Package 'elcf4R'

May 26, 2026

Title Electricity Load Curves Forecasting at Individual Level

Version 0.4.2

Date 2026-05-26

Depends R (>= 3.5.0)

Imports stats, utils, mgcv, earth, keras3, Rcpp, tensorflow, data.table, wavelets, jsonlite, xml2, DBI, RSQLite

LinkingTo Rcpp

Suggests knitr, reticulate, rmarkdown, testthat (>= 3.0.0)

Author Frederic Bertrand [cre, aut] (ORCID: <https://orcid.org/0000-0002-0837-8281>), Fatima Fahs [aut], Myriam Maumy-Bertrand [aut] (ORCID: <https://orcid.org/0000-0002-4615-1512>)

Maintainer Frederic Bertrand <frederic.bertrand@lecnam.net>

Description Implements forecasting methods for individual electricity load curves, including Kernel Wavelet Functional (KWF), clustered KWF, Generalized Additive Models (GAM), Multivariate Adaptive Regression Splines (MARS), and Long Short-Term Memory (LSTM) models. Provides normalized dataset adapters for iFlex, StoreNet, Smart-Grid Smart-City, Low Carbon London, and REFIT; download and read support for IDEAL and GX; explicit Python backend selection for TensorFlow-based LSTM fits; helpers for daily segmentation and rolling-origin benchmarking; and compact shipped example panels and benchmark-result datasets.

LazyLoad yes

LazyData yes

VignetteBuilder knitr

License GPL-3

Encoding UTF-8

URL https://fbertran.github.io/elcf4R/, https://github.com/fbertran/elcf4R

BugReports https://github.com/fbertran/elcf4R/issues

RoxygenNote 7.3.3

NeedsCompilation yes

Repository CRAN

Date/Publication 2026-05-26 17:40:02 UTC

Contents

`elcf4R-package` *Forecasting Individual Electricity Load Curves*

Description

`elcf4R` provides methods and supporting workflows for day-ahead forecasting of individual electricity load curves. The current package surface includes Kernel Wavelet Functional models, clustered KWF, GAM, MARS and LSTM estimators, an explicit helper to configure the Python backend used by the LSTM path, dataset adapters for iFlex, StoreNet, Smart-Grid Smart-City, Low Carbon London and REFIT, scaffolded download/read support for IDEAL and GX, helpers to build daily segments, and rolling-origin benchmarking utilities.

Author(s)

Maintainer: Frederic Bertrand <frederic.bertrand@lecnam.net> (ORCID)

Authors:

- Fatima Fahs <fatima.fahs@es.fr>
- Myriam Maumy-Bertrand <myriam.maumy@ehesp.fr> (ORCID)

See Also

Useful links:

- https://fbertran.github.io/elcf4R/
- https://github.com/fbertran/elcf4R
- Report bugs at https://github.com/fbertran/elcf4R/issues

`elcf4r_assign_kwf_clusters`
Assign segments to a fitted KWF clustering model

Description

Assign segments to a fitted KWF clustering model

Usage

```
elcf4r_assign_kwf_clusters(object, segments)
```

Arguments

object	An elcf4r_kwf_clusters object.
segments	Matrix or data frame of daily segments.

Value

A character vector of cluster labels.

---

elcf4r_benchmark *Run a rolling-origin benchmark on a normalized panel*

Description

Evaluate the package forecasting methods on a normalized panel using a deterministic rolling-origin design. The runner supports the current temperature-aware gam, mars, kwf, kwf_clustered and lstm wrappers and returns both aggregate scores and, optionally, saved point forecasts.

Usage

```
elcf4r_benchmark(
  panel,
  benchmark_index = NULL,
  methods = NULL,
  entity_ids = NULL,
  cohort_size = NULL,
  train_days = 28L,
  test_days = 5L,
  benchmark_name = NULL,
  dataset = NULL,
  use_temperature = TRUE,
  method_args = NULL,
  include_predictions = TRUE,
  thermosensitivity_panel = NULL,
  benchmark_index_carry_cols = NULL,
  seed = NULL,
  tz = "UTC"
)
```

Arguments

panel	Normalized panel data, typically returned by one of the elcf4r_read_*() adapters.
benchmark_index	Optional day-level index. If NULL, it is created with elcf4r_build_benchmark_index().
methods	Character vector of method names to evaluate. Supported values are "gam", "mars", "kwf", "kwf_clustered" and "lstm". If NULL, the runner uses gam, mars, kwf, kwf_clustered and adds lstm only when its backend is available.

`entity_ids` Optional character vector of entity IDs to benchmark.

`cohort_size` Optional maximum number of eligible entities to keep after sorting by `entity_id`.

`train_days` Number of days in each training window.

`test_days` Number of one-day rolling test origins per entity.

`benchmark_name` Optional benchmark identifier. If `NULL`, one is derived from the dataset label and benchmark design.

`dataset` Optional dataset label overriding `unique(panel$dataset)`.

`use_temperature` Logical; if `TRUE`, methods that support temperature will use it when non-missing temperature information is available for the current window.

`method_args` Optional named list of per-method argument overrides.

`include_predictions` Logical; if `TRUE`, return a long table of saved point forecasts and naive forecasts.

`thermosensitivity_panel` Optional normalized panel used for thermosensitivity classification. Defaults to `panel`.

`benchmark_index_carry_cols` Optional `carry_cols` passed to `elcf4r_build_benchmark_index()` when `benchmark_index` is not supplied.

`seed` Optional integer seed forwarded to methods that support user-supplied seeding, such as LSTM, unless overridden in `method_args`.

`tz` Time zone used to derive dates and within-day positions.

Value

An object of class `elcf4r_benchmark` with elements `results`, `predictions`, `cohort_index`, `spec` and `backend`.

