# Package 'fHMM'

October 12, 2023

**Type** Package

**Title** Fitting Hidden Markov Models to Financial Data

**Version** 1.1.1

**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, cli

**LinkingTo** Rcpp, RcppArmadillo

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

**RoxygenNote** 7.2.3

**VignetteBuilder** knitr

**Config/testthat/edition** 3

**LazyData** true

**LazyDataCompression** xz

**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** 2023-10-12 14:30:13 UTC

# R topics documented:

coef.fHMM_model . . . 2
compare_models . . . 3
compute_residuals . . . 4
dax . . . 4
dax_model_2n . . . 5
dax_model_3t . . . 6
dax_vw_model . . . 7
decode_states . . . 8
download_data . . . 8
fHMM_data . . . 9
fHMM_events . . . 11
fHMM_model . . . 12
fHMM_parameters . . . 13
fit_model . . . 14
npar . . . 15
plot.fHMM_data . . . 16
plot.fHMM_model . . . 16
predict.fHMM_model . . . 18
prepare_data . . . 18
reorder_states . . . 19
residuals.fHMM_model . . . 20
set_controls . . . 20
sim_model_2gamma . . . 24
sim_model_4lnorm . . . 24
spx . . . 25
unemp . . . 26
unemp_spx_model_3_2 . . . 27
vw . . . 28

**Index** **29**

---

`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, digits = 2, ...)
```

### Arguments

| | |
|---|---|
| object | An object of class fHMM_model. |
| alpha | A numeric between 0 and 1, the alpha level for the confidence interval. By default, alpha = 0.05, which computes a 95% confidence interval. |
| digits | An integer, the number of significant digits to be used. By default, digits = 2. |
| ... | Ignored. |

### Value

A data.frame.

---

## compare_models *Compare multiple models*

---

### Description

This function performs model comparison by comparing multiple fHMM_model objects 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

```
### 3-state HMM with t-distributions is preferred over 2-state HMM with
### normal distributions for the DAX data based on AIC and BIC
compare_models(dax_model_2n, dax_model_3t)
```

## compute_residuals *Compute (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 (default) to print progress messages. |

### Value

An object of class fHMM_model with residuals included.

### Examples

```
compute_residuals(dax_model_3t)
res <- residuals(dax_model_3t)
summary(res)
```

## dax *Deutscher Aktienindex (DAX) index data*

### Description

Deutscher Aktienindex (DAX) index data from 1988 to 2022 from Yahoo Finance.

### Usage

```
dax
```

## Format

A `data.frame` with 9012 rows and the following 7 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.

## Details

The data was obtained via:

```
dax <- download_data(
  symbol = "^GDAXI",   # DAX identifier on Yahoo Finance
  from = "1988-01-01", # first observation
  to = "2022-12-31"    # last observation
)
```

The data is also available as `.csv` file via:

```
system.file("extdata", "dax.csv", package = "fHMM")
```

## Source

https://finance.yahoo.com/quote/%5EGDAXI

---

# dax_model_2n *DAX 2-state HMM with normal distributions*

---

## Description

A pre-computed HMM on closing prices of the DAX from 2000 to 2022 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 estimated via:

```
controls <- list(
  states = 2,
  sdds   = "t(df = Inf)",
  data   = list(
    file        = dax,
    date_column = "Date",
    data_column = "Close",
    logreturns  = TRUE,
    from        = "2000-01-03",
    to          = "2022-12-31"
  ),
  fit    = list(runs = 100)
)
controls <- set_controls(controls)
dax_data <- prepare_data(controls)
dax_model_2n <- fit_model(dax_data)
```

---

`dax_model_3t` *DAX 3-state HMM with t-distributions*

---

## Description

A pre-computed HMM on closing prices of the DAX from 2000 to 2022 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 estimated via:

```
controls <- list(
  states = 3,
  sdds   = "t",
  data   = list(
    file        = dax,
    date_column = "Date",
    data_column = "Close",
```

```
    logreturns  = TRUE,
    from        = "2000-01-03",
    to          = "2022-12-31"
  ),
  fit    = list(runs = 200)
)
controls <- set_controls(controls)
dax_data <- prepare_data(controls)
dax_model_3t <- fit_model(dax_data)
```

---

dax_vw_model *DAX/VW hierarchical HMM with t-distributions*

---

## Description

A pre-computed HHMM with monthly averaged closing prices of the DAX from 2010 to 2022 on the coarse scale, Volkswagen AG stock data on the fine scale, two hidden fine-scale and coarse-scale states, respectively, and state-dependent t-distributions for demonstration purpose.

## Usage

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

## Format

An object of class `fHMM_model`.

## Details

The model was estimated via:

```
controls <- list(
  hierarchy = TRUE,
  states    = c(2, 2),
  sdds      = c("t", "t"),
  period    = "m",
  data      = list(
    file       = list(dax, vw),
    from       = "2010-01-01",
    to         = "2022-12-31",
    logreturns = c(TRUE, TRUE)
  ),
  fit       = list(
    runs = 200
  )
)
controls <- set_controls(controls)
dax_vw_data <- prepare_data(controls)
dax_vw_model <- fit_model(dax_vw_data)
```

## decode_states *Decode the underlying hidden state sequence*

### Description

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

### 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

```
decode_states(dax_model_3t)
plot(dax_model_3t, type = "ts")
```

## download_data *Download financial data from Yahoo Finance*

### Description

This function downloads stock data from https://finance.yahoo.com/.

### Usage

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

### Arguments

| | |
|---|---|
| symbol | A character, the stock's symbol. It must match the identifier on https://finance.yahoo.com/. |
| from | A character, a date in format "YYYY-MM-DD", setting the lower data bound. Must not be earlier than "1902-01-01" (default). |
| to | A character, a date in format "YYYY-MM-DD", setting the upper data bound. Default is the current date Sys.date(). |
| file | Either<br>• NULL (default) to return the data as a data.frame,<br>• or a character, the name of the file where the data is saved as a .csv-file. |
| verbose | Set to TRUE to return information about download success. |

### Details

The downloaded data has 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

A data.frame if file = NULL.

### Examples

```
### download 21st century DAX data
data <- download_data(symbol = "^GDAXI", from = "2000-01-03")
head(data)
```

---

## fHMM_data *Constructor of an* fHMM_data *object*

---

### Description

This function constructs an object of class fHMM_data, which contains the financial data for modeling.

## Usage

```
fHMM_data(
  dates,
  time_points,
  markov_chain,
  data,
  time_series,
  T_star,
  controls,
  true_parameters
)

## S3 method for class 'fHMM_data'
print(x, ...)

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

## Arguments

| | |
|---|---|
| dates | The dates in the empirical case. |
| time_points | The time points in the simulated case. |
| markov_chain | The states in the simulated case. |
| data | The data for modeling. |
| time_series | The data before transformation. |
| T_star | The fine-scale chunk sizes. |
| controls | The fHMM_controls object. |
| true_parameters | The fHMM_parameters object in the simulated case. |
| x | An object of class fHMM_data. |
| ... | Currently not used. |
| object | An object of class fHMM_data. |

## 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.

---

fHMM_events *Checking events*

---

## Description

This function checks the input events.

## Usage

```
fHMM_events(events)

## S3 method for class 'fHMM_events'
print(x, ...)
```

## Arguments

| | |
|---|---|
| events | A list of two elements.<br>• The first element is named "dates" and contains a character vector in format "YYYY-MM-DD".<br>• The second element is named "labels" and is a character vector of the same length as "dates". |
| x | An object of class fHMM_events. |
| ... | Currently not used. |

## 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_model` *Constructor of a model object*

## Description

This function constructs an object of class `fHMM_model`, which contains details about the fitted (hierarchical) Hidden Markov model.

## Usage

```
fHMM_model(
  data,
  estimate,
  nlm_output,
  estimation_time,
  ll,
  lls,
  gradient,
  hessian,
  decoding
)
```

## Arguments

| | |
|---|---|
| `data` | An object of class `fHMM_data`. |
| `estimate` | A `numeric` vector of unconstrained model estimates. |
| `nlm_output` | The output of `nlm` for the selected optimization run. |
| `estimation_time` | A `diff.time` object, the total estimation time. |
| `ll` | A `numeric`, the model log-likelihood. |
| `lls` | A `numeric` vector, the model log-likelihoods in all optimization runs. |
| `gradient` | A `numeric` vector, the gradient at the optimum. |
| `hessian` | A `matrix`, the Hessian at the optimum. |
| `decoding` | A `numeric` vector, the decoded time series. |

## Value

An object of class `fHMM_model`.

fHMM_parameters *Set and check 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)
)

## S3 method for class 'fHMM_parameters'
print(x, ...)
```

## Arguments

| | |
|---|---|
| controls | An object of class fHMM_controls. |
| Gamma | A matrix, a tpm (transition probability matrix) of dimension controls$states[1]. |
| mus | A numeric vector of expectations of length controls$states[1]. |
| sigmas | A numeric vector of standard deviations of length controls$states[1]. |
| dfs | A numeric vector of degrees of freedom of length controls$states[1]. Only relevant in case of a state-dependent 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$states[1] of numeric vectors of (fine-scale) expectations. Each vector must be of length controls$states[2]. |
| sigmas_star | A list of length controls$states[1] of numeric vectors of standard deviations. Each vector must be of length controls$states[2]. |
| dfs_star | A list of length controls$states[1] of numeric vectors of (fine-scale) degrees of freedom. Each vector must be of length controls$states[2]. Only relevant in case of a state-dependent t-distribution. |
| seed | Set a seed for the sampling of parameters. No seed per default. |

| | |
|---|---|
| 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. |
| x | An object of class fHMM_parameters. |
| ... | Currently not used. |

## 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 fHMM_data via numerical likelihood maximization.

## Usage

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

## S3 method for class 'fHMM_model'
print(x, ...)
```

## Arguments

| | |
|---|---|
| data | An object of class fHMM_data. |
| ncluster | Set the number of clusters for parallelization. By default, ncluster = 1. |
| seed | Set a seed for the sampling of initial values. No seed by default. |
| 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. |
| x | An object of class fHMM_model. |
| ... | Currently not used. |

### Details

The function is parallelized 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

```
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, title = NULL, from = NULL, to = NULL, ...)
```

### Arguments

| | |
|---|---|
| x | An object of class fHMM_data. |
| events | An object of class fHMM_events. |
| title | Optionally a character for a custom title. |
| from | Optionally a character, a date in format "YYYY-MM-DD", setting the lower date bound for plotting. By default, from = NULL, i.e. no lower bound. |
| to | Optionally a character, a date in format "YYYY-MM-DD", setting the upper date bound for plotting. By default, to = NULL, i.e. no upper bound. |
| ... | Currently not used. |

### Value

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

### Examples

```
plot(dax_model_3t$data, title = "DAX time series")
```

## 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,
  ll_relative = TRUE,
  title = NULL,
  from = NULL,
  to = 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<br>• "ll" for a visualization of the likelihood values in the different optimization runs,<br>• "sdds" for a visualization of the estimated state-dependent distributions,<br>• "pr" for a visualization of the model's (pseudo-) residuals,<br>• "ts" for a visualization of the financial time series. |
| events | An object of class fHMM_events. |
| colors | Either NULL (default) or a character vector of color names or hexadecimal RGB triplets. |
| ll_relative | A logical, set to TRUE (default) to plot the differences from the best log-likelihood value. Set to FALSE to plot the absolute values. |
| title | Optionally a character for a custom title. |
| from | Optionally a character, a date in format "YYYY-MM-DD", setting the lower date bound for plotting. By default, from = NULL, i.e. no lower bound. |
| to | Optionally a character, a date in format "YYYY-MM-DD", setting the upper date bound for plotting. By default, to = NULL, i.e. no upper bound. |
| ... | Currently not used. |

## 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 | A numeric between 0 and 1, the alpha level for the confidence interval. By default, alpha = 0.05, which computes a 95% confidence interval. |
| ... | Ignored. |

### Value

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

### Examples

```
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. By default, true_parameters = NULL, i.e., sampled true parameters. |
| seed | Set a seed for the data simulation. No seed per default. |

## Value

An object of class fHMM_data.

## Examples

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

---

`reorder_states` *Reorder 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.<br>• 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$states. If the old state number x should be the new state number y, put the value x at the position y of state_order. E.g. for a 2-state HMM, specifying state_order = c(2,1) swaps the states.<br>• If x$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

```
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

```
compute_residuals(dax_model_3t)
res <- residuals(dax_model_3t)
head(res)
```

## set_controls *Set and validate controls*

### Description

This function sets and validates the specification of controls for model estimation with the {fHMM} package.

### Usage

```
set_controls(controls = NULL)

## S3 method for class 'fHMM_controls'
print(x, ...)
```

## Arguments

`controls` A `list` of controls, see below.

Either none, all, or selected parameters can be specified. Unspecified parameters are set to default values (see the values in brackets below).

If `hierarchy = TRUE`, parameters marked 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 `logical`, set to `TRUE` for an hierarchical HMM.
- `states` `(*)` (`2`): An `integer`, the number of states of the underlying Markov chain.
- `sdds` `(*)` (`"t(df = Inf)"`): A `character`, specifying the state-dependent distribution. One of `"t"` (the t-distribution), or `"gamma"` (the gamma distribution), or `"lnorm"` (the log-normal distribution). You can fix the parameters (mean `mu`, standard deviation `sigma`, degrees of freedom `df`) of these distributions via, e.g., `"t(df = Inf)"` or `"gamma(mu = 0, sigma = 1)"`. To fix different values of a 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`. In this case, a `character` which 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 (default). Otherwise:
  - `file` `(*)`: Either:
    - A `data.frame`, which must have a column named `date_column` (with dates) and `data_column` (with financial data). If `hierarchy = TRUE`, this `data.frame` is used for both the coarse- and the fine-scale layer. To have different data sets for theses layers, `file` can be a `list` of two `data.frame`.
    - 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` 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`): A `character` of the format `"YYYY-MM-DD"`, setting a lower data limit. No lower limit if `from = NA`. Ignored if `controls$data$date_column` is `NA`.

  - `to` (`NA`): A `character` of the format `"YYYY-MM-DD"`, setting an upper data limit. No upper limit if `from = NA`. Ignored if `controls$data$date_column` is `NA`.
  - `logreturns` (`*`) (`FALSE`): A `logical`, if `TRUE` the data is transformed to log-returns.
  - `merge` (`function(x) mean(x)`): Only relevant if `hierarchy = TRUE`. In this case, a `function` with one argument `x`, 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) sum(abs(x))` for the sum 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 randomly initialized optimization runs from which the best one is selected as the final model.
  - `origin` (`FALSE`): A `logical`, 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` to control the verbosity of the optimization, passed on to `nlm`.
  - `steptol` (`1e-6`): A positive `numeric` value, passed on to `nlm`.

`x` An object of class `fHMM_controls`.

`...` Currently not used.

## Details

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

## Value

An object of class `fHMM_controls`.

## Examples

```
### HMM controls for simulation
controls <- list(
  states  = 2,
  sdds    = "t(mu = 0)",
```

```
  fit     = list("runs" = 50)
)
set_controls(controls)

### HMM controls with empirical data
data <- download_data("^GDAXI", file = NULL)
controls <- list(
  states  = 3,
  sdds    = "lnorm",
  data    = list(
    "file"        = data,
    "date_column" = "Date",
    "data_column" = "Adj.Close"
  )
)
set_controls(controls)

### HMM controls with empirical data from .csv-file
controls <- list(
  states  = 4,
  sdds    = "t",
  data    = list(
    "file"        = system.file("extdata", "dax.csv", package = "fHMM"),
    "date_column" = "Date",
    "data_column" = "Close",
    "logreturns"  = TRUE
  )
)
set_controls(controls)

### HHMM controls for simulation
controls <- list(
  hierarchy = TRUE,
  states    = c(3, 2)
)
set_controls(controls)

### HHMM controls with empirical data
controls <- list(
  hierarchy = TRUE,
  states  = c(3, 2),
  sdds    = c("t", "t"),
  data    = list(
    "file"        = list(dax, vw),
    "date_column" = c("Date", "Date"),
    "data_column" = c("Close", "Close"),
    "logreturns"  = c(TRUE, TRUE)
  )
)
set_controls(controls)
```

## sim_model_2gamma *Simulated 2-state HMM with gamma distributions*

### 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_sim <- prepare_data(controls, true_parameters = pars, seed = 1)
sim_model_2gamma <- fit_model(data_sim, seed = 1)
```

## sim_model_4lnorm *Simulated 4-state HMM with log-normal distributions*

### Description

A pre-computed 4-state HMM with state-dependent log-normal distributions on `1000` simulated observations.

### Usage

```
data("sim_model_4lnorm")
```

## Format

An object of class fHMM_model.

## Details

The model was estimated via:

```
controls <- list(
  states  = 4,
  sdds    = "lnorm",
  horizon = 1000,
  fit     = list(runs = 50)
)
controls <- set_controls(controls)
data_sim <- prepare_data(controls, seed = 1)
sim_model_4lnorm <- fit_model(data_sim, seed = 1)
```

---

# spx *Standard & Poor's 500 (S&P 500) index data*

---

## Description

Standard & Poor's 500 (S&P 500) index data from 1928 to 2022 from Yahoo Finance.

## Usage

