Package ‘fHMM’

July 7, 2022

Type Package

Title Fitting Hidden Markov Models to Financial Data

Version 1.0.3

Date 2022-07-07

Description Fitting (hierarchical) hidden Markov models to financial data via maximum likelihood estimation. See Oelschläger, L. and Adam, T. ``Detecting bearish and bullish markets in financial time series using hierarchical hidden Markov models'' (2021, Statistical Modelling) <doi:10.1177/1471082X211034048> for a reference.

Language en-US

URL `https://loelschlaeger.de/fHMM/`

BugReports `https://github.com/loelschlaeger/fHMM/issues`

License GPL-3

Encoding UTF-8

Depends R (>= 4.0.0)

Imports MASS, Rcpp, progress, foreach

LinkingTo Rcpp, RcppArmadillo

Suggests parallel, doSNOW, rmarkdown, knitr, testthat (>= 3.0.0), covr, printr, tseries, spelling

RoxygenNote 7.1.2

VignetteBuilder knitr

Config/testthat/edition 3

LazyData true

NeedsCompilation yes

Author Lennart Oelschläger [aut, cre] (<`https://orcid.org/0000-0001-5421-9313`>), Timo Adam [aut] (<`https://orcid.org/0000-0001-9079-3259`>), Rouven Michels [aut] (<`https://orcid.org/0000-0002-5433-6197`>)

Maintainer Lennart Oelschläger <`oelschlaeger.lennart@gmail.com`>

Repository CRAN

Date/Publication 2022-07-07 10:50:02 UTC

R topics documented:

`coef.fHMM_model` *Model coefficients*

Description

This function returns the estimated model coefficients and an `alpha` confidence interval.

Usage

```
## S3 method for class 'fHMM_model'
coef(object, alpha = 0.05, ...)
```

Arguments

`object`	An object of class `fHMM_model`.
`alpha`	The alpha level for the confidence interval, a numeric between 0 and 1. Per default, `alpha = 0.05`, which computes a 95% confidence interval.
`...`	Ignored.

Value

A `data.frame`.

`compare_models` *Comparing multiple* `fHMM_model`*-objects*

Description

This function compares multiple `fHMM_model` with respect to

- the number of model parameters,
- the log-likelihood value,
- the AIC value,
- the BIC value.

Usage

```
compare_models(...)
```

Arguments

`...`	A list of one or more objects of class `fHMM_model`.

Value

A data frame with models in rows and comparison criteria in columns.

Examples

```
data("dax_model_3t")
compare_models(dax_model_3t)
```

`compute_residuals` *Computing (pseudo-) residuals*

Description

This function computes (pseudo-) residuals of an `fHMM_model` object.

Usage

```
compute_residuals(x, verbose = TRUE)
```

Arguments

`x`	An object of class `fHMM_model`.
`verbose`	Set to `TRUE` to print progress messages.

Value

An object of class `fHMM_model` with residuals included.

Examples

```
data("dax_model_3t")
compute_residuals(dax_model_3t)
residuals(dax_model_3t)
```

`dax_model_2n` *DAX 2-state HMM*

Description

A pre-computed HMM on closing prices of the DAX from 2000 to 2021 with two hidden states and normal state-dependent distributions for demonstration purpose.

Usage

```
data("dax_model_2n")
```

Format

An object of class `fHMM_model`.

Details

The model was derived via specifying

```
controls <- list(
  states = 2,
  sdds   = "t(df = Inf)",
 data   = list(file       = system.file("extdata", "dax.csv", package = "fHMM"),
              date_column = "Date",
              data_column = "Close",
              logreturns  = TRUE,
              from        = "2000-01-03",
              to          = "2021-12-31"),
  fit    = list("runs" = 100)
  )
```

dax_model_3t *DAX 3-state HMM*

Description

A pre-computed HMM on closing prices of the DAX from 2000 to 2021 with three hidden states and state-dependent t-distributions for demonstration purpose.

Usage

```
data("dax_model_3t")
```

Format

An object of class `fHMM_model`.