Examples

```
id1 <- subset(
  elcf4r_iflex_example,
  entity_id == unique(elcf4r_iflex_example$entity_id)[1]
)
keep_dates <- sort(unique(id1$date))[1:6]
panel_small <- subset(id1, date %in% keep_dates)

bench <- elcf4r_benchmark(
  panel = panel_small,
  methods = "gam",
  cohort_size = 1,
  train_days = 4,
  test_days = 1,
  include_predictions = TRUE
)
head(bench$results)
```

`elcf4r_build_benchmark_index` *Build a day-level benchmark index from a normalized panel*

Description

Create a compact day-level index from a normalized panel. The returned object contains one row per complete entity-day and can be reused to define deterministic benchmark cohorts without shipping the full panel.

Usage

```
elcf4r_build_benchmark_index(
  data,
  carry_cols = NULL,
  id_col = "entity_id",
  timestamp_col = "timestamp",
  value_col = "y",
  temp_col = "temp",
  resolution_minutes = NULL,
  complete_days_only = TRUE,
  drop_na_value = TRUE,
  tz = "UTC"
)
```

Arguments

`data`	Normalized panel data, typically returned by one of the `elcf4r_read_*()` adapters.
`carry_cols`	Optional character vector of additional day-level columns to propagate into the benchmark index. If `NULL`, all non-core columns are carried.
`id_col`	Name of the entity identifier column.
`timestamp_col`	Name of the timestamp column.
`value_col`	Name of the load column.
`temp_col`	Name of the temperature column.
`resolution_minutes`	Sampling resolution in minutes. If `NULL`, it is inferred from the data.
`complete_days_only`	Passed to `elcf4r_build_daily_segments()`.
`drop_na_value`	Passed to `elcf4r_build_daily_segments()`.
`tz`	Time zone used to derive dates and within-day positions.

Value

A day-level data frame suitable for `elcf4r_benchmark()`.

Examples

```
idx <- elcf4r_build_benchmark_index(
  elcf4r_iflex_example,
  carry_cols = c("dataset", "participation_phase", "price_signal")
)
head(idx)
```

`elcf4r_build_daily_segments` *Build daily load-curve segments from a normalized panel*

Description

Convert a long-format load table into one row per entity-day and one column per within-day time index. This is the matrix representation required by functional load-curve models and rolling benchmark scripts.

Usage

```
elcf4r_build_daily_segments(
  data,
  id_col = "entity_id",
  timestamp_col = "timestamp",
  value_col = "y",
  temp_col = "temp",
  carry_cols = NULL,
  expected_points_per_day = NULL,
  resolution_minutes = NULL,
  complete_days_only = TRUE,
  drop_na_value = TRUE,
  tz = "UTC"
)
```

Arguments

`data`	Data frame containing at least entity id, timestamp and load.
`id_col`	Name of the entity identifier column.
`timestamp_col`	Name of the timestamp column.
`value_col`	Name of the load column.
`temp_col`	Optional name of a temperature column used to derive day summaries.
`carry_cols`	Optional day-level columns to propagate into the returned covariate table. Their first non-missing value within each day is kept.
`expected_points_per_day`	Expected number of samples per day. If `NULL`, it is derived from `resolution_minutes`.

`resolution_minutes`
: Sampling resolution in minutes. If `NULL`, it is inferred from timestamps or from a `resolution_minutes` column. Fractional minute values are allowed.

`complete_days_only`
: If `TRUE`, incomplete or duplicated days are dropped from the output.

`drop_na_value`
: If `TRUE`, days with missing load values are dropped.

`tz`
: Time zone used to derive dates and within-day positions.

Value

A list with components `segments`, `covariates`, `resolution_minutes` and `points_per_day`.

Examples

```
id1 <- subset(
  elcf4r_iflex_example,
  entity_id == unique(elcf4r_iflex_example$entity_id)[1]
)
daily <- elcf4r_build_daily_segments(id1, carry_cols = "participation_phase")
dim(daily$segments)
names(daily$covariates)
```

---

elcf4r_calendar_groups

Derive deterministic KWF calendar groups

Description

Build the deterministic day groups used by the residential KWF workflow: weekdays, `pre_holiday`, and `holiday`.

Usage

```
elcf4r_calendar_groups(dates, holidays = NULL)
```

Arguments

`dates`
: Vector coercible to `Date`.

`holidays`
: Optional vector of holiday dates. If supplied, holiday dates are labelled `"holiday"` and the dates immediately before them are labelled `"pre_holiday"`.

Value

An ordered factor with levels `monday`, `tuesday`, `wednesday`, `thursday`, `friday`, `saturday`, `sunday`, `pre_holiday`, `holiday`.

Examples

```
elcf4r_calendar_groups(
  as.Date(c("2024-12-24", "2024-12-25", "2024-12-26")),
  holidays = as.Date("2024-12-25")
)
```

`elcf4r_classify_thermosensitivity` *Classify thermosensitivity from daily load data*

Description

Estimate thermosensitivity using the residential rule based on the ratio between mean winter load and mean summer load.

Usage

```
elcf4r_classify_thermosensitivity(
  data,
  id_col = "entity_id",
  date_col = "date",
  value_col = "y",
  threshold = 1.5,
  winter_months = c(12L, 1L, 2L),
  summer_months = c(6L, 7L, 8L)
)
```

Arguments

`data`	Data frame containing at least an identifier, a date and a load column. Long-format panels are accepted and are aggregated to mean daily load before classification.
`id_col`	Name of the entity identifier column.
`date_col`	Name of the date column.
`value_col`	Name of the load column.
`threshold`	Ratio threshold above which the series is classified as thermosensitive. Defaults to `1.5`.
`winter_months`	Integer vector of winter months.
`summer_months`	Integer vector of summer months.

Value

A data frame with one row per entity and columns `winter_mean`, `summer_mean`, `ratio`, `thermosensitive`, and `status`.

