Package 'FastRet'

October 25, 2026

Title Retention Time Prediction in Liquid Chromatography

Version 1.5.2

Description A framework for predicting retention times in liquid chromatography. Users can train custom models for specific chromatography columns, predict retention times using existing models, or adjust existing models to account for altered experimental conditions. The provided functionalities can be accessed either via the R console or via a graphical user interface. For details see Fadil et al. (2026) <doi:10.1021/acs.jcim.6c01344>. Related work: Bonini et al. (2020) <doi:10.1021/acs.analchem.9b05765>.

License GPL-3

Language en-US

URL https://github.com/spang-lab/FastRet/, https://spang-lab.github.io/FastRet/

BugReports https://github.com/spang-lab/FastRet/issues

biocViews Retention, Time, Chromatography, LC-MS

Encoding UTF-8

RoxygenNote 7.3.3

Depends R (>= 4.1.0)

Imports bslib, cluster, data.table, DBI, DT, future (>= 1.40.0), ggplot2, glmnet, htmltools, openxlsx, promises, rcdk, rlang, RSQLite, shiny (>= 1.8.1), shinyhelper, shinyjs, withr, xgboost

Suggests callr, caret, chromote, cli, devtools, knitr, languageserver, lintr, pkgdown, pkgbuild, pkgload, rmarkdown, servr, shinytest2, tibble, testthat (>= 3.0.0), toscutil, usethis

LazyData true

Config/testthat/edition 3

Config/testthat/parallel true

Config/testthat/start-first getCDs, parLapply2, get_predictors, plot_frm, train_frm-lasso, train_frm-gbtree, fit_gbtree

NeedsCompilation no

Author Tobias Schmidt [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0001-9681-9253>),
Christian Amesoeder [aut, cph] (ORCID:
<https://orcid.org/0000-0002-1668-8351>),
Marian Schoen [aut, cph],
Fadi Fadil [aut, cph] (ORCID: <https://orcid.org/0000-0002-9532-1901>),
Katja Dettmer [aut, cph] (ORCID:
<https://orcid.org/0000-0001-7337-2380>),
Peter Oefner [ths, cph] (ORCID:
<https://orcid.org/0000-0002-1499-3977>)

Maintainer Tobias Schmidt <tobias.schmidt331@gmail.com>

Repository CRAN

Date/Publication 2026-10-24 22:00:35 UTC

Contents

adjust_frm 2
clip_predictions 5
fastret_app 6
getCDs 7
plot_frm 8
predict.frm 9
preprocess_data 10
read_retip_hilic_data 11
read_rpadj_xlsx 13
read_rp_lasso_model_rds 13
read_rp_xlsx 14
RP 14
selective_measuring 15
start_gui 17
train_frm 18

Index **21**

`adjust_frm` *Adjust an existing FastRet model for use with a new column*

Description

The goal of this function is to train a model that predicts RT_ADJ (retention time measured on a new, adjusted column) from RT (retention time measured on the original column) and to attach this adjustment model to an existing FastRet model.

Usage

```
adjust_frm(
  frm,
  new_data,
  predictors = 1:6,
  nfolds = 5,
  verbose = 1,
  seed = NULL,
  do_cv = TRUE,
  adj_type = "lm",
  add_cds = NULL,
  match_rts = TRUE,
  match_keys = NULL,
  cache = TRUE
)
```

