fts, quantile.fts

Examples

```
# Fraiman-Muniz depth was arguably the oldest functional depth.
sd(x = ElNino_ERSST_region_1and2, method = "FM")
sd(x = ElNino_ERSST_region_1and2, method = "coordinate")
sd(x = ElNino_ERSST_region_1and2, method = "mode")
sd(x = ElNino_ERSST_region_1and2, method = "RP")
sd(x = ElNino_ERSST_region_1and2, method = "RPD")
sd(x = ElNino_ERSST_region_1and2, method = "radius",
 alpha = 0.5, weight = "hard")
sd(x = ElNino_ERSST_region_1and2, method = "radius",
 alpha = 0.5, weight = "soft")
```

---

sim_FARMA *Simulating data from a functional autoregressive moving average process*

Description

Simulating data from a stationary functional autoregressive moving average process

Usage

```
sim_FARMA(n = 500, d = 0.1, no_grid = 101, seed_number, process = c("FAR", "FARMA"))
```

Arguments

n	Number of curves
d	Memory parameter
no_grid	Number of grid points within a unit interval
seed_number	Pseudo-random seed number
process	FAR or FARMA process

Value

Simulated curves

Author(s)

Han Lin Shang

References

D. Li, P. M. Robinson, H. L. Shang (2020) Long-range dependent curve time series. Journal of the American Statistical Association, 115(530), 957-971.

See Also

H_estimate_RS

Examples

```
sim_data_set = sim_FARMA(seed_number = 123, process = "FAR")
```

---

skew_t_fun *Skewed t distribution*

Description

Fitting a parametric skewed t distribution using Fernandez and Steel's (1998) method

Usage

```
skew_t_fun(data, gridpoints, M = 5001)
```

Arguments

data	a data matrix of dimension n by p
gridpoints	Grid points
M	number of grid points

Details

1) Fit a skewed t distribution to data, and obtain four latent parameters; 2) Transform the four latent parameters so that they are un-constrained; 3) Fit a vector autoregressive model to these transformed latent parameters; 4) Obtain their forecasts, and then back-transform them to the original scales; 5) Via the skewed t distribution in Step 1), we obtain forecast density using the forecast latent parameters.

Value

m	Grid points within data range
skewed_t_den_fore	Density forecasts via a skewed t distribution

Note

This is a parametric approach for fitting and forecasting density.

Author(s)

Han Lin Shang

References

Fernandez, C. and Steel, M. F. J. (1998), 'On Bayesian modeling of fat tails and skewness', *Journal of the American Statistical Association: Theory and Methods*, **93**(441), 359-371.

See Also

CoDa_FPCA, Horta_Ziegelmann_FPCA, LQDT_FPCA

Examples

```
skew_t_fun(DJI_return)
```

---

stop_time_detect *Detection of the optimal stopping time in a curve time series*

Description

Detecting the optimal stopping time for the glue curing of wood panels in an automatic process environment.

Usage

```
stop_time_detect(data, forecasting_method = c("ets", "arima", "rw"))
```

Arguments

data	An object of class fts
forecasting_method	A univariate time series forecasting method

Value

break_points_strucchange	Breakpoints detected by the regression approach
break_points_ecp	Breakpoints detected by the distance-based approach
err_forward	Forward integrated squared forecast errors
err_backward	Backward integrated squared forecast errors (ISFEs)
ncomp_select_forward	Number of components selected by the eigenvalue ratio tests based on the forward ISFEs
ncomp_select_backward	Number of components selected by the eigenvalue ratio tests based on the backward ISFEs

Author(s)

Han Lin Shang

References

Bekhta, P., Ortynska, G. and Sedliacik, J. (2014). Properties of modified phenol-formaldehyde adhesive for plywood panels manufactured from high moisture content veneer. *Drvna Industrija*, **65**(4), 293-301.

See Also

stop_time_sim_data

Examples

```
sim_dat = stop_time_sim_data(sample_size = 50, omega = 0.1, seed_number = 123)
stop_time_detect(data = sim_dat, forecasting_method = "rw")
```

---

stop_time_sim_data *Simulated functional time series from a functional autoregression of order one*

Description

For detecting the optimal stopping time, we simulate a curve time series that follows a functional autoregression of order 1, with a breakpoint in the middle point of the entire sample.