Examples

```
example_ts <- data.frame(
  entity_id = rep("home_1", 4),
  date = as.Date(c("2024-01-10", "2024-01-11", "2024-07-10", "2024-07-11")),
  y = c(12, 11, 6, 5)
)
elcf4r_classify_thermosensitivity(example_ts)
```

`elcf4r_download_elmas` *Download the ELMAS dataset from figshare*

Description

This function downloads the original ELMAS archive from its public figshare URL and unpacks it to a local directory.

Usage

```
elcf4r_download_elmas(dest_dir)
```

Arguments

`dest_dir`	Directory where the files should be unpacked.

Value

A character vector with the paths of the extracted files.

`elcf4r_download_gx` *Download selected GX dataset components*

Description

Download selected assets from the official GX figshare dataset record. The helper only uses the dataset record itself and does not rely on the authors' code repository.

Usage

```
elcf4r_download_gx(dest_dir, components = "shapefile", overwrite = FALSE)
```

Arguments

`dest_dir`	Directory where the downloaded files should be stored.
`components`	Character vector of GX components to fetch. Supported values are `"shapefile"` and `"database"`.
`overwrite`	Logical; if `TRUE`, existing local files are replaced.

Value

A character vector with the downloaded local file paths. Zip assets are extracted into `dest_dir` and the extracted paths are returned.

---

elcf4r_download_ideal *Download selected IDEAL dataset components*

Description

Download selected assets from the IDEAL Household Energy Dataset record on Edinburgh DataShare. The helper is docs-first: it always retrieves the licence/readme files and `documentation.zip`, while heavy raw-data archives must be requested explicitly through `components`.

Usage

```
elcf4r_download_ideal(
  dest_dir,
  components = "documentation",
  overwrite = FALSE
)
```

Arguments

`dest_dir`	Directory where the downloaded files should be stored.
`components`	Character vector of IDEAL components to fetch. Supported values are `"documentation"`, `"metadata_and_surveys"`, `"coding"`, `"auxiliary"`, `"household_sensors"` and `"room_and_appliance_sensors"`.
`overwrite`	Logical; if `TRUE`, existing local files are replaced.

Value

A character vector with the downloaded local file paths.

---

elcf4r_download_sgsc *Download the Smart-Grid Smart-City interval-reading archive*

Description

Download the Data.gov.au Smart-Grid Smart-City Customer Trial electricity interval-reading resource to a local directory. The full raw archive is not shipped with the package and is never downloaded by examples, tests or vignettes.

Usage

```
elcf4r_download_sgsc(
  destdir = file.path(tempdir(), "elcf4R-sgsc"),
  overwrite = FALSE,
  quiet = FALSE
)
```

Arguments

destdir	Directory where the SGSC archive should be stored. Defaults to a session-specific subdirectory under `tempdir()`.
overwrite	Logical; if `TRUE`, re-download an existing archive.
quiet	Logical; passed to `utils::download.file()`.

Value

Character scalar giving the local archive path.

Examples

```
## Not run:
archive <- elcf4r_download_sgsc()
archive

## End(Not run)
```

---

elcf4r_download_storenet

Download one or more StoreNet household files from figshare

Description

Download one or more StoreNet household files such as `H6_W.csv` into a local directory. The helper uses the figshare article API to resolve the actual file download URL when household-level article IDs are available. Otherwise it falls back to the public StoreNet archive and extracts the requested household files into `dest_dir`.

Usage

```
elcf4r_download_storenet(
  dest_dir,
  ids = "H6_W",
  article_ids = NULL,
  overwrite = FALSE,
  archive_url = "https://figshare.com/ndownloader/files/45123456"
)
```

Arguments

dest_dir	Directory where the downloaded files should be stored.
ids	Character vector of StoreNet household identifiers, for example "H6_W". Use NULL to extract every H*_W.csv file from the archive.
article_ids	Optional named integer vector that maps each requested household identifier to a figshare article ID. When NULL, the built-in mapping is used.
overwrite	Logical; if TRUE, existing local files are replaced.
archive_url	Optional figshare archive download URL used when a requested identifier is not present in the article-ID mapping.

Details

The default mapping currently covers the H6_W household file used by the package examples. Additional households can be downloaded either by providing a named article_ids vector or by relying on the public archive fallback.

Value

A character vector with the downloaded local file paths.

---

elcf4r_elmas_toy *Toy subset of ELMAS hourly cluster profiles*

Description

A compact subset of the public ELMAS dataset containing hourly load profiles for 3 commercial or industrial load clusters over 70 days. The object is intended for lightweight examples and tests that demonstrate time-series or segment-based workflows without shipping the full source archive.

Format

A tibble with 5,040 rows and 3 variables:

time Hourly timestamp.

cluster_id Cluster identifier, one of 3 retained ELMAS clusters.

load_mwh Cluster load in MWh.

Source

Public ELMAS dataset, reduced with package data-raw scripts for examples and tests.

elcf4r_fit_gam *Fit a GAM model for load curves*

Description

Fit a GAM model for load curves

Usage

```
elcf4r_fit_gam(data, use_temperature = FALSE)
```

Arguments

data	Data frame with columns y (load), time_index (numeric or factor for within day position), dow, month, optional temp and other covariates.
use_temperature	Logical. If TRUE, include temperature as smooth effect and interactions.

Value

An object of class elcf4r_model with method = "gam".

Examples

```
id1 <- subset(
  elcf4r_iflex_example,
  entity_id == unique(elcf4r_iflex_example$entity_id)[1]
)
train_data <- subset(id1, date < sort(unique(id1$date))[11])
test_data <- subset(id1, date == sort(unique(id1$date))[11])
fit <- elcf4r_fit_gam(train_data[, c("y", "time_index", "dow", "month", "temp")], TRUE)
pred <- predict(fit, newdata = test_data[, c("y", "time_index", "dow", "month", "temp")])
length(pred)
```

elcf4r_fit_kwf *Fit a Kernel Wavelet Functional model for daily load curves*

Description

Fit a day-ahead Kernel Wavelet Functional (KWF) model on ordered daily load curves. The implementation computes wavelet-detail distances on the historical context days, applies Gaussian kernel weights, restricts those weights to matching calendar groups when available, and can apply the approximation/detail correction used for mean-level non-stationarity.