Arguments

frm	An object of class frm as returned by train_frm().
new_data	Data frame with required columns "RT", "NAME", "SMILES"; optional "INCHIKEY". "RT" must be the retention time measured on the adjusted column. Each row must match at least one row in frm$df. The exact matching behavior is described in 'Details'.
predictors	Numeric vector specifying which transformations to include in the model. Available options are: 1=RT, 2=RT^2, 3=RT^3, 4=log(RT), 5=exp(RT), 6=sqrt(RT). Note that predictor 1 (RT) is always included, even if not specified explicitly.
nfolds	The number of folds for cross validation.
verbose	Show progress messages?
seed	An integer value to set the seed for random number generation to allow for reproducible results.
do_cv	A logical value indicating whether to perform cross-validation. If FALSE, the cv element in the returned adjustment object will be NULL.
adj_type	A string representing the adjustment model type. Either "lm", "lasso", "ridge", or "gbtree".
add_cds	A logical value indicating whether to add chemical descriptors as predictors to new data. Default is TRUE if adj_type is "lasso", "ridge" or "gbtree" and FALSE if adj_type is "lm".
match_rts	Logical indicating how to obtain the RT feature used during training of the adjustment model. If TRUE (default), RT is obtained by matching rows in new_data to rows in frm$df based on match_keys. If FALSE, RT is obtained by applying the base model to new_data.
match_keys	Which columns to use for matching new_data to frm$df if match_rts is TRUE. Can be any combination of "INCHIKEY", "SMILES" and "NAME". If left at NULL, matching is done via "INCHIKEY" alone if complete INCHIKEYs are available in both new_data and frm$df, else via c("SMILES", "NAME").

INCHIKEY is a structure-invariant identifier, so it alone uniquely identifies a molecule; adding SMILES (matched as an exact raw string) would only make the key more fragile. See 'Details' for more information on the matching procedure.

cache — Cache chemical descriptors on disk (TRUE, default) or bypass the cache and recompute every descriptor (FALSE). Passed to `getCDs()`.

Details

Matching is done via `"INCHIKEY"` alone if both datasets have non-missing INCHIKEYs for all rows; otherwise via `c("SMILES", "NAME")`. This can be overridden by supplying `match_keys`. If `match_rts = FALSE`, no matching is performed and retention times are generated by calling `predict(frm, new_data, adjust = FALSE)`. If multiple rows in `frm$df` match the same row in `new_data`, their RT values are averaged first, and this average is used for training the adjustment model.

Example: if `frm$df` equals data.frame OLD shown below and `new_data` equals data.frame NEW, then the resulting, paired data.frame will look like PAIRED.

```
OLD <- data.frame(
    NAME   = c("A", "B",  "B",  "C"  ),
    SMILES = c("C", "CC", "CC", "CCC"),
    RT     = c(5.0,  8.0,  8.2,  9.0 )
)
NEW <- data.frame(
    NAME   = c("A", "B",  "B",  "B"),
    SMILES = c("C", "CC", "CC", "CC"),
    RT     = c(2.5,  5.5,  5.7,  5.6)
)
PAIRED <- data.frame(
    NAME   = c("A", "B",  "B",  "B"),
    SMILES = c("C", "CC", "CC", "CC"),
    RT     = c(5.0,  8.1,  8.1,  8.1), # Average of OLD$RT[2:3]
    RT_ADJ = c(2.5,  5.5,  5.7,  5.6)  # Taken from NEW
)
```

If `do_cv` is TRUE, the adjustment procedure is evaluated in cross-validation. However, care must be taken when interpreting the CV results, as the model performance depends on both the adjustment layer and the original model, which was trained on the full base dataset. Therefore, the observed CV metrics should be read as "expected performance when predicting RTs for molecules that were part of the base-model training but not part of the adjustment set" instead of "expected performance when predicting RTs for completely new molecules".

Value

An object of class `frm`, as returned by `train_frm()`, but with an additional element `adj` containing the adjustment model. Components of `adj` are:

- `model`: The fitted adjustment model. Class depends on `adj_type` and is one of `lm`, `glmnet`, or `xgb.Booster`.

- df: The data frame used for training the adjustment model. Including columns "NAME", "SMILES", "RT", "RT_ADJ" and optionally "INCHIKEY", as well as any additional predictors specified via the predictors argument.
- cv: A named list containing the cross validation results (see 'Details'), or NULL if do_cv = FALSE. When not NULL, elements are:
  - folds: A list of integer vectors specifying the samples in each fold.
  - models: A list of adjustment models trained on each fold.
  - stats: A list of vectors with RMSE, Rsquared, MAE, pBelow1Min per fold. Added with v1.3.0.
  - preds: Retention time predictions obtained during CV by applying the adjustment model to the hold-out data.
  - preds_adjonly: Removed (i.e. NULL) since v1.3.0.
- args: Function arguments used for adjustment (excluding frm, new_data and verbose). Added with v1.3.0.
- version: The version of the FastRet package used to train the adjustment model. Added with v1.3.0.