Details

The model was derived via specifying

```
controls <- list(
  states = 3,
  sdds   = "t",
  data  = list(file       = system.file("extdata", "dax.csv", package = "fHMM"),
                date_column = "Date",
                data_column = "Close",
                logreturns  = TRUE,
                from        = "2000-01-03",
                to          = "2021-12-31"),
  fit    = list("runs" = 100)
  )
```

dax_vw_model *DAX/VW hierarchical HMM*

Description

A pre-computed HHMM with monthly averaged closing prices of the DAX from 2000 to 2021 on the coarse scale, VW stock data on the fine scale, two hidden fine-scale and coarse-scale states, respectively, and state-dependent t-distributions with degrees of freedom fixed to 1 for demonstration purpose.

Usage

```
data("dax_vw_model")
```

Format

An object of class fHMM_model.

Details

The model was derived via specifying

```
controls <- list(
  hierarchy = TRUE,
  states    = c(2,2),
  sdds      = c("t(df = 1)", "t(df = 1)"),
  period    = "m",
  data     = list(file = c(system.file("extdata", "dax.csv", package = "fHMM"),
                           system.file("extdata", "vw.csv", package = "fHMM")),
                   from = "2015-01-01",
                   to = "2020-01-01",
                   logreturns = c(TRUE,TRUE))
)
```

decode_states *Decoding the underlying hidden state sequence*

Description

This function decodes the (most likely) underlying hidden state sequence by applying the Viterbi algorithm.

Usage

```
decode_states(x, verbose = TRUE)
```

Arguments

x	An object of class fHMM_model.
verbose	Set to TRUE to print progress messages.

Value

An object of class fHMM_model with decoded state sequence included.

References

https://en.wikipedia.org/wiki/Viterbi_algorithm

Examples

```
data("dax_model_3t")
decode_states(dax_model_3t)
```

download_data *Downloading financial data*

Description

This function downloads stock data from https://finance.yahoo.com/ and saves it as a .csv-file.

Usage

```
download_data(
  symbol,
  from = "1902-01-01",
  to = Sys.Date(),
  file = paste0(symbol, ".csv"),
  verbose = TRUE
)
```

Arguments

symbol	A character, the stock's symbol. It must match the identifier on https://finance.yahoo.com/.
from	A date in format "YYYY-MM-DD", setting the lower data bound. Must not be earlier than "1902-01-01".
to	A date in format "YYYY-MM-DD", setting the upper data bound. Default is the current date Sys.date().
file	The name of the file where the .csv-file is saved. Per default, it is saved in the current working directory with the name "symbol.csv".
verbose	If TRUE returns information about download success.

Details

The downloaded data is a .csv-file with the following columns:

- Date: The date.
- Open: Opening price.
- High: Highest price.
- Low: Lowest price.
- Close: Close price adjusted for splits.
- Adj.Close: Close price adjusted for dividends and splits.
- Volume: Trade volume.

Value

No return value.

Examples

```
### download 21st century DAX data
download_data(
  symbol = "^GDAXI", from = "2000-01-03",
  file = paste0(tempfile(), ".csv")
)
```

fHMM_events *Checking events*

Description

This function checks the input `events`.

Usage

```
fHMM_events(events)
```

Arguments

`events` A list of two elements. The first element is named `"dates"` and contains characters in format "YYYY-MM-DD". The second element is named `"labels"` and is a character vector of the same length as `"dates"`.

Value

An object of class `fHMM_events`.

Examples

```
events <- list(
  dates = c("2001-09-11", "2008-09-15", "2020-01-27"),
  labels = c(
    "9/11 terrorist attack", "Bankruptcy Lehman Brothers",
    "First COVID-19 case Germany"
  )
)
events <- fHMM_events(events)
```

fHMM_parameters *Setting and checking model parameters*

Description

This function sets and checks model parameters for the fHMM package.