```
spx
```

## Format

A data.frame with 23864 rows and the following 7 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.

## Details

The data was obtained via:

```
spx <- download_data(
  symbol = "^GSPC",    # S&P 500 identifier on Yahoo Finance
  from = "1928-01-01", # first observation
  to = "2022-12-31"    # last observation
)
```

The data is also available as `.csv` file via:

```
system.file("extdata", "spx.csv", package = "fHMM")
```

## Source

https://finance.yahoo.com/quote/%5EGSPC

---

# unemp *Unemployment rate data USA*

---

## Description

The monthly unemployment rate in the USA from 1955 to 2022 on a daily observation basis.

## Usage

```
unemp
```

## Format

A `data.frame` with 24806 rows and the following 3 columns:

- `date`: The date.
- `rate`: The unemployment rate.
- `rate_diff`: The difference rate to previous month.

## Source

OECD (2023), Unemployment rate (indicator). doi: 10.1787/52570002-en (Accessed on 18 January 2023) https://data.oecd.org/unemp/unemployment-rate.htm

`unemp_spx_model_3_2` *Unemployment rate and S&P 500 hierarchical HMM*

## Description

A pre-computed HHMM with monthly unemployment rate in the US on the coarse scale using 3 states and S&P 500 index data on the fine scale using 2 states from 1970 to 2020 for demonstration purpose.

## Usage

```
data("unemp_spx_model_3_2")
```

## Format

An object of class `fHMM_model`.

## Details

The model was estimated via:

```
controls <- list(
  hierarchy = TRUE,
  states    = c(3, 2),
  sdds      = c("t", "t"),
  period    = "m",
  data      = list(
    file       = list(unemp, spx),
    date_column = c("date", "Date"),
    data_column = c("rate_diff", "Close"),
    from       = "1970-01-01",
    to         = "2020-01-01",
    logreturns = c(FALSE, TRUE)
  ),
  fit       = list(
    runs    = 200,
    iterlim = 300
  )
)
controls <- set_controls(controls)
unemp_spx_data <- prepare_data(controls)
unemp_spx_model_3_2 <- fit_model(unemp_spx_data)
```

vw *Volkswagen AG (VW) stock data*

## Description

Volkswagen AG (VW) stock data from 1998 to 2022 from Yahoo Finance.

## Usage

```
vw
```

## Format

A `data.frame` with 6260 rows and the following 7 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.

## Details

The data was obtained via:

```
vw <- download_data(
  symbol = "VOW3.DE",  # Volkswagen AG identifier on Yahoo Finance
  from = "1988-07-22", # first observation
  to = "2022-12-31"    # last observation
)
```

The data is also available as `.csv` file via:

```
system.file("extdata", "vw.csv", package = "fHMM")
```

## Source

https://finance.yahoo.com/quote/vow3.de

# Index


∗ **data**
  dax, 4
  spx, 25
  unemp, 26
  vw, 28
∗ **model**
  dax_model_2n, 5
  dax_model_3t, 6
  dax_vw_model, 7
  sim_model_2gamma, 24
  sim_model_4lnorm, 24
  unemp_spx_model_3_2, 27

coef.fHMM_model, 2
compare_models, 3
compute_residuals, 4

dax, 4
dax_model_2n, 5
dax_model_3t, 6
dax_vw_model, 7
decode_states, 8
download_data, 8

fHMM_data, 9, *12*, *14*, *19*
fHMM_events, 11, *16, 17*
fHMM_model, *3–8*, *12*, 12, *14–20*, *24, 25*, *27*
fHMM_parameters, 13
fit_model, 14

nlm, *12*, *22*
npar, 15

plot.fHMM_data, 16
plot.fHMM_model, 16
predict.fHMM_model, 18
prepare_data, 18
print.fHMM_controls *(*set_controls*)*, 20
print.fHMM_data *(*fHMM_data*)*, 9
print.fHMM_events *(*fHMM_events*)*, 11
print.fHMM_model *(*fit_model*)*, 14
print.fHMM_parameters
  *(*fHMM_parameters*)*, 13

reorder_states, 19
residuals.fHMM_model, 20

set_controls, 20
sim_model_2gamma, 24
sim_model_4lnorm, 24
spx, 25
summary.fHMM_data *(*fHMM_data*)*, 9

unemp, 26
unemp_spx_model_3_2, 27

vw, 28