Examples

```
frm <- read_rp_lasso_model_rds()
new_data <- read_rpadj_xlsx()
frm_adj <- adjust_frm(frm, new_data, verbose = 0)
frm_adj <- adjust_frm(frm, new_data, verbose = 0, adj_type = "lasso")
```

---

clip_predictions *Clip predictions to observed range*

Description

Clips predicted retention times by fitting a log-normal distribution to the observed training RTs and bounding predictions to the central 99.99% interval. All observed RTs must be positive to estimate the distribution. If the estimated lower bound would be negative, it is replaced by 1% of the observed minimum RT instead.

Usage

```
clip_predictions(yhat, y)
```

Arguments

yhat	Numeric vector of predicted retention times.
y	Numeric vector of observed retention times used to derive bounds.

Value

Numeric vector of clipped (bounded) predictions.

Examples

```
# Draw only a few samples (10) and clip based on these. The allowed range will
# be much bigger than the observed range.

set.seed(42)
y <- rlnorm(n = 1000, meanlog = 2, sdlog = 0.1)
yhat <- y
yhat[1] <- -100 # way too low to be realistic
yhat[2] <- 1000 # way too high to be realistic
yhat <- clip_predictions(yhat, y)
range(y)  # [ 6.18,  8.93]
yhat[1:2] # [ 4.96, 10.61] # Limited by theoretical bounds


# Draw more samples (1000) and clip based on these. The allowed range will
# be almost identical to the observed range.

set.seed(42)
y <- rnorm(n = 100, mean = 100, sd = 5)
yhat <- y
yhat[1] <- -100
yhat[2] <- 1000
yhat <- clip_predictions(yhat, y)
range(y)  # 83.14, 117.47
yhat[1:2] # 83.14, 117.72
```

`fastret_app` *The FastRet GUI*

Description

Creates the FastRet GUI

Usage

```
fastret_app(port = 8080, host = "0.0.0.0", reload = FALSE, nsw = 1)
```

Arguments

`port`	The port the application should listen on
`host`	The address the application should listen on
`reload`	Whether to reload the application when the source code changes
`nsw`	The number of subworkers each worker is allowed to start. The higher this number, the faster individual tasks like model fitting can be processed.

Value

An object of class `shiny.appobj`.

Examples

```
x <- fastret_app()
if (interactive()) shiny::runApp(x)
```

`getCDs` *Get Chemical Descriptors for a list of molecules*

Description

Calculate Chemical Descriptors (CDs) for a list of molecules. Molecules can appear multiple times in the list.

Usage

```
getCDs(df, verbose = 1, nw = 1, keepdf = TRUE, cache = TRUE)
```

Arguments

`df`	dataframe with two mandatory columns: "NAME" and "SMILES"
`verbose`	0: no output, 1: progress, 2: more progress and warnings
`nw`	number of workers for parallel processing
`keepdf`	If TRUE, `cbind(df, CDs)` is returned. Else `CDs`.
`cache`	If TRUE (default), descriptors are cached on disk (an SQLite database under tools::R_user_dir("FastRet", "cache")): already-known descriptors are read from the cache and newly computed ones are written back, so repeated calls on the same molecules are fast and the cache is shared across processes. If FALSE, the cache is bypassed entirely and every descriptor is (re)computed via rCDK – useful for benchmarking the uncached runtime.

Value

A dataframe with all input columns (if `keepdf` is TRUE) and chemical descriptors as remaining columns.

Examples

```
cds <- getCDs(head(RP, 3), verbose = 1, nw = 1)
```

`plot_frm` *Plot predictions for a FastRet model*

Description

Creates scatter plots of measured vs. predicted retention times (RT) for a FastRet Model (FRM). Supports plotting cross-validation (CV) predictions and fitted predictions on the training set, as well as their adