Package 'finnts'

August 27, 2026

Title Microsoft Finance Time Series Forecasting Framework

Version 0.7.0

Description
Automated time series forecasting developed by Microsoft Finance. The Microsoft Finance Time Series Forecasting Framework, aka Finn, can be used to forecast any component of the income statement, balance sheet, or any other area of interest by finance. Any numerical quantity over time,
Finn can be used to forecast it. While it can be applied outside of the finance domain, Finn was built
to meet the needs of financial analysts to better forecast their businesses within a company, and has a lot of built in features that are specific to the needs of financial forecasters. Happy forecasting!

URL https://microsoft.github.io/finnts/,
https://github.com/microsoft/finnts

BugReports https://github.com/microsoft/finnts/issues

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.3

Additional_repositories https://bgreenwell.r-universe.dev

Imports callr, cli, Cubist, dials (>= 1.3.0), digest, doParallel, dplyr, earth, feasts, foreach, forecast, fs, generics, glue, glmnet, gtools, httr, hts, jsonlite, kernlab, lubridate, magrittr, methods, parallel, parsnip, plyr, purrr, recipes, rlang, rsample, rules, snakecase, stringr, tibble, tidyr, tidyselect, timetk, tune, vroom, workflows

Suggests arrow (>= 8.0.0), AzureStor, Boruta (>= 8.0.0), caret, corrr, ellmer (>= 0.4.0), energy, knitr, Microsoft365R, nixtlar, notebookutils, qs2, ranger, reactable, rmarkdown, sparklyr, testthat (>= 3.0.0), tseries, withr, xgboost, vip (>= 0.5.0)

Config/testthat/edition 3

Depends R (>= 4.0), modeltime

VignetteBuilder knitr

NeedsCompilation no

Author Mike Tokic [aut, cre] (ORCID: <https://orcid.org/0000-0002-7630-7055>),
Aadharsh Kannan [aut] (ORCID: <https://orcid.org/0000-0002-6475-8211>)

Maintainer Mike Tokic <mftokic@gmail.com>

Repository CRAN

Date/Publication 2026-08-27 03:30:02 UTC

Contents

ask_agent 2
ensemble_models 3
final_models 5
forecast_time_series 7
get_agent_forecast 11
get_best_agent_run 12
get_eda_data 13
get_forecast_data 14
get_prepped_data 15
get_prepped_models 16
get_run_info 17
get_summarized_models 18
get_trained_models 20
iterate_forecast 21
list_models 22
prep_data 23
prep_models 25
set_agent_info 27
set_project_info 29
set_run_info 31
train_models 32
update_forecast 34

Index **37**

`ask_agent` *Ask Questions About Finn Agent Forecast Results*

Description

This function allows users to ask questions about their Finn AI Agent forecast results and get answers based on the outputs from iterate_forecast() or update_forecast(). It uses an LLM-driven workflow to generate and execute R code to answer questions.

Usage

```
ask_agent(agent_info, question)
```

Arguments

agent_info	Agent info from set_agent_info()
question	A character string containing the question to ask about the forecast

Value

A character string containing the answer to the question

Examples

```
## Not run:
# After running iterate_forecast() or update_forecast()

# Ask about exploratory data analysis
answer <- ask_agent(
  agent_info = agent_info,
  question = "Were there any missing values in the data?"
)

# Ask about forecast accuracy
answer <- ask_agent(
  agent_info = agent_info,
  question = "What is the average weighted MAPE across all time series?"
)

# Ask about models used
answer <- ask_agent(
  agent_info = agent_info,
  question = "Which models were used for the forecast?"
)

# Ask about feature importance
answer <- ask_agent(
  agent_info = agent_info,
  question = "What are the top 5 most important features in the xgboost model?"
)

# Ask about specific time series
answer <- ask_agent(
  agent_info = agent_info,
  question = "What is the forecast for product XYZ for the next 3 months?"
)

## End(Not run)
```

---

ensemble_models *Ensemble Models*

Description

Create ensemble model forecasts

Usage

```
ensemble_models(
  run_info,
  parallel_processing = NULL,
  inner_parallel = FALSE,
  num_cores = NULL,
  seed = 123
)
```

Arguments

run_info	run info using the set_run_info() function
parallel_processing	Default of NULL runs no parallel processing and forecasts each individual time series one after another. 'local_machine' leverages all cores on current machine Finn is running on. 'spark' runs time series in parallel on a spark cluster in Azure Databricks or Azure Synapse.
inner_parallel	Run components of forecast process inside a specific time series in parallel. Can only be used if parallel_processing is set to NULL or 'spark'.
num_cores	Number of cores to run when parallel processing is set up. Used when running parallel computations on local machine or within Azure. Default of NULL uses total amount of cores on machine minus one. Can't be greater than number of cores on machine minus 1.
seed	Set seed for random number generator. Numeric value.

Value

Ensemble model outputs are written to disk

Examples

```
data_tbl <- timetk::m4_monthly %>%
  dplyr::rename(Date = date) %>%
  dplyr::mutate(id = as.character(id)) %>%
  dplyr::filter(
    Date >= "2013-01-01",
    Date <= "2015-06-01",
    id == "M750"
  )

run_info <- set_run_info()

prep_data(run_info,
  input_data = data_tbl,
  combo_variables = c("id"),
```

```
  target_variable = "value",
  date_type = "month",
  forecast_horizon = 3
)

prep_models(run_info,
  models_to_run = c("arima", "glmnet"),
  num_hyperparameters = 2
)

train_models(run_info,
  run_global_models = FALSE
)

ensemble_models(run_info)
```