Usage

```
fHMM_parameters(
  controls,
  Gamma = NULL,
  mus = NULL,
  sigmas = NULL,
  dfs = NULL,
  Gammas_star = NULL,
  mus_star = NULL,
  sigmas_star = NULL,
  dfs_star = NULL,
  seed = NULL,
  scale_par = c(1, 1)
)
```

Arguments

`controls`	An object of class `fHMM_controls`.
`Gamma`	A tpm (transition probability matrix) of dimension `controls$states[1]`.
`mus`	A vector of expectations of length `controls$states[1]`.
`sigmas`	A vector of standard deviations of length `controls$states[1]`.
`dfs`	A vector of degrees of freedom of length `controls$states[1]`. Only relevant if sdd is a t-distribution.
`Gammas_star`	A list of length `controls$states[1]` of (fine-scale) tpm's. Each tpm must be of dimension `controls$states[2]`.
`mus_star`	A list of length `controls$states[1]` of vectors of (fine-scale) expectations. Each vector must be of length `controls$states[2]`.
`sigmas_star`	A list of length `controls$states[1]` of vectors of standard deviations. Each vector must be of length `controls$states[2]`.
`dfs_star`	A list of length `controls$states[1]` of vectors of (fine-scale) degrees of freedom. Each vector must be of length `controls$states[2]`. Only relevant if sdd is a t-distribution.
`seed`	Set a seed for the sampling of parameters.
`scale_par`	A positive numeric vector of length two, containing scales for sampled expectations and standard deviations. The first entry is the scale for `mus` and `sigmas`, the second entry is the scale for `mus_star` and `sigmas_star`. Set an entry to `1` for no scaling.

Details

See the vignette on the model definition for more details.

Value

An object of class `fHMM_parameters`.

Examples

```
controls <- set_controls()
fHMM_parameters(controls)
```

`fit_model` *Model fitting*

Description

This function fits a HMM to `data` via maximum likelihood estimation.

Usage

```
fit_model(data, ncluster = 1, seed = NULL, verbose = TRUE, init = NULL)
```

Arguments

`data`	An object of class `fHMM_data`.
`ncluster`	Set the number of clusters for parallelization.
`seed`	Set a seed for the sampling of initial values.
`verbose`	Set to `TRUE` to print progress messages.
`init`	Optionally an object of class `parUncon` for initialization. This can for example be the estimate of a previously fitted model `model`, i.e. the element `model$estimate`. The initial values are computed via `replicate(n, jitter(init, amount = 1), simplify = FALSE)`, where `n <- data$controls$fit$runs`.

Details

The function is parallelized only if `ncluster > 1`.

Value

An object of class `fHMM_model`.

`npar` *Number of model parameters*

Description

This function extracts the number of model parameters of an `fHMM_model` object.

Usage

```
npar(object, ...)

## S3 method for class 'fHMM_model'
npar(object, ...)
```

Arguments

`object`	An object of class `fHMM_model`.
`...`	Optionally more objects of class `fHMM_model`.

Value

Either a numeric value (if just one object is provided) or a numeric vector.

Examples

```
data("dax_model_3t", package = "fHMM")
data("dax_model_2n", package = "fHMM")
npar(dax_model_3t, dax_model_2n)
```

`plot.fHMM_data` *Plot method for an object of class* `fHMM_data`

Description

This function is the plot method for an object of class `fHMM_data`.

Usage

```
## S3 method for class 'fHMM_data'
plot(x, events = NULL, ...)
```

Arguments

`x`	An object of class `fHMM_data`.
`events`	Either `NULL` or an object of class `fHMM_events`.
`...`	Ignored.

Value

No return value. Draws a plot to the current device.

`plot.fHMM_model` *Plot method for an object of class* `fHMM_model`

Description

This function is the plot method for an object of class `fHMM_model`.

Usage

```
## S3 method for class 'fHMM_model'
plot(x, plot_type = "ts", events = NULL, colors = NULL, ...)
```

Arguments

`x` An object of class `fHMM_model`.

`plot_type` A character (vector), specifying the type of plot and can be one (or more) of

- `"ll"` for a visualization of the likelihood values in the different optimization runs,
- `"sdds"` for a visualization of the estimated state-dependent distributions,
- `"pr"` for a visualization of the model's (pseudo-) residuals,
- `"ts"` for a visualization of the financial time series.

`events` An object of class `fHMM_events`.

`colors` Either `NULL` or a character vector of color names or hexadecimal RGB triplets.

`...` Ignored.

Value

No return value. Draws a plot to the current device.

`predict.fHMM_model` *Prediction*

Description

This function predicts the next `ahead` states and data points based on an `fHMM_model` object.

Usage

```
## S3 method for class 'fHMM_model'
predict(object, ahead = 5, alpha = 0.05, ...)
```

Arguments

`object`	An object of class `fHMM_model`.
`ahead`	A positive integer, the forecast horizon.
`alpha`	The alpha level for the confidence interval, a numeric between 0 and 1. Per default, `alpha = 0.05`, which computes a 95% confidence interval.
`...`	Ignored.

Value

An data frame of state probabilities and data point estimates along with confidence intervals.

Examples

```
data("dax_model_3t")
predict(dax_model_3t)
```

`prepare_data` *Prepare data*

Description

This function simulates or reads financial data for the fHMM package.

Usage

```
prepare_data(controls, true_parameters = NULL, seed = NULL)
```

Arguments

`controls`	An object of class `fHMM_controls`.
`true_parameters`	An object of class `fHMM_parameters`, used as simulation parameters.
`seed`	Set a seed for the data simulation.

Value

An object of class `fHMM_data`, which is a list containing the following elements:

- The matrix of the `dates` if `simulated = FALSE` and `controls$data$data_column` is specified,
- the matrix of the `time_points` if `simulated = TRUE` or `controls$data$data_column` is not specified,
- the matrix of the simulated `markov_chain` if `simulated = TRUE`,
- the matrix of the simulated or empirical `data` used for estimation,

- the matrix `time_series` of empirical data before the transformation to log-returns if `simulated = FALSE`,
- the vector of fine-scale chunk sizes `T_star` if `controls$hierarchy = TRUE`,
- the input `controls`,
- the `true_parameters`.

Examples

```
controls <- set_controls()
prepare_data(controls)
```

`reorder_states` *Reordering of estimated states*

Description

This function reorders the estimated states, which can be useful for a comparison to true parameters or the interpretation of states.

Usage

```
reorder_states(x, state_order)
```

Arguments

`x` An object of class `fHMM_model`.

`state_order` A vector or a matrix which determines the new ordering.

- If `x$data$controls$hierarchy = FALSE`, `state_order` must be a vector of length `x$data$controls$states` with integer values from `1` to `x$data$controls$states`. If the old state number `x` should be the new state number `y`, put the value `x` at the position `y` of `state_order`. E.g. for a 2-state HMM, specifying `state_order = c(2,1)` swaps the states.
- If `x$data$controls$hierarchy = TRUE`, `state_order` must be a matrix of dimension `x$data$controls$states[1]` x `x$data$controls$states[2] + 1`. The first column orders the coarse-scale states with the logic as described above. For each row, the elements from second to last position order the fine-scale states of the coarse-scale state specified by the first element. E.g. for an HHMM with 2 coarse-scale and 2 fine-scale states, specifying `state_order = matrix(c(2,1,2,1,1,2),2,3)` swaps the coarse-scale states and the fine-scale states of coarse-scale state 2.

Value

An object of class `fHMM_model`, in which states are reordered.

Examples

```
data("dax_model_3t")
reorder_states(dax_model_3t, state_order = 3:1)
```

residuals.fHMM_model *Residuals*

Description

This function extracts the computed (pseudo-) residuals of an fHMM_model object.

Usage

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

Arguments

object	An object of class fHMM_model.
...	Ignored.

Value

A vector (or a matrix, in case of an hierarchical HMM) with (pseudo-) residuals for each observation.

Examples

```
data("dax_model_3t")
compute_residuals(dax_model_3t)
residuals(dax_model_3t)
```

set_controls *Set and check controls*

Description

This function sets and checks the specification of controls for the fHMM package.

Usage

```
set_controls(controls = NULL)
```

Arguments

controls

A list of controls. Either none, all, or selected parameters can be specified. Unspecified parameters are set to default values (the values in brackets). If hierarchy = TRUE, parameters with a (*) must be a vector of length 2, where the first entry corresponds to the coarse-scale and the second entry to the fine-scale layer.

- hierarchy (FALSE): A boolean, set to TRUE for an hierarchical HMM.
- states (*) (2): The number of states of the underlying Markov chain.
- sdds (*) ("t(df = Inf)"): Specifying the state-dependent distribution, one of "t", or "gamma" (the gamma distribution), or "lnorm" (the log-normal distribution). You can fix the parameters (mean mu, standard deviation \codesigma, degrees of freedom df) of these distributions, e.g. "t(df = Inf)" or "gamma(mu = 0, sigma = 1)", respectively. To fix different values of one parameter for different states, separate by "|", e.g. "t(mu = -1|1)".
- horizon (*) (100): A numeric, specifying the length of the time horizon. The first entry of horizon is ignored if data is specified.
- period ("m"): Only relevant if hierarchy = TRUE and horizon[2] = NA_integer_. In this case, it specifies a flexible, periodic fine-scale time horizon and can be one of
  - "w" for a week,
  - "m" for a month,
  - "q" for a quarter,
  - "y" for a year.
- data (NA): A list of controls specifying the data. If data = NA, data gets simulated. Otherwise:
  - file (*): A character, the path to a .csv-file with financial data, which must have a column named date_column (with dates) and data_column (with financial data).
  - date_column (*) ("Date"): A character, the name of the column in file with dates. Can be NA_character_ in which case consecutive integers are used as time points.
  - data_column (*) ("Close"): A character, the name of the column in file with financial data.
  - from (NA_character_): A character of the format "YYYY-MM-DD", setting a lower data limit. No lower limit if from = NA_character_. Ignored if controls$data$date_column is NA.
  - to (NA_character_): A character of the format "YYYY-MM-DD", setting an upper data limit. No upper limit if from = NA_character_. Ignored if controls$data$date_column is NA_character_.
  - logreturns (*) (FALSE): A boolean, if TRUE the data is transformed to log-returns.
  - merge (function(x) mean(x)): Only relevant if hierarchy = TRUE. In this case, a function, which merges a numeric vector of fine-scale data x into one coarse-scale observation. For example,
    - merge = function(x) mean(x) defines the mean of the fine-scale data as the coarse-scale observation,

    - `merge = function(x) mean(abs(x))` for the mean of the absolute values,
    - `merge = function(x) (abs(x))` for the sum of of the absolute values,
    - `merge = function(x) (tail(x,1)-head(x,1))/head(x,1)` for the relative change of the first to the last fine-scale observation.
- `fit`: A list of controls specifying the model fitting:
  - `runs` (`100`): An integer, setting the number of optimization runs.
  - `origin` (`FALSE`): A boolean, if `TRUE` the optimization is initialized at the true parameter values. Only for simulated data. If `origin = TRUE`, this sets `run = 1` and `accept = 1:5`.
  - `accept` (`1:3`): An integer (vector), specifying which optimization runs are accepted based on the output code of `nlm`.
  - `gradtol` (`1e-6`): A positive numeric value, passed on to `nlm`.
  - `iterlim` (`200`): A positive integer, passed on to `nlm`.
  - `print.level` (`0`): One of `0`, `1`, and `2`, passed on to `nlm`.
  - `steptol` (`1e-6`): A positive numeric value, passed on to `nlm`.

Details

See the vignettes for more information on how to specify `controls`.

Value

An object of class `fHMM_controls`.

Examples

```
### HMM controls
controls <- list(
  states  = 2,
  sdds    = "t(mu = 0, sigma = 1, df = 1)",
  horizon = 400,
  fit     = list("runs" = 50)
)
set_controls(controls)

### HHMM controls
controls <- list(
  hierarchy = TRUE
)
set_controls(controls)
```

`sim_model_2gamma` *Simulated 2-state HMM*

Description

A pre-computed 2-state HMM with state-dependent gamma distributions with means fixed to `0.5` and `2` on `500` simulated observations.

Usage

```
data("sim_model_2gamma")
```

Format

An object of class `fHMM_model`.

Details

The model was estimated via:

```
controls <- list(
  states  = 2,
  sdds    = "gamma(mu = 1|2)",
  horizon = 200,
  fit     = list(runs = 50)
)
controls <- set_controls(controls)
pars <- fHMM_parameters(
  controls = controls, Gamma = matrix(c(0.9,0.2,0.1,0.8), nrow = 2),
  sigmas = c(0.5,1)
)
data <- prepare_data(controls, true_parameters = pars, seed = 1)
sim_model_2gamma <- fit_model(data, seed = 1, verbose = FALSE)
```

Index