Usage

```
stop_time_sim_data(sample_size, omega, seed_number)
```

Arguments

sample_size	Number of curves
omega	Noise level
seed_number	Random seed number

Value

An object of class `fts`

Author(s)

Han Lin Shang

See Also

stop_time_detect

Examples

```
stop_time_sim_data(sample_size = 401, omega = 0.1, seed_number = 123)
```

---

summary.fm *Summary for functional time series model*

Description

Summarizes a basis function model fitted to a functional time series. It returns various measures of goodness-of-fit.

Usage

```
## S3 method for class 'fm'
summary(object, ...)
```

Arguments

object	Output from ftsm or fplsr.
...	Other arguments.

Value

None.

Author(s)

Rob J Hyndman

See Also

ftsm, forecast.ftsm, residuals.fm, plot.fm, plot.fmres

Examples

```
summary(object = ftsm(y = ElNino_ERSST_region_1and2))
```

T_stationary *Testing stationarity of functional time series*

Description

A hypothesis test for stationarity of functional time series.

Usage

```
T_stationary(sample, L = 49, J = 500, MC_rep = 1000, cumulative_var = .90,
      Ker1 = FALSE, Ker2 = TRUE, h = ncol(sample)^.5, pivotal = FALSE,
      use_table = FALSE, significance)
```

Arguments

sample	A matrix of discretised curves of dimension (p by n), where p represents the dimensionality and n represents sample size.
L	Number of Fourier basis functions.
J	Truncation level used to approximate the distribution of the squared integrals of Brownian bridges that appear in the limit distribution.
MC_rep	Number of replications.
cumulative_var	Amount of variance explained.
Ker1	Flat top kernel in (4.1) of Horvath et al. (2014).
Ker2	Flat top kernel in (7) of Politis (2003).
h	Kernel bandwidth.
pivotal	If pivotal = TRUE, a pivotal statistic is used.
use_table	If use_table = TRUE, use the critical values that are available in the book titled Inference for Functional Data (Table 6.1, page 88).
significance	Level of significance. Possibilities are "10%", "5%", "1%".

Details

As in traditional (scalar and vector) time series analysis, many inferential procedures for functional time series assume stationarity. Stationarity is required for functional dynamic regression models, for bootstrap and resampling methods for functional time series and for the functional analysis of volatility.

Value

p-value	When p-value is less than any level of significance, we reject the null hypothesis and conclude that the tested functional time series is not stationary.

Author(s)

Gregory Rice and Han Lin Shang

References

L. Horvath and Kokoszka, P. (2012) Inference for Functional Data with Applications, Springer, New York.

L. Horvath, P. Kokoszka, G. Rice (2014) "Testing stationarity of functional time series", *Journal of Econometrics*, **179**(1), 66-82.

D. N. Politis (2003) "Adaptive bandwidth choice", *Journal of Nonparametric Statistics*, **15**(4-5), 517-533.

A. Aue, G. Rice, O. S\"onmez (2018) "Detecting and dating structural breaks in functional data without dimension reduction", *Journal of the Royal Statistical Society: Series B*, **80**(3), 509-529.

See Also

`farforecast`

Examples

```
result = T_stationary(sample = pm_10_GR_sqrt$y)
result_pivotal = T_stationary(sample = pm_10_GR_sqrt$y, J = 100, MC_rep = 5000,
    h = 20, pivotal = TRUE)
```

`var` *Variance*

Description

Generic functions for variance.

Usage

```
var(...)
```

Arguments

`...`	Arguments passed to specific methods.

Details

The `var` functions in the `stats` package are replaced by `var.default`.

Value

Refer to specific methods. For numeric vectors, see the `var` functions in the `stats` package.

Author(s)

Rob J Hyndman and Han Lin Shang

See Also

var.fts

Examples

```
var(1:10)
var(x = ElNino_ERSST_region_1and2, method = "coordinate")
```

---

var.fts *Variance functions for functional time series*

Description

Computes variance functions of functional time series at each variable.