Usage

```
elcf4r_fit_kwf(
  segments,
  covariates = NULL,
  target_covariates = NULL,
  use_temperature = FALSE,
  wavelet = "la12",
  bandwidth = NULL,
  use_mean_correction = TRUE,
  group_col = NULL,
  holidays = NULL,
  weights = NULL,
  recency_decay = NULL,
  temperature_bandwidth = NULL
)
```

Arguments

segments	Matrix or data frame of past daily load curves (rows are days, columns are within-day time points) in chronological order.
covariates	Optional data frame with one row per training segment. When present, the function looks for deterministic grouping information in context_group, kwf_group, calendar_group, or the column named by group_col. If no explicit group column is present, groups are derived from date and holidays, or from dow as a fallback.
target_covariates	Optional one-row data frame describing the day to forecast. When it contains date, the previous day is used as the context day for calendar grouping, which matches the residential KWF protocol for pre-holiday handling.
use_temperature	Deprecated and ignored. Kept for backward compatibility with earlier package examples.
wavelet	Wavelet filter name passed to wavelets::dwt(). Defaults to "la12", the least-asymmetric filter. If the series is too short for the requested filter, the function falls back to "haar".
bandwidth	Optional positive bandwidth for the Gaussian kernel on wavelet distances. If NULL, it is inferred from the distances to the last observed segment.
use_mean_correction	Logical; if TRUE, apply the approximation/detail correction used for mean-level non-stationarity.
group_col	Optional column name containing precomputed KWF groups in covariates and target_covariates.
holidays	Optional vector of holiday dates used by elcf4r_calendar_groups() when deterministic groups are derived from date.
weights	Optional numeric prior weights of length nrow(segments). Only the first nrow(segments) - 1 values are used in the historical pairing step.

`recency_decay` Optional non-negative recency coefficient applied as an exponential prior on the historical context days.

`temperature_bandwidth` Deprecated and ignored. Kept only for backward compatibility with older examples.

Value

An object of class `elcf4r_model` with `method = "kwf"`.

Examples

```
id1 <- subset(
  elcf4r_iflex_example,
  entity_id == unique(elcf4r_iflex_example$entity_id)[1]
)
daily <- elcf4r_build_daily_segments(id1, carry_cols = "participation_phase")
fit <- elcf4r_fit_kwf(
  segments = daily$segments[1:10, ],
  covariates = daily$covariates[1:10, ],
  target_covariates = daily$covariates[11, , drop = FALSE]
)
length(predict(fit))
```

---

elcf4r_fit_kwf_clustered

Fit a clustered KWF model for daily load curves

Description

Cluster dyadically resampled daily curves in a wavelet-energy feature space and use the resulting cluster labels as the grouping structure inside the KWF forecast.

Usage

```
elcf4r_fit_kwf_clustered(
  segments,
  covariates = NULL,
  target_covariates = NULL,
  wavelet = "la12",
  bandwidth = NULL,
  use_mean_correction = TRUE,
  max_clusters = 10L,
  nstart = 30L,
  cluster_seed = NULL,
  weights = NULL,
  recency_decay = NULL,
  clustering = NULL
)
```

Arguments

segments	Matrix or data frame of past daily load curves in chronological order.
covariates	Optional data frame with one row per segment.
target_covariates	Optional one-row data frame for the target day.
wavelet	Wavelet filter name passed to wavelets::dwt(). Defaults to "la12".
bandwidth	Optional positive bandwidth for the Gaussian kernel in the underlying KWF fit.
use_mean_correction	Logical; if TRUE, apply the approximation/detail correction in the underlying KWF fit.
max_clusters	Maximum number of candidate clusters considered by the Sugar jump heuristic.
nstart	Number of random starts for kmeans.
cluster_seed	Deprecated and ignored. Clustered KWF now uses deterministic non-random starts.
weights	Optional prior weights passed through to the base KWF fit.
recency_decay	Optional recency prior passed through to the base KWF fit.
clustering	Optional elcf4r_kwf_clusters object. When supplied, the stored clustering model is reused instead of being refit.

Value

An object of class elcf4r_model with method = "kwf_clustered".

---

elcf4r_fit_lstm *Fit an LSTM model for daily load curves*

Description

The LSTM implementation uses one or more previous daily curves to predict the next daily curve. When use_temperature = TRUE and temp_mean is available in covariates, the daily mean temperature is added as a second input feature repeated across the within-day time steps.

Usage

```
elcf4r_fit_lstm(
  segments,
  covariates = NULL,
  use_temperature = FALSE,
  lookback_days = 1L,
  units = 16L,
  epochs = 10L,
  batch_size = 8L,
  validation_split = 0,
  seed = NULL,
  verbose = 0L
)
```

Arguments

segments	Matrix or data frame of past daily load curves (rows are days, columns are time points).
covariates	Optional data frame with one row per training day.
use_temperature	Logical. If TRUE, use temp_mean from covariates as an additional input feature when available.
lookback_days	Number of past daily curves used as one training input.
units	Number of hidden units in the LSTM layer.
epochs	Number of training epochs.
batch_size	Batch size used in keras3::fit().
validation_split	Validation split passed to keras3::fit().
seed	Optional integer seed passed to TensorFlow. When NULL, the current backend RNG state is used.
verbose	Verbosity level passed to keras3::fit() and predict().

Value

An object of class elcf4r_model with method = "lstm".

Examples

```
if (interactive() &&
    requireNamespace("reticulate", quietly = TRUE) &&
    reticulate::virtualenv_exists("r-tensorflow")) {
  elcf4r_use_tensorflow_env(virtualenv = "r-tensorflow")
  if (isTRUE(getFromNamespace(".elcf4r_lstm_backend_available", "elcf4R")())) {
    id1 <- subset(
      elcf4r_iflex_example,
      entity_id == unique(elcf4r_iflex_example$entity_id)[1]
    )
    daily <- elcf4r_build_daily_segments(id1)
    fit <- elcf4r_fit_lstm(
      segments = daily$segments[1:10, ],
      covariates = daily$covariates[1:10, ],
      use_temperature = TRUE,
      epochs = 1,
      units = 4,
      batch_size = 2,
      verbose = 0
    )
    length(predict(fit))
  }
}
```

elcf4r_fit_mars *Fit a MARS model for load curves*

Description

Fit a MARS model for load curves

Usage

```
elcf4r_fit_mars(data, use_temperature = FALSE)
```

Arguments

data	Data frame with columns y (load), time_index (numeric or factor for within day position), dow, month, optional temp and other covariates.
use_temperature	Logical. If TRUE, include temperature as smooth effect and interactions.

Value

An elcf4r_model object with method = "mars".

Examples

```
id1 <- subset(
  elcf4r_iflex_example,
  entity_id == unique(elcf4r_iflex_example$entity_id)[1]
)
train_data <- subset(id1, date < sort(unique(id1$date))[11])
test_data <- subset(id1, date == sort(unique(id1$date))[11])
fit <- elcf4r_fit_mars(train_data[, c("y", "time_index", "dow", "month", "temp")], TRUE)
pred <- predict(fit, newdata = test_data[, c("y", "time_index", "dow", "month", "temp")])
length(pred)
```

elcf4r_iflex_benchmark_index *iFlex benchmark index of complete participant-days*

Description

A compact index of complete days derived from the public iFlex hourly panel. Each row represents one participant-day with enough metadata to define deterministic benchmark cohorts without shipping the full raw panel.

Format

A data frame with 563,150 rows and 11 variables:

day_key Unique key built as `entity_id__date`.

entity_id Participant identifier.

date Calendar date.

dow Day of week.

month Month as a two-digit factor.

temp_mean Mean daily outdoor temperature in degrees Celsius.

temp_min Minimum daily outdoor temperature in degrees Celsius.

temp_max Maximum daily outdoor temperature in degrees Celsius.

participation_phase Experiment phase from the source dataset.

price_signal Experimental price-signal label, when available.

n_points Number of hourly samples retained for the day.

Source

Public iFlex raw file `data_hourly.csv`, reduced with `data-raw/elcf4r_iflex_subsets.R`.

`elcf4r_iflex_benchmark_results` *iFlex benchmark results for shipped forecasting methods*

Description

Saved benchmark results for a deterministic rolling-origin evaluation on a subset of the iFlex data. The shipped results use a fixed participant cohort, a 28-day training window and multiple one-day rolling test forecasts per participant. The current shipped benchmark includes the operational `gam`, `mars`, `kwf`, `kwf_clustered` and `lstm` wrappers.

Format

A data frame with 20 variables:

benchmark_name Identifier of the benchmark design.

dataset Dataset label, always `"iflex"`.

entity_id Participant identifier.

method Forecasting method: `gam`, `mars`, `kwf`, `kwf_clustered` or `lstm`.

test_date Date of the forecast target day.

train_start First day in the training window.

train_end Last day in the training window.

train_days Number of training days.

test_points Number of hourly points in the target day.

use_temperature Logical flag for temperature-aware fitting.

thermosensitive Thermosensitivity flag when seasonal coverage is sufficient, otherwise NA.

thermosensitivity_status Status of the winter/summer ratio classification step.

thermosensitivity_ratio Estimated winter/summer mean-load ratio when available.

fit_seconds Elapsed fit-and-predict time in seconds.

status Benchmark execution status.

error_message Error message when a fit failed.

nmae Normalized mean absolute error.

nrmse Normalized root mean squared error.

smape Symmetric mean absolute percentage error.

mase Mean absolute scaled error.

Source

Derived from elcf4r_iflex_benchmark_index and the public iFlex raw file with data-raw/elcf4r_iflex_benchmark_re

---

elcf4r_iflex_example *iFlex example panel for package examples*

Description

A compact hourly electricity-demand panel extracted from the public iFlex dataset. The object contains 14 complete days for each of 3 participants and is intended for examples, tests and lightweight vignettes.

Format

A data frame with 1,008 rows and 16 variables:

dataset Dataset label, always "iflex".

entity_id Participant identifier.

timestamp Hourly UTC timestamp.

date Calendar date of the observation.

time_index Within-day hourly index from 1 to 24.

y Hourly electricity demand in kWh.

temp Outdoor temperature in degrees Celsius.

dow Day of week.

month Month as a two-digit factor.

resolution_minutes Sampling resolution in minutes.

participation_phase Experiment phase from the source dataset.

price_signal Experimental price-signal label, when available.

price_nok_kwh Experimental electricity price in NOK per kWh.

temp24 Lagged 24-hour temperature feature from the source file.

temp48 Lagged 48-hour temperature feature from the source file.

temp72 Lagged 72-hour temperature feature from the source file.

Source

Public iFlex raw file `data_hourly.csv`, reduced with `data-raw/elcf4r_iflex_subsets.R`.

`elcf4r_kwf_cluster_days`

Cluster daily segments for clustered KWF

Description

Build a reusable clustering model for daily load-curve segments in the wavelet-energy feature space used by the clustered KWF workflow.

Usage

```
elcf4r_kwf_cluster_days(
  segments,
  wavelet = "la12",
  max_clusters = 10L,
  nstart = 30L,
  cluster_seed = NULL
)
```

Arguments

`segments`	Matrix or data frame of daily load curves in chronological order.
`wavelet`	Wavelet filter name passed to `wavelets::dwt()`. Defaults to `"la12"`.
`max_clusters`	Maximum number of candidate clusters considered by the Sugar jump heuristic.
`nstart`	Number of random starts for `kmeans`.
`cluster_seed`	Deprecated and ignored. Clustering now uses deterministic non-random starts.

Value

An object of class `elcf4r_kwf_clusters`.

elcf4r_lcl_benchmark_results

Low Carbon London benchmark results for shipped forecasting methods

Description

Saved rolling-origin benchmark results for the shipped methods on a fixed Low Carbon London cohort of households. The benchmark is based on 30-minute load curves and reports NMAE, NRMSE, sMAPE and MASE.

Format

A data frame with the same benchmark-result schema as `elcf4r_iflex_benchmark_results`.

Source

Derived from the local LCL raw file with `data-raw/elcf4r_lcl_artifacts.R`.

elcf4r_lcl_example

Low Carbon London example panel for package examples

Description

A compact normalized panel extracted from a small group of households in the Low Carbon London dataset. The object contains complete 30-minute days and is intended for examples and lightweight benchmarking workflows.

Format

A data frame with normalized panel fields:

dataset Dataset label, always `"lcl"`.

entity_id Low Carbon London household identifier.

timestamp,date,time_index,y,temp,dow,month,resolution_minutes Common normalized panel fields.

Source

Public LCL raw file `LCL_2013.csv`, reduced with `data-raw/elcf4r_lcl_artifacts.R`.

elcf4r_metrics *Forecast accuracy metrics for load curves*

Description

Compute NMAE, NRMSE, sMAPE and MASE between observed and predicted load curves, as in the posters.

Usage

```
elcf4r_metrics(truth, pred, seasonal_period = NULL, naive_pred = NULL)
```

Arguments

`truth`	Numeric vector or matrix of observed values.
`pred`	Numeric vector or matrix of predicted values, same shape.
`seasonal_period`	Seasonal period for the naive benchmark in the MASE denominator (for daily curves with half hourly sampling, a value of 48 is appropriate).
`naive_pred`	Optional numeric vector or matrix of naive benchmark predictions with the same shape as `truth`. When supplied, MASE is computed against this explicit naive forecast instead of inferring the denominator from `seasonal_period` within `truth`.

Value

A named list with elements `nmae`, `nrmse`, `smape`, `mase`.

elcf4r_normalize_panel *Normalize a load panel to the elcf4R schema*

Description

Convert a raw long-format load table into a normalized panel that uses the column names expected by the package examples and model wrappers.

Usage

```
elcf4r_normalize_panel(
  data,
  id_col,
  timestamp_col,
  load_col,
  temp_col = NULL,
```

```
  dataset = NA_character_,
  resolution_minutes = NULL,
  tz = "UTC",
  keep_cols = NULL
)
```

Arguments

data	Data frame containing at least an entity identifier, a time stamp and a load column.
id_col	Name of the entity identifier column.
timestamp_col	Name of the timestamp column.
load_col	Name of the load column.
temp_col	Optional name of the temperature column.
dataset	Short dataset label stored in the normalized output.
resolution_minutes	Sampling resolution in minutes. If NULL, it is inferred from the timestamps. Fractional minute values are allowed for high-frequency data.
tz	Time zone used to parse timestamps.
keep_cols	Optional character vector of extra source columns to keep.

Value

A data frame with normalized columns dataset, entity_id, timestamp, date, time_index, y, temp, dow, month and resolution_minutes, plus any requested keep_cols.

---

elcf4r_read_gx *Read and normalize the GX residential transformer-level scaffold*

Description

Read the GX dataset from either the official SQLite database or a flat export and return a normalized long-format panel. GX is treated as a transformer/community-level dataset rather than an individual-household dataset.

Usage

```
elcf4r_read_gx(
  path = "data-raw",
  ids = NULL,
  start = NULL,
  end = NULL,
  tz = "Asia/Shanghai",
  n_max = NULL,
  drop_na_load = TRUE
)
```

Arguments

path	Path to a GX SQLite database, a flat export file, or a directory containing one of them.
ids	Optional vector of GX community/profile identifiers to keep.
start	Optional inclusive lower time bound.
end	Optional inclusive upper time bound.
tz	Time zone used to parse timestamps. Defaults to "Asia/Shanghai".
n_max	Optional maximum number of rows to read.
drop_na_load	Logical; if TRUE, rows with missing load values are dropped.

Value

A normalized data frame with GX transformer-level data.

---

elcf4r_read_ideal *Read and normalize the IDEAL hourly aggregate-electricity scaffold*

Description

Read a direct IDEAL hourly aggregate-electricity file or search an extracted auxiliarydata.zip directory for a matching hourly summary file, then return a normalized long-format panel.

Usage

```
elcf4r_read_ideal(
  path = "data-raw",
  ids = NULL,
  start = NULL,
  end = NULL,
  tz = "Europe/London",
  n_max = NULL,
  source = "auxiliary_hourly",
  drop_na_load = TRUE
)
```

Arguments

path	Path to an IDEAL hourly summary file or to an extracted IDEAL auxiliary-data directory.
ids	Optional vector of IDEAL household identifiers to keep.
start	Optional inclusive lower time bound.
end	Optional inclusive upper time bound.
tz	Time zone used to parse timestamps. Defaults to "Europe/London".
n_max	Optional maximum number of rows to read.
source	IDEAL source flavor. Currently only "auxiliary_hourly" is supported.
drop_na_load	Logical; if TRUE, rows with missing load values are dropped.

Value

A normalized data frame with IDEAL household data.

`elcf4r_read_iflex` *Read and normalize the iFlex hourly dataset*

Description

Read the iFlex hourly consumption table and return a normalized long-format panel ready for feature engineering, segmentation and benchmarking.