`final_models` *Final Models*

Description

Select Best Models and Prep Final Outputs

Usage

```
final_models(
  run_info,
  average_models = TRUE,
  max_model_average = 3,
  weekly_to_daily = TRUE,
  parallel_processing = NULL,
  inner_parallel = FALSE,
  num_cores = NULL
)
```

Arguments

`run_info`	run info using the `set_run_info()` function.
`average_models`	If TRUE, create simple averages of individual models and save the most accurate one.
`max_model_average`	Max number of models to average together. Will create model averages for 2 models up until input value or max number of models ran.
`weekly_to_daily`	If TRUE, convert a week forecast down to day by evenly splitting across each day of week. Helps when aggregating up to higher temporal levels like month or quarter.

parallel_processing
: Default of NULL runs no parallel processing and forecasts each individual time series one after another. 'local_machine' leverages all cores on current machine Finn is running on. 'spark' runs time series in parallel on a spark cluster in Azure Databricks or Azure Synapse.

inner_parallel
: Run components of forecast process inside a specific time series in parallel. Can only be used if parallel_processing is set to NULL or 'spark'.

num_cores
: Number of cores to run when parallel processing is set up. Used when running parallel computations on local machine or within Azure. Default of NULL uses total amount of cores on machine minus one. Can't be greater than number of cores on machine minus 1.

Value

Final model outputs are written to disk.

Examples

```
data_tbl <- timetk::m4_monthly %>%
  dplyr::rename(Date = date) %>%
  dplyr::mutate(id = as.character(id)) %>%
  dplyr::filter(
    Date >= "2013-01-01",
    Date <= "2015-06-01"
  )

run_info <- set_run_info()

prep_data(run_info,
  input_data = data_tbl,
  combo_variables = c("id"),
  target_variable = "value",
  date_type = "month",
  forecast_horizon = 3
)

prep_models(run_info,
  models_to_run = c("arima", "ets"),
  back_test_scenarios = 3
)

train_models(run_info,
  run_global_models = FALSE
)

final_models(run_info)
```

`forecast_time_series` *Finn Forecast Framework*

Description

Calls the Finn forecast framework to automatically forecast any historical time series.

Usage

```
forecast_time_series(
  run_info = NULL,
  input_data,
  combo_variables,
  target_variable,
  date_type,
  forecast_horizon,
  external_regressors = NULL,
  hist_start_date = NULL,
  hist_end_date = NULL,
  combo_cleanup_date = NULL,
  fiscal_year_start = 1,
  clean_missing_values = TRUE,
  clean_outliers = FALSE,
  back_test_scenarios = NULL,
  back_test_spacing = NULL,
  modeling_approach = "accuracy",
  forecast_approach = "bottoms_up",
  parallel_processing = NULL,
  inner_parallel = FALSE,
  num_cores = NULL,
  negative_forecast = FALSE,
  fourier_periods = NULL,
  lag_periods = NULL,
  rolling_window_periods = NULL,
  recipes_to_run = NULL,
  pca = NULL,
  models_to_run = NULL,
  models_not_to_run = NULL,
  run_global_models = NULL,
  run_local_models = TRUE,
  run_ensemble_models = NULL,
  average_models = TRUE,
  max_model_average = 3,
  feature_selection = FALSE,
  weekly_to_daily = TRUE,
  seed = 123,
  run_model_parallel = FALSE,
```

```
  return_data = TRUE,
  run_name = "finnts_forecast"
)
```

Arguments

run_info	Run info using set_run_info()
input_data	A data frame or tibble of historical time series data. Can also include external regressors for both historical and future data.
combo_variables	List of column headers within input data to be used to separate individual time series.
target_variable	The column header formatted as a character value within input data you want to forecast.
date_type	The date granularity of the input data. Finn accepts the following as a character string day, week, month, quarter, year.
forecast_horizon	Number of periods to forecast into the future.
external_regressors	List of column headers within input data to be used as features in multivariate models.
hist_start_date	Date value of when your input_data starts. Default of NULL is to use earliest date value in input_data.
hist_end_date	Date value of when your input_data ends.Default of NULL is to use the latest date value in input_data.
combo_cleanup_date	Date value to remove individual time series that don't contain non-zero values after that specified date. Default of NULL is to not remove any time series and attempt to forecast all of them.
fiscal_year_start	Month number of start of fiscal year of input data, aids in building out date features. Formatted as a numeric value. Default of 1 assumes fiscal year starts in January.
clean_missing_values	If TRUE, cleans missing values. Only impute values for missing data within an existing series, and does not add new values onto the beginning or end, but does provide a value of 0 for said values. Turned off when running hierarchical forecasts.
clean_outliers	If TRUE, outliers are cleaned and inputted with values more in line with historical data
back_test_scenarios	Number of specific back test folds to run when determining the best model. Default of NULL will automatically choose the number of back tests to run based on historical data size, which tries to always use a minimum of 80% of the data when training a model.

back_test_spacing
: Number of periods to move back for each back test scenario. Default of NULL moves back 1 period at a time for year, quarter, and month data. Moves back 4 for week and 7 for day data.

modeling_approach
: How Finn should approach your data. Current default and only option is 'accuracy'. In the future this could evolve to other areas like optimizing for interpretability over accuracy.

forecast_approach
: How the forecast is created. The default of 'bottoms_up' trains models for each individual time series. 'grouped_hierarchy' creates a grouped time series to forecast at while 'standard_hierarchy' creates a more traditional hierarchical time series to forecast, both based on the hts package.

parallel_processing
: Default of NULL runs no parallel processing and forecasts each individual time series one after another. 'local_machine' leverages all cores on current machine Finn is running on. 'spark' runs time series in parallel on a spark cluster in Azure Databricks or Azure Synapse.

inner_parallel
: Run components of forecast process inside a specific time series in parallel. Can only be used if parallel_processing is set to NULL or 'spark'.

num_cores
: Number of cores to run when parallel processing is set up. Used when running parallel computations on local machine or within Azure. Default of NULL uses total amount of cores on machine minus one. Can't be greater than number of cores on machine minus 1.

negative_forecast
: If TRUE, allow forecasts to dip below zero.

fourier_periods
: List of values to use in creating fourier series as features. Default of NULL automatically chooses these values based on the date_type.

lag_periods
: List of values to use in creating lag features. Default of NULL automatically chooses these values based on date_type.

rolling_window_periods
: List of values to use in creating rolling window features. Default of NULL automatically chooses these values based on date type.

recipes_to_run
: List of recipes to run on multivariate models that can run different recipes. A value of NULL runs all recipes, but only runs the R1 recipe for weekly and daily date types, and also for global models to prevent memory issues. A value of "all" runs all recipes, regardless of date type or if it's a local/global model. A list like c("R1") or c("R2") would only run models with the R1 or R2 recipe.

pca
: If TRUE, run principle component analysis on any lagged features to speed up model run time. Default of NULL runs PCA on day and week date types across all local multivariate models, and also for global models across all date types.

models_to_run
: List of models to run. Default of NULL runs all models.

models_not_to_run
: List of models not to run, overrides values in models_to_run. Default of NULL doesn't turn off any model.

run_global_models
: If TRUE, run multivariate models on the entire data set (across all time series) as a global model. Can be override by models_not_to_run. Default of NULL runs global models for all date types except week and day.

run_local_models
: If TRUE, run models by individual time series as local models.

run_ensemble_models
: If TRUE, run ensemble models. Default of NULL runs ensemble models only for quarter and month date types.

average_models
: If TRUE, create simple averages of individual models.

max_model_average
: Max number of models to average together. Will create model averages for 2 models up until input value or max number of models ran.

feature_selection
: Implement feature selection before model training. This requires the optional `Boruta`, `corrr`, `ranger`, and `vip` packages. `vip` is available from https://bgreenwell.r-universe.dev and requires R 4.1 or newer.

weekly_to_daily
: If TRUE, convert a week forecast down to day by evenly splitting across each day of week. Helps when aggregating up to higher temporal levels like month or quarter.

seed
: Set seed for random number generator. Numeric value.

run_model_parallel
: If TRUE, runs model training in parallel, only works when parallel_processing is set to 'local_machine' or 'spark'. Recommended to use a value of FALSE and leverage inner_parallel for new features.

return_data
: If TRUE, return the forecast results. Used to be backwards compatible with previous finnts versions. Recommended to use a value of FALSE and leverage `get_forecast_data()` for new features.

run_name
: Name used when submitting jobs to external compute like Azure Batch. Formatted as a character string.

Value

A list of three separate data sets: the future forecast, the back test results, and the best model per time series.

Examples

```
run_info <- set_run_info()

finn_forecast <- forecast_time_series(
  run_info = run_info,
  input_data = m750 %>% dplyr::rename(Date = date),
  combo_variables = c("id"),
  target_variable = "value",
```

```
  date_type = "month",
  forecast_horizon = 3,
  back_test_scenarios = 6,
  run_model_parallel = FALSE,
  models_to_run = c("arima", "ets", "snaive"),
  return_data = FALSE
)

fcst_tbl <- get_forecast_data(run_info)

models_tbl <- get_trained_models(run_info)
```

`get_agent_forecast` *Get the final best forecast for an agent*

Description

This function retrieves the final forecast for a Finn agent after the forecast iteration process is complete.

Usage

```
get_agent_forecast(agent_info)
```

Arguments

`agent_info`	Agent info from `set_agent_info()`

Value

A tibble containing the final forecast for the agent.

Examples

```
## Not run:
# load example data
hist_data <- timetk::m4_monthly %>%
  dplyr::filter(date >= "2013-01-01") %>%
  dplyr::rename(Date = date) %>%
  dplyr::mutate(id = as.character(id))

# set up Finn project
project <- set_project_info(
  project_name = "Demo_Project",
  combo_variables = c("id"),
  target_variable = "value",
  date_type = "month"
)
```

```
# set up LLM
llm <- ellmer::chat_azure_openai(model = "gpt-4o-mini")

# set up agent info
agent_info <- set_agent_info(
  project_info = project,
  llm = llm,
  input_data = hist_data,
  forecast_horizon = 6
)

# run the forecast iteration process
iterate_forecast(
  agent_info = agent_info,
  max_iter = 3,
  weighted_mape_goal = 0.03
)

# get the final forecast for the agent
final_forecast <- get_agent_forecast(agent_info = agent_info)

## End(Not run)
```

`get_best_agent_run` *Get the best run for an agent*

Description

This function retrieves the best run information for a Finn agent after the forecast iteration process is complete.

Usage

```
get_best_agent_run(agent_info)
```

Arguments

`agent_info`	Agent info from `set_agent_info()`

Value

A tibble containing the best run information for the agent.

Examples

```
## Not run:
# load example data
hist_data <- timetk::m4_monthly %>%
  dplyr::filter(date >= "2013-01-01") %>%
  dplyr::rename(Date = date) %>%
```

```
  dplyr::mutate(id = as.character(id))

# set up Finn project
project <- set_project_info(
  project_name = "Demo_Project",
  combo_variables = c("id"),
  target_variable = "value",
  date_type = "month"
)

# set up LLM
llm <- ellmer::chat_azure_openai(model = "gpt-4o-mini")

# set up agent info
agent_info <- set_agent_info(
  project_info = project,
  llm = llm,
  input_data = hist_data,
  forecast_horizon = 6
)

# run the forecast iteration process
iterate_forecast(
  agent_info = agent_info,
  max_iter = 3,
  weighted_mape_goal = 0.03
)

# get the best run information for the agent
best_run_info <- get_best_agent_run(agent_info = agent_info)

## End(Not run)
```

`get_eda_data` *Get EDA Data*

Description

Load exploratory data analysis results from a Finn Agent run and return as a single data frame

Usage

```
get_eda_data(agent_info)
```

Arguments

`agent_info`	Agent info from `set_agent_info()`

Value

A data frame containing all EDA results with columns:

- Combo: Time series identifier
- Analysis_Type: Type of EDA analysis (e.g., "ACF", "PACF", "Stationarity", etc.)
- Metric: Specific metric or measure within each analysis type
- Value: Numeric or character value of the metric

Examples

```
## Not run:
# Get EDA results for all time series
eda_df <- get_eda_data(agent_info)

# Filter for specific analysis types
acf_results <- eda_df %>%
  dplyr::filter(Analysis_Type == "ACF")

# Filter for specific time series
ts_results <- eda_df %>%
  dplyr::filter(Combo == "Product_A--Region_1")

## End(Not run)
```

---

get_forecast_data *Get Final Forecast Data*

Description

Get Final Forecast Data

Usage

```
get_forecast_data(run_info, return_type = "df")
```

Arguments

run_info	run info using the set_run_info() function
return_type	return type

Value

table of final forecast results

Examples

```
data_tbl <- timetk::m4_monthly %>%
  dplyr::rename(Date = date) %>%
  dplyr::mutate(id = as.character(id)) %>%
  dplyr::filter(
    id == "M2",
    Date >= "2012-01-01",
    Date <= "2015-06-01"
  )

run_info <- set_run_info()

prep_data(run_info,
  input_data = data_tbl,
  combo_variables = c("id"),
  target_variable = "value",
  date_type = "month",
  forecast_horizon = 3,
  recipes_to_run = "R1"
)

prep_models(run_info,
  models_to_run = c("arima", "ets"),
  num_hyperparameters = 1
)

train_models(run_info,
  run_local_models = TRUE
)

final_models(run_info,
  average_models = FALSE
)

fcst_tbl <- get_forecast_data(run_info)
```

---

get_prepped_data *Get Prepped Data*

Description

Get Prepped Data

Usage

```
get_prepped_data(run_info, recipe, return_type = "df")
```

Arguments

run_info	run info using the set_run_info() function
recipe	recipe to return. Either a value of "R1" or "R2"
return_type	return type

Value

table of prepped data

Examples

```
data_tbl <- timetk::m4_monthly %>%
  dplyr::rename(Date = date) %>%
  dplyr::mutate(id = as.character(id)) %>%
  dplyr::filter(
    id == "M2",
    Date >= "2012-01-01",
    Date <= "2015-06-01"
  )

run_info <- set_run_info()

prep_data(run_info,
  input_data = data_tbl,
  combo_variables = c("id"),
  target_variable = "value",
  date_type = "month",
  forecast_horizon = 3,
  recipes_to_run = "R1"
)

R1_prepped_data_tbl <- get_prepped_data(run_info,
  recipe = "R1"
)
```

---

get_prepped_models *Get Prepped Model Info*

Description

Get Prepped Model Info

Usage

```
get_prepped_models(run_info)
```

Arguments

run_info	run info using the set_run_info() function

Value

table with data related to model workflows, hyperparameters, and back testing

Examples

```
data_tbl <- timetk::m4_monthly %>%
  dplyr::rename(Date = date) %>%
  dplyr::mutate(id = as.character(id)) %>%
  dplyr::filter(
    id == "M2",
    Date >= "2012-01-01",
    Date <= "2015-06-01"
  )

run_info <- set_run_info()

prep_data(run_info,
  input_data = data_tbl,
  combo_variables = c("id"),
  target_variable = "value",
  date_type = "month",
  forecast_horizon = 3,
  recipes_to_run = "R1"
)

prep_models(run_info,
  models_to_run = c("arima", "ets"),
  num_hyperparameters = 1
)

prepped_models_tbl <- get_prepped_models(run_info = run_info)
```

---

get_run_info *Get run info*

Description

Lets you get all of the logging associated with a specific project or run.

Usage

```
get_run_info(
  project_name = NULL,
  run_name = NULL,
```

```
  storage_object = NULL,
  path = NULL
)
```

Arguments

project_name	Name used to group similar runs under a single project name.
run_name	Name to distinguish one run of Finn from another. The current time in UTC is appended to the run name to ensure a unique run name is created.
storage_object	Used to store outputs during a run to other storage services in Azure. Could be a storage container object from the 'AzureStor' package to connect to ADLS blob storage or a OneDrive/SharePoint object from the 'Microsoft365R' package to connect to a OneDrive folder or SharePoint site. Default of NULL will save outputs to the local file system.
path	String showing what file path the outputs should be written to. Default of NULL will write the outputs to a temporary directory within R, which will delete itself after the R session closes.

Value

Data frame of run log information

Examples

```
run_info <- set_run_info(
  project_name = "finn_forecast",
  run_name = "test_run"
)

run_info_tbl <- get_run_info(
  project_name = "finn_forecast"
)
```

---

get_summarized_models *Get the trained model summaries info for an agent*

Description

This function retrieves the final summarized model info (hyperparameters, recipe steps, feature importance, etc.) after an agent completes its run. Variable-importance rows are included when the optional `vip` package was available while summaries were generated; all other summary sections remain available without it.

Usage

```
get_summarized_models(agent_info)
```

Arguments

agent_info	Agent info from set_agent_info()

Value

A tibble containing the summarized models for the agent. The importance section is omitted when vip was unavailable during summary generation.

Examples

```
## Not run:
# load example data
hist_data <- timetk::m4_monthly %>%
  dplyr::filter(date >= "2013-01-01") %>%
  dplyr::rename(Date = date) %>%
  dplyr::mutate(id = as.character(id))

# set up Finn project
project <- set_project_info(
  project_name = "Demo_Project",
  combo_variables = c("id"),
  target_variable = "value",
  date_type = "month"
)

# set up LLM
llm <- ellmer::chat_azure_openai(model = "gpt-4o-mini")

# set up agent info
agent_info <- set_agent_info(
  project_info = project,
  llm = llm,
  input_data = hist_data,
  forecast_horizon = 6
)

# run the forecast iteration process
iterate_forecast(
  agent_info = agent_info,
  max_iter = 3,
  weighted_mape_goal = 0.03
)

# get the final model summaries for an agent
model_summary <- get_summarized_models(agent_info = agent_info)

## End(Not run)
```

`get_trained_models` *Get Final Trained Models*

Description

Get Final Trained Models

Usage

```
get_trained_models(run_info)
```

Arguments

`run_info`	run info using the `set_run_info()` function

Value

table of final trained models

Examples

```
data_tbl <- timetk::m4_monthly %>%
  dplyr::rename(Date = date) %>%
  dplyr::mutate(id = as.character(id)) %>%
  dplyr::filter(
    id == "M2",
    Date >= "2012-01-01",
    Date <= "2015-06-01"
  )

run_info <- set_run_info()

prep_data(run_info,
  input_data = data_tbl,
  combo_variables = c("id"),
  target_variable = "value",
  date_type = "month",
  forecast_horizon = 3,
  recipes_to_run = "R1"
)

prep_models(run_info,
  models_to_run = c("arima", "ets"),
  num_hyperparameters = 1
)

train_models(run_info,
  run_global_models = FALSE,
  run_local_models = TRUE
)
```

```
final_models(run_info,
  average_models = FALSE
)

models_tbl <- get_trained_models(run_info)
```

`iterate_forecast` *Run the Finn Agent Forecast Iteration Process*

Description

This function orchestrates the forecast iteration process for a Finn agent, including exploratory data analysis,

Usage

```
iterate_forecast(
  agent_info,
  max_iter = 3,
  weighted_mape_goal = 0.03,
  parallel_processing = NULL,
  inner_parallel = FALSE,
  num_cores = NULL,
  seed = 123
)
```

Arguments

`agent_info`	Agent info from `set_agent_info()`
`max_iter`	Maximum number of iterations for forecast optimization.
`weighted_mape_goal`	Weighted MAPE goal the agent is trying to achieve for each time series
`parallel_processing`	Default of NULL runs no parallel processing and forecasts each individual time series one after another. 'local_machine' leverages all cores on current machine Finn is running on. 'spark' runs time series in parallel on a spark cluster in Azure Databricks or Azure Synapse. Parallel agent workflows require ellmer 0.4.0 or later on the main process and every worker.
`inner_parallel`	Run components of forecast process inside a specific time series in parallel. Can only be used if parallel_processing is set to NULL or 'spark'.
`num_cores`	Number of cores to run when parallel processing is set up. Used when running parallel computations on local machine or within Azure. Default of NULL uses total amount of cores on machine minus one. Can't be greater than number of cores on machine minus 1.
`seed`	Set seed for random number generator. Numeric value.

Examples

```
## Not run:
# load example data
hist_data <- timetk::m4_monthly %>%
  dplyr::filter(date >= "2013-01-01") %>%
  dplyr::rename(Date = date) %>%
  dplyr::mutate(id = as.character(id))

# set up Finn project
project <- set_project_info(
  project_name = "Demo_Project",
  combo_variables = c("id"),
  target_variable = "value",
  date_type = "month"
)

# set up LLM
llm <- ellmer::chat_azure_openai(model = "gpt-4o-mini")

# set up agent info
agent_info <- set_agent_info(
  project_info = project,
  llm = llm,
  input_data = hist_data,
  forecast_horizon = 6
)

# run the forecast iteration process
iterate_forecast(
  agent_info = agent_info,
  max_iter = 3,
  weighted_mape_goal = 0.03
)

## End(Not run)
```

---

list_models *List all available models*

Description

List all available models

Usage

```
list_models()
```

Value

list of models

prep_data *Prep Data*

Description

Preps data with various feature engineering recipes to create features before training models

Usage

```
prep_data(
  run_info,
  input_data,
  combo_variables,
  target_variable,
  date_type,
  forecast_horizon,
  external_regressors = NULL,
  hist_start_date = NULL,
  hist_end_date = NULL,
  combo_cleanup_date = NULL,
  fiscal_year_start = 1,
  clean_missing_values = TRUE,
  clean_outliers = FALSE,
  box_cox = FALSE,
  stationary = TRUE,
  forecast_approach = "bottoms_up",
  parallel_processing = NULL,
  num_cores = NULL,
  fourier_periods = NULL,
  lag_periods = NULL,
  rolling_window_periods = NULL,
  recipes_to_run = NULL,
  multistep_horizon = FALSE
)
```

Arguments

run_info	Run info using set_run_info()
input_data	A standard data frame, tibble, or spark data frame using sparklyr of historical time series data. Can also include external regressors for both historical and future data.
combo_variables	List of column headers within input data to be used to separate individual time series.
target_variable	The column header formatted as a character value within input data you want to forecast.

`date_type` The date granularity of the input data. Finn accepts the following as a character string: day, week, month, quarter, year.

`forecast_horizon`
Number of periods to forecast into the future.

`external_regressors`
List of column headers within input data to be used as features in multivariate models.

`hist_start_date`
Date value of when your input_data starts. Default of NULL uses earliest date value in input_data.

`hist_end_date` Date value of when your input_data ends. Default of NULL uses the latest date value in input_data.

`combo_cleanup_date`
Date value to remove individual time series that don't contain non-zero values after that specified date. Default of NULL is to not remove any time series and attempt to forecast all time series.

`fiscal_year_start`
Month number of start of fiscal year of input data, aids in building out date features. Formatted as a numeric value. Default of 1 assumes fiscal year starts in January.

`clean_missing_values`
If TRUE, cleans missing values. Only impute values for missing data within an existing series, and does not add new values onto the beginning or end, but does provide a value of 0 for said values.

`clean_outliers` If TRUE, outliers are cleaned and inputted with values more in line with historical data.

`box_cox` Apply box-cox transformation to normalize variance in data

`stationary` Apply differencing to make data stationary

`forecast_approach`
How the forecast is created. The default of 'bottoms_up' trains models for each individual time series. Value of 'grouped_hierarchy' creates a grouped time series to forecast at while 'standard_hierarchy' creates a more traditional hierarchical time series to forecast, both based on the hts package.

`parallel_processing`
Default of NULL runs no parallel processing and forecasts each individual time series one after another. Value of 'local_machine' leverages all cores on current machine Finn is running on. Value of 'spark' runs time series in parallel on a spark cluster in Azure Databricks/Synapse.

`num_cores` Number of cores to run when parallel processing is set up. Used when running parallel computations on local machine or within Azure. Default of NULL uses total amount of cores on machine minus one. Can't be greater than number of cores on machine minus 1.

`fourier_periods`
List of values to use in creating fourier series as features. Default of NULL automatically chooses these values based on the date_type.

lag_periods List of values to use in creating lag features. Default of NULL automatically chooses these values based on date_type.

rolling_window_periods
List of values to use in creating rolling window features. Default of NULL automatically chooses these values based on date_type.

recipes_to_run List of recipes to run on multivariate models that can run different recipes. A value of NULL runs all recipes, but only runs the R1 recipe for weekly and daily date types. A value of "all" runs all recipes, regardless of date type. A list like c("R1") or c("R2") would only run models with the R1 or R2 recipe.

multistep_horizon
Use a multistep horizon approach when training multivariate models with R1 recipe.

Value

No return object. Feature engineered data is written to disk based on the output locations provided in set_run_info().

Examples

```
data_tbl <- timetk::m4_monthly %>%
  dplyr::rename(Date = date) %>%
  dplyr::mutate(id = as.character(id)) %>%
  dplyr::filter(
    Date >= "2013-01-01",
    Date <= "2015-06-01"
  )

run_info <- set_run_info()

prep_data(run_info,
  input_data = data_tbl,
  combo_variables = c("id"),
  target_variable = "value",
  date_type = "month",
  forecast_horizon = 3,
  recipes_to_run = "R1"
)
```

---

prep_models *Prep Models*

Description

Preps various aspects of run before training models. Things like train/test splits, creating hyperparameters, etc.

Usage

```
prep_models(
  run_info,
  back_test_scenarios = NULL,
  back_test_spacing = NULL,
  models_to_run = NULL,
  models_not_to_run = NULL,
  run_ensemble_models = TRUE,
  pca = NULL,
  num_hyperparameters = 10,
  seasonal_period = NULL,
  seed = 123
)
```

Arguments

run_info	Run info using the set_run_info() function.
back_test_scenarios	Number of specific back test folds to run when determining the best model. Default of NULL will automatically choose the number of back tests to run based on historical data size, which tries to always use a minimum of 80% of the data when training a model.
back_test_spacing	Number of periods to move back for each back test scenario. Default of NULL moves back 1 period at a time for year, quarter, and month data. Moves back 4 for week and 7 for day data.
models_to_run	List of models to run. Default of NULL runs all models.
models_not_to_run	List of models not to run, overrides values in models_to_run. Default of NULL doesn't turn off any model.
run_ensemble_models	If TRUE, prep for ensemble models.
pca	If TRUE, run principle component analysis on any lagged features to speed up model run time. Default of NULL runs PCA on day and week date types across all local multivariate models, and also for global models across all date types.
num_hyperparameters	Number of hyperparameter combinations to test out on validation data for model tuning.
seasonal_period	List of numbers to be used for seasonal periods in specific univariate models like tbats.
seed	Set seed for random number generator. Numeric value.

Value

Writes outputs related to model prep to disk.

Examples

```
data_tbl <- timetk::m4_monthly %>%
  dplyr::rename(Date = date) %>%
  dplyr::mutate(id = as.character(id)) %>%
  dplyr::filter(
    Date >= "2012-01-01",
    Date <= "2015-06-01"
  )

run_info <- set_run_info()

prep_data(run_info,
  input_data = data_tbl,
  combo_variables = c("id"),
  target_variable = "value",
  date_type = "month",
  forecast_horizon = 3
)

prep_models(run_info,
  models_to_run = c("arima", "ets", "glmnet")
)
```

---

set_agent_info *Set up Finn Agent Run Information*

Description

This function sets up the necessary information for a Finn Agent run, including input data, forecast horizon, and other parameters. It checks for existing runs and allows for overwriting if specified.

Usage

```
set_agent_info(
  project_info,
  llm,
  input_data,
  forecast_horizon,
  external_regressors = NULL,
  hist_end_date = NULL,
  hist_start_date = NULL,
  back_test_scenarios = NULL,
  back_test_spacing = NULL,
  combo_cleanup_date = NULL,
  allow_hierarchical_forecast = FALSE,
  negative_forecast = FALSE,
  run_global_models = NULL,
```

```
  run_local_models = TRUE,
  overwrite = FALSE
)
```

Arguments

project_info	A Finn project from set_project_info()
llm	A Chat LLM object used as the template for isolated agent sessions
input_data	A data frame or tibble containing the input data
forecast_horizon	The number of periods to forecast
external_regressors	Optional character vector of external regressors
hist_end_date	Optional Date object indicating the end of the historical data
hist_start_date	Optional Date object indicating the start of the historical data
back_test_scenarios	Optional character vector of back test scenarios
back_test_spacing	Optional numeric value for back test spacing
combo_cleanup_date	Optional Date object for combo cleanup
allow_hierarchical_forecast	Logical indicating whether to allow hierarchical forecasting
negative_forecast	If TRUE, allow forecasts to dip below zero.
run_global_models	If TRUE, run multivariate models on the entire data set (across all time series) as a global model. Default of NULL runs global models for all date types except week and day.
run_local_models	If TRUE, run models by individual time series as local models. Default is TRUE.
overwrite	Logical indicating whether to overwrite existing agent run info

Value

A list containing the agent run information

Examples

```
## Not run:
# load example data
hist_data <- timetk::m4_monthly %>%
  dplyr::filter(date >= "2013-01-01") %>%
  dplyr::rename(Date = date) %>%
  dplyr::mutate(id = as.character(id))
```

```
# set up Finn project
project <- set_project_info(
  project_name = "Demo_Project",
  combo_variables = c("id"),
  target_variable = "value",
  date_type = "month"
)

# set up LLM
llm <- ellmer::chat_azure_openai(model = "gpt-4o-mini")

# set up agent info
agent_info <- set_agent_info(
  project_info = project,
  llm = llm,
  input_data = hist_data,
  forecast_horizon = 6
)

## End(Not run)
```

`set_project_info` *Set up new finnts forecast project*

Description

Creates list object of information helpful in logging information about your entire forecast project.

Usage

```
set_project_info(
  project_name = "finn_project",
  path = NULL,
  combo_variables,
  target_variable,
  date_type,
  fiscal_year_start = 1,
  weekly_to_daily = TRUE,
  storage_object = NULL,
  data_output = "csv",
  object_output = "rds",
  overwrite = FALSE
)
```

Arguments

`project_name`	Name used to group similar runs under a single project name.

path String showing what file path the outputs should be written to. Default of NULL will write the outputs to a temporary directory within R, which will delete itself after the R session closes.

combo_variables
Character vector of variables to combine into a combo variable.

target_variable
Character string of the target variable to forecast.

date_type Character string of the type of date variable

fiscal_year_start
Numeric value of the month that the fiscal year starts in.

weekly_to_daily
Logical value of whether to convert weekly data to daily data. Default of FALSE will not convert weekly data to daily data.

storage_object Used to store outputs during the project to other storage services in Azure. Could be a storage container object from the 'AzureStor' package to connect to ADLS blob storage or a OneDrive/SharePoint object from the 'Microsoft365R' package to connect to a OneDrive folder or SharePoint site. Default of NULL will save outputs to the local file system.

data_output String value describing the file type for data outputs. Default will write data frame outputs as csv files. The other option of 'parquet' will instead write parquet files.

object_output String value describing the file type for object outputs. Default will write object outputs like trained models as rds files. The other option of 'qs2' will instead serialize R objects as qs2 files by using the 'qs2' package.

overwrite Logical value of whether to overwrite existing project

Value

A list of project information

Examples

```
## Not run:
project_info <- set_project_info(
  project_name = "test_project",
  combo_variables = c("Store", "Product"),
  target_variable = "Sales",
  date_type = "month"
)

## End(Not run)
```

set_run_info *Set up finnts submission*

Description

Creates list object of information helpful in logging information about your run.

Usage

```
set_run_info(
  project_name = "finn_project",
  run_name = "finn_fcst",
  storage_object = NULL,
  path = NULL,
  data_output = "csv",
  object_output = "rds",
  add_unique_id = TRUE
)
```

Arguments

project_name	Name used to group similar runs under a single project name.
run_name	Name to distinguish one run of Finn from another.
storage_object	Used to store outputs during a run to other storage services in Azure. Could be a storage container object from the 'AzureStor' package to connect to ADLS blob storage or a OneDrive/SharePoint object from the 'Microsoft365R' package to connect to a OneDrive folder or SharePoint site. Default of NULL will save outputs to the local file system.
path	String showing what file path the outputs should be written to. Default of NULL will write the outputs to a temporary directory within R, which will delete itself after the R session closes.
data_output	String value describing the file type for data outputs. Default will write data frame outputs as csv files. The other option of 'parquet' will instead write parquet files.
object_output	String value describing the file type for object outputs. Default will write object outputs like trained models as rds files. The other option of 'qs2' will instead serialize R objects as qs2 files by using the 'qs2' package.
add_unique_id	Add a unique id to end of run_name based on submission time. Set to FALSE to supply your own unique run name, which is helpful in multistage ML pipelines.

Value

A list of run information

Examples

```
run_info <- set_run_info(
  project_name = "test_exp",
  run_name = "test_run_1"
)
```

`train_models` *Train Individual Models*

Description

Train Individual Models

Usage

```
train_models(
  run_info,
  run_global_models = FALSE,
  run_local_models = TRUE,
  global_model_recipes = c("R1"),
  feature_selection = FALSE,
  negative_forecast = FALSE,
  parallel_processing = NULL,
  inner_parallel = FALSE,
  num_cores = NULL,
  seed = 123,
  debug = FALSE
)
```

Arguments

`run_info`	run info using the `set_run_info()` function
`run_global_models`	If TRUE, run multivariate models on the entire data set (across all time series) as a global model. Can be override by models_not_to_run. Default of NULL runs global models for all date types except week and day.
`run_local_models`	If TRUE, run models by individual time series as local models.
`global_model_recipes`	Recipes to use in global models.
`feature_selection`	Implement feature selection before model training. This requires the optional `Boruta`, `corrr`, `ranger`, and `vip` packages. `vip` is available from https://bgreenwell.r-universe.dev and requires R 4.1 or newer.

negative_forecast
: If TRUE, allow forecasts to dip below zero.

parallel_processing
: Default of NULL runs no parallel processing and forecasts each individual time series one after another. 'local_machine' leverages all cores on current machine Finn is running on. 'spark' runs time series in parallel on a spark cluster in Azure Databricks or Azure Synapse.

inner_parallel
: Run components of forecast process inside a specific time series in parallel. Can only be used if parallel_processing is set to NULL or 'spark'.

num_cores
: Number of cores to run when parallel processing is set up. Used when running parallel computations on local machine or within Azure. Default of NULL uses total amount of cores on machine minus one. Can't be greater than number of cores on machine minus 1.

seed
: Set seed for random number generator. Numeric value.

debug
: If TRUE, will stop on errors and show traceback.

Value

trained model outputs are written to disk.

Examples

```
data_tbl <- timetk::m4_monthly %>%
  dplyr::rename(Date = date) %>%
  dplyr::mutate(id = as.character(id)) %>%
  dplyr::filter(
    Date >= "2013-01-01",
    Date <= "2015-06-01"
  )

run_info <- set_run_info()

prep_data(run_info,
  input_data = data_tbl,
  combo_variables = c("id"),
  target_variable = "value",
  date_type = "month",
  forecast_horizon = 3
)

prep_models(run_info,
  models_to_run = c("arima", "glmnet"),
  num_hyperparameters = 2,
  back_test_scenarios = 6,
  run_ensemble_models = FALSE
)

train_models(run_info)
```

`update_forecast` *Update Forecast with Latest Data and Inputs*

Description

This function updates the forecast agent with the latest data and inputs. If new time series are detected in the data (up to 20\ with a floor of 10), simple forecasts are automatically created for them using default local model inputs without LLM involvement. If the number of new series exceeds the cap, an error directs the user to use `iterate_forecast()` instead.

Usage

```
update_forecast(
  agent_info,
  weighted_mape_goal = 0.1,
  allow_iterate_forecast = FALSE,
  max_iter = 3,
  parallel_processing = NULL,
  inner_parallel = FALSE,
  num_cores = NULL,
  seed = 123
)
```

Arguments

`agent_info`	Agent info from `set_agent_info()`
`weighted_mape_goal`	Weighted MAPE goal the agent is trying to achieve for each time series
`allow_iterate_forecast`	Logical indicating if the forecast iteration should be allowed if poor performance is detected, meaning >40% of time series with >20% worse weighted MAPE than previous agent run
`max_iter`	Numeric indicating the maximum number of iterations if iterate_forecast is ran
`parallel_processing`	Default of NULL runs no parallel processing and forecasts each individual time series one after another. 'local_machine' leverages all cores on current machine Finn is running on. 'spark' runs time series in parallel on a spark cluster in Azure Databricks or Azure Synapse.
`inner_parallel`	Run components of forecast process inside a specific time series in parallel. Can only be used if parallel_processing is set to NULL or 'spark'.
`num_cores`	Number of cores to run when parallel processing is set up. Used when running parallel computations on local machine or within Azure. Default of NULL uses total amount of cores on machine minus one. Can't be greater than number of cores on machine minus 1.
`seed`	Set seed for random number generator. Numeric value.

Details

If individual time series fail during the global or local model update process, they are automatically re-forecast using default local model inputs (the same treatment as new time series). If more than 20\ existing series (with a floor of 10) fail to update, an error is raised directing the user to use `iterate_forecast()` instead.

Value

Nothing

Examples

```
## Not run:
# load example data
hist_data <- timetk::m4_monthly %>%
  dplyr::filter(date >= "2013-01-01") %>%
  dplyr::rename(Date = date) %>%
  dplyr::mutate(id = as.character(id))

# set up Finn project
project <- set_project_info(
  project_name = "Demo_Project",
  combo_variables = c("id"),
  target_variable = "value",
  date_type = "month"
)

# set up LLM
llm <- ellmer::chat_azure_openai(model = "gpt-4o-mini")

# set up agent info
agent_info <- set_agent_info(
  project_info = project,
  llm = llm,
  input_data = hist_data,
  forecast_horizon = 6,
  hist_end_date = as.Date("2014-12-01")
)

# run the forecast iteration process
iterate_forecast(
  agent_info = agent_info,
  max_iter = 3,
  weighted_mape_goal = 0.03
)

# update the forecast with latest data and inputs
agent_info <- set_agent_info(
  project_info = project,
  llm = llm,
  input_data = hist_data,
  forecast_horizon = 6,
```

```
  hist_end_date = as.Date("2014-12-01"),
  overwrite = TRUE # required to update the agent for latest data and inputs
)

update_forecast(
  agent_info = agent_info,
  weighted_mape_goal = 0.03
)

## End(Not run)
```

Index

ask_agent, 2

ensemble_models, 3

final_models, 5
forecast_time_series, 7

get_agent_forecast, 11
get_best_agent_run, 12
get_eda_data, 13
get_forecast_data, 14
get_forecast_data(), *10*
get_prepped_data, 15
get_prepped_models, 16
get_run_info, 17
get_summarized_models, 18
get_trained_models, 20

iterate_forecast, 21

list_models, 22

prep_data, 23
prep_models, 25

set_agent_info, 27
set_project_info, 29
set_run_info, 31
set_run_info(), *4, 5*, *8*, *14*, *16, 17*, *20*, *23*, *25, 26*, *32*

train_models, 32

update_forecast, 34