Usage

```
## S3 method for class 'fts'
var(x, method = c("coordinate", "FM", "mode", "RP", "RPD", "radius"),
 trim = 0.25, alpha, weight, ...)
```

Arguments

x	An object of class fts.
method	Method for computing variance function.
trim	Percentage of trimming.
alpha	Tuning parameter when method="radius".
weight	Hard thresholding or soft thresholding.
...	Other arguments.

Details

If method = "coordinate", it computes coordinate-wise variance.

If method = "FM", it computes the variance of trimmed functional data ordered by the functional depth of Fraiman and Muniz (2001).

If method = "mode", it computes the variance of trimmed functional data ordered by h-modal functional depth.

If method = "RP", it computes the variance of trimmed functional data ordered by random projection depth.

If method = "RPD", it computes the variance of trimmed functional data ordered by random projection derivative depth.

If method = "radius", it computes the standard deviation function of trimmed functional data ordered by the notion of alpha-radius.

Value

A list containing x = variables and y = variance rates.

Author(s)

Han Lin Shang

References

R. Fraiman and G. Muniz (2001) "Trimmed means for functional data", *Test*, **10**, 419-440.

O. H\"ossjer and C. Croux (1995) "Generalized univariate signed rank statistics for testing and estimating a multivariate location parameter", *Journal of Nonparametric Statistics*, **4**(3), 293-308.

A. Cuevas, M. Febrero and R. Fraiman (2006) "On the use of bootstrap for estimating functions with functional data", *Computational Statistics and Data Analysis*, **51**(2), 1063-1074.

A. Cuevas and M. Febrero and R. Fraiman (2007), "Robust estimation and classification for functional data via projection-based depth notions", *Computational Statistics*, **22**(3), 481-496.

M. Febrero and P. Galeano and W. Gonzalez-Manteiga (2007) "A functional analysis of NOx levels: location and scale estimation and outlier detection", *Computational Statistics*, **22**(3), 411-427.

M. Febrero and P. Galeano and W. Gonzalez-Manteiga (2008) "Outlier detection in functional data by depth measures, with application to identify abnormal NOx levels", *Environmetrics*, **19**(4), 331-345.

M. Febrero and P. Galeano and W. Gonzalez-Manteiga (2010) "Measures of influence for the functional linear model with scalar response", *Journal of Multivariate Analysis*, **101**(2), 327-339.

J. A. Cuesta-Albertos and A. Nieto-Reyes (2010) "Functional classification and the random Tukey depth. Practical issues", Combining Soft Computing and Statistical Methods in Data Analysis, *Advances in Intelligent and Soft Computing*, **77**, 123-130.

D. Gervini (2012) "Outlier detection and trimmed estimation in general functional spaces", *Statistica Sinica*, **22**(4), 1639-1660.

See Also

mean.fts, median.fts, sd.fts, quantile.fts

Examples

```
# Fraiman-Muniz depth was arguably the oldest functional depth.
var(x = ElNino_ERSST_region_1and2, method = "FM")
var(x = ElNino_ERSST_region_1and2, method = "coordinate")
var(x = ElNino_ERSST_region_1and2, method = "mode")
var(x = ElNino_ERSST_region_1and2, method = "RP")
var(x = ElNino_ERSST_region_1and2, method = "RPD")
var(x = ElNino_ERSST_region_1and2, method = "radius",
alpha = 0.5, weight = "hard")
var(x = ElNino_ERSST_region_1and2, method = "radius",
alpha = 0.5, weight = "soft")
```

Index

∗ **datagen**
stop_time_sim_data, 72
∗ **datasets**
DJI_return, 12
pm_10_GR, 63
∗ **distribution**
skew_t_fun, 70
∗ **hplot**
plot.fm, 56
plot.fmres, 58
plot.ftsf, 59
∗ **methods**
CoDa_BayesNW, 8
CoDa_FPCA, 10
ER_GR, 18
error, 16
facf, 20
ftsmweightselect, 35
GAEVforecast, 36
H_estimate_LW_NFTS, 40
Horta_Ziegelmann_FPCA, 38
long_run_covariance_estimation, 45
LQDT_FPCA, 46
MAF_multivariate, 47
mean.fts, 48
median.fts, 50
quantile.fts, 65
sd.fts, 67
stop_time_detect, 71
var.fts, 76
∗ **models**
centre, 7
dynamic_FLR, 12
dynupdate, 14
extract, 19
farforecast, 21
forecast.ftsm, 24
forecastfplsr, 26
fplsr, 27
ftsm, 31
ftsmiterativeforecasts, 33
isfe.fts, 43
MFDM, 52
pcscorebootstrapdata, 54
plot.ftsm, 60
plotfplsr, 62
quantile, 64
residuals.fm, 65
sd, 66
skew_t_fun, 70
summary.fm, 73
T_stationary, 74
var, 75
∗ **multivariate**
fbootstrap, 22
∗ **package**
ftsa-package, 3
∗ **time series**
H_estimate_RS, 41
sim_FARMA, 69
∗ **ts**
diff.fts, 11
is.fts, 43

centre, 7
CoDa_BayesNW, 8
CoDa_FPCA, 9, 10, *40*, *47*, *71*
contour, *58*

diff.fts, 11
DJI_return, 12
dynamic_FLR, 12
dynupdate, *14*, 14

ER_GR, 18
error, 16
ets, *24*
extract, 19

facf, 20

farforecast, 21, *75*
fbootstrap, 22, *56*
fds, *21*
filled.contour, *58*
forecast.ftsm, *16*, *22*, 24, *27*, *31*, *33*, *36*, *44*, *54*, *57–59*, *62, 63*, *66*, *73*
forecastfplsr, *22*, *26*, 26, *54*
fplsr, 27, *54*, *57, 58*, *62*, *66*, *73*
fts, *45*
ftsa *(*ftsa-package*)*, 3
ftsa-package, 3
ftsm, *16*, *19*, *24–27*, *31*, 31, *34–36*, *44*, *48*, *54*, *57, 58*, *60–63*, *66*, *73*
ftsmiterativeforecasts, 33
ftsmweightselect, *33*, 35

GAEVforecast, 36

H_estimate_LW_NFTS, 40
H_estimate_RS, *41*, 41, *70*
Horta_Ziegelmann_FPCA, *11*, 38, *47*, *71*

image, *58*
is.fts, 43
isfe.fts, 43

long_run_covariance_estimation, 45
LQDT_FPCA, *11*, *40*, 46, *71*

MAF_multivariate, 47
mean.fts, *8*, 48, *52*, *65*, *69*, *77*
median.fts, *8*, *50*, 50, *65*, *69*, *77*
MFDM, 52

pcscorebootstrapdata, *8*, *23*, 54
persp, *58*
plot.fm, *16*, *26, 27*, *31*, *33*, *35*, *44*, 56, *58*, *60*, *62, 63*, *66*, *73*
plot.fmres, *31*, *44*, *57*, 58, *60*, *66*, *73*
plot.ftsf, *26, 27*, *33*, *35*, *57*, 59, *62, 63*
plot.ftsm, 60
plotfplsr, 62
pm_10_GR, 63
pm_10_GR_sqrt *(*pm_10_GR*)*, 63

quantile, *64*, 64
quantile.fts, *50*, *52*, *64*, 65, *69*, *77*

residuals.fm, *16*, *26, 27*, *31*, *33*, *35*, *44*, *57, 58*, *60*, *62, 63*, 65, *73*

sd, 66, *67*
sd.fts, *8*, *50*, *52*, *65*, *67*, 67, *77*
sim_FARMA, *42*, 69
skew_t_fun, *11*, *40*, *47*, 70
stop_time_detect, 71, *73*
stop_time_sim_data, *72*, 72
summary.fm, *16*, *26, 27*, *31*, *33*, *35*, *44*, *57, 58*, *60*, *62, 63*, *66*, 73

T_stationary, 74

VAR, *21*
var, *75*, 75
var.fts, *8*, *50*, *52*, *65*, *69*, *76*, 76