Usage

```
elcf4r_read_iflex(
  path = file.path("data-raw", "iFlex"),
  ids = NULL,
  start = NULL,
  end = NULL,
  tz = "UTC",
  n_max = NULL
)
```

Arguments

`path`	Path to `data_hourly.csv` or to the directory that contains it.
`ids`	Optional vector of participant identifiers to keep.
`start`	Optional inclusive lower time bound.
`end`	Optional inclusive upper time bound.
`tz`	Time zone used to parse timestamps. Defaults to `"UTC"` because the iFlex timestamps are stored with a trailing `Z`.
`n_max`	Optional maximum number of rows to read. Intended for quick prototyping on a small subset of the raw file.

Value

A normalized data frame with load, temperature and calendar fields. The output also keeps `participation_phase`, `price_signal`, `price_nok_kwh`, `temp24`, `temp48` and `temp72`.

elcf4r_read_lcl *Read and normalize the Low Carbon London dataset*

Description

Read a wide Low Carbon London (LCL) smart-meter file and reshape it into a normalized long-format panel with one row per household timestamp.

Usage

```
elcf4r_read_lcl(
  path = file.path("data-raw", "LCL_2013.csv"),
  ids = NULL,
  start = NULL,
  end = NULL,
  tz = "UTC",
  n_max = NULL,
  drop_na_load = TRUE
)
```

Arguments

path	Path to an LCL CSV file or to a directory containing one.
ids	Optional vector of LCL household identifiers to keep, for example "MAC000002".
start	Optional inclusive lower time bound.
end	Optional inclusive upper time bound.
tz	Time zone used to parse timestamps.
n_max	Optional maximum number of timestamp rows to read.
drop_na_load	Logical; if TRUE, rows with missing load values are dropped after reshaping.

Value

A normalized data frame with LCL household data.

elcf4r_read_refit *Read and normalize the REFIT cleaned household dataset*

Description

Read one or more CLEAN_House*.csv files from the REFIT dataset, optionally select appliance channels, resample them to a regular time grid, and return a normalized long-format panel.

Usage

```
elcf4r_read_refit(
  path = "data-raw",
  house_ids = NULL,
  channels = "Aggregate",
  start = NULL,
  end = NULL,
  tz = "UTC",
  resolution_minutes = 1L,
  agg_fun = c("mean", "sum", "last"),
  n_max = NULL,
  drop_na_load = TRUE
)
```

Arguments

path	Path to a REFIT file or to a directory containing CLEAN_House*.csv files.
house_ids	Optional vector of house identifiers to keep. These are matched against file stems such as "CLEAN_House1".
channels	Character vector of load channels to extract. Defaults to "Aggregate".
start	Optional inclusive lower time bound.
end	Optional inclusive upper time bound.
tz	Time zone used to parse timestamps.
resolution_minutes	Target regular resolution in minutes for the normalized output. Defaults to 1.
agg_fun	Aggregation used when resampling to the target grid. One of "mean", "sum" or "last".
n_max	Optional maximum number of raw rows to read per file.
drop_na_load	Logical; if TRUE, rows with missing load values are dropped after resampling.

Value

A normalized data frame with REFIT household data.

---

elcf4r_read_sgsc *Read and normalize the Smart-Grid Smart-City interval-reading dataset*

Description

Read an extracted Smart-Grid Smart-City Customer Trial CSV and return a normalized half-hourly load panel. The reader works on local files only and never downloads external data.

Usage

```
elcf4r_read_sgsc(
  path = file.path("data-raw", "SGSC", "cdintervalreadingallnoquotes.csv"),
  ids = NULL,
  start = NULL,
  end = NULL,
  tz = "Australia/Sydney",
  n_max = NULL,
  id_col = NULL,
  timestamp_col = NULL,
  load_col = NULL,
  generation_col = NULL,
  keep_cols = NULL,
  drop_na_load = TRUE
)
```

Arguments

path	Path to cdintervalreadingallnoquotes.csv or to a directory containing that extracted CSV.
ids	Optional vector of SGSC customer identifiers to keep.
start	Optional inclusive lower time bound.
end	Optional inclusive upper time bound.
tz	Time zone used to parse timestamps. Defaults to "Australia/Sydney".
n_max	Optional maximum number of rows to read.
id_col	Optional explicit SGSC identifier column.
timestamp_col	Optional explicit timestamp column. When NULL, the reader looks for a combined timestamp column first and then falls back to separate date/time columns when available.
load_col	Optional explicit load column. When NULL, the reader prefers general-supply/import consumption aliases.
generation_col	Optional explicit generation column to preserve as metadata when available.
keep_cols	Optional extra source columns to keep in addition to the normalized core schema and detected SGSC metadata fields.
drop_na_load	Logical; if TRUE, rows with missing load values are dropped.

Value

A normalized data frame with SGSC household data.

`elcf4r_read_storenet` *Read and normalize the StoreNet household dataset*

Description

Read one or more StoreNet-style household CSV files such as `H6_W.csv`, derive the household identifier from the file name, and return a normalized long-format panel.

Usage

```
elcf4r_read_storenet(
  path = file.path("data-raw", "H6_W.csv"),
  ids = NULL,
  start = NULL,
  end = NULL,
  tz = "UTC",
  n_max = NULL,
  load_col = "Consumption(W)",
 keep_cols = c("Discharge(W)", "Charge(W)", "Production(W)", "State of Charge(%)")
)
```

Arguments

`path`	Path to a StoreNet CSV file or to a directory containing files named like `H6_W.csv`.
`ids`	Optional vector of household identifiers to keep. Identifiers are matched against the file stem, for example `"H6_W"`.
`start`	Optional inclusive lower time bound.
`end`	Optional inclusive upper time bound.
`tz`	Time zone used to parse timestamps.
`n_max`	Optional maximum number of rows to read per file.
`load_col`	Name of the load column to normalize. Defaults to `"Consumption(W)"`.
`keep_cols`	Optional extra source columns to keep. Defaults to the main battery and production fields when present.

Value

A normalized data frame with StoreNet household data.

elcf4r_refit_benchmark_results *REFIT benchmark results for shipped forecasting methods*

Description

Saved rolling-origin benchmark results for the shipped methods on the REFIT example cohort after resampling to 15-minute resolution. The benchmark reports NMAE, NRMSE, sMAPE and MASE.

Format

A data frame with the same benchmark-result schema as `elcf4r_iflex_benchmark_results`.

Source

Derived from the local REFIT raw files with `data-raw/elcf4r_refit_artifacts.R`.

elcf4r_refit_example *REFIT example panel for package examples*

Description

A compact normalized panel extracted from the REFIT cleaned dataset after resampling to 15-minute resolution. The object contains complete days for one house and is intended for examples and lightweight benchmarking workflows.

Format

A data frame with normalized panel columns plus REFIT-specific fields:

dataset Dataset label, always `"refit"`.

entity_id Entity identifier, here the aggregate household channel.

timestamp,date,time_index,y,temp,dow,month,resolution_minutes Common normalized panel fields.

house_id REFIT house identifier derived from the file name.

channel Load channel name, for example `"Aggregate"`.

unix Minimum Unix timestamp within the resampling bucket.

issues Maximum issues flag within the resampling bucket.

Source

Public REFIT cleaned raw files, reduced with `data-raw/elcf4r_refit_artifacts.R`.

elcf4r_sgsc_example *Compact Smart-Grid Smart-City example panel*

Description

A compact normalized example panel following the schema of the Australian Smart-Grid Smart-City Customer Trial electricity interval-reading resource. It is intended for examples, vignettes and tests.

Format

A data frame with normalized columns plus SGSC-specific metadata:

dataset Dataset label, always `"sgsc"`.

entity_id Household or meter identifier.

timestamp,date,time_index,y,temp,dow,month,resolution_minutes Common normalized panel fields.

customer_id Source customer identifier from the SGSC CSV.

generation_kwh Interval generation metadata when available.

export_kwh Interval export metadata when available.

controlled_load_kwh Controlled-load metadata when available.

source_file Source CSV file name.

Details

The full raw 7zip archive is not redistributed with the package. Maintainers can rebuild a compact local example from the official Data.gov.au interval CSV with `data-raw/elcf4r_sgsc_artifacts.R`.

Source

Data.gov.au Smart-Grid Smart-City Customer Trial Data, Electricity Use Interval Reading resource. Compact local rebuild script: `data-raw/elcf4r_sgsc_artifacts.R`.

elcf4r_storenet_benchmark_results *StoreNet benchmark results for shipped forecasting methods*

Description

Saved rolling-origin benchmark results for the shipped methods on the local StoreNet household example. The benchmark is derived from complete 1-minute household days and reports NMAE, NRMSE, sMAPE and MASE for every shipped row. The clustered KWF variant is only included when the shipped StoreNet cohort is classified as thermosensitive.

Format

A data frame with the same benchmark-result schema as `elcf4r_iflex_benchmark_results`.

Source

Derived from the local StoreNet raw file with `data-raw/elcf4r_storenet_artifacts.R`.

`elcf4r_storenet_example`

StoreNet example panel for package examples

Description

A compact normalized panel extracted from the local StoreNet household file `H6_W.csv`. The object contains a small set of complete 1-minute household days and is intended for examples and lightweight benchmarking workflows.

Format

A data frame with normalized panel columns plus StoreNet-specific fields:

dataset Dataset label, always `"storenet"`.

entity_id Household identifier derived from the file name.

timestamp,date,time_index,y,temp,dow,month,resolution_minutes Common normalized panel fields.

discharge_w,charge_w,production_w Battery and production fields from the source file in watts.

state_of_charge_pct Battery state of charge in percent.

source_file Source CSV file name.

Source

Public StoreNet raw file `H6_W.csv`, reduced with `data-raw/elcf4r_storenet_artifacts.R`.

elcf4r_use_tensorflow_env

Select the Python environment used for TensorFlow-backed LSTM fits

Description

This helper provides an explicit, user-invoked way to bind the Python environment used by `reticulate` before calling `elcf4r_fit_lstm()`.

Usage

```
elcf4r_use_tensorflow_env(python = NULL, virtualenv = NULL, required = TRUE)
```

Arguments

`python`	Optional path to a Python interpreter passed to `reticulate::use_python()`.
`virtualenv`	Optional virtualenv name or path passed to `reticulate::use_virtualenv()`.
`required`	Logical passed to the corresponding `reticulate` selector.

Value

Invisibly returns the selected Python interpreter path when it can be determined.

Examples

```
if (interactive() &&
    requireNamespace("reticulate", quietly = TRUE) &&
    reticulate::virtualenv_exists("r-tensorflow")) {
  elcf4r_use_tensorflow_env(virtualenv = "r-tensorflow")
}
```

predict.elcf4r_kwf_clusters

Assign new segments to a fitted KWF clustering model

Description

Assign new segments to a fitted KWF clustering model

Usage

```
## S3 method for class 'elcf4r_kwf_clusters'
predict(object, segments, ...)
```

Arguments

object	An elcf4r_kwf_clusters object returned by elcf4r_kwf_cluster_days().
segments	Matrix or data frame of new daily segments.
...	Unused, present for method compatibility.

Value

A character vector of cluster labels.

---

predict.elcf4r_model *Predict from an* elcf4r_model

Description

Predict from an elcf4r_model

Usage

```
## S3 method for class 'elcf4r_model'
predict(object, newdata = NULL, ...)
```

Arguments

object	An elcf4r_model created by one of the package fit functions.
newdata	Optional new data for methods that need it. For gam and mars, this should be a long-format data frame with the same predictor columns used for fitting. For lstm, newdata may be a matrix or data frame of recent daily segments, or a list with elements segments and optional covariates.
...	Unused, present for method compatibility.

Value

Numeric predictions. For KWF and LSTM this is a forecast daily curve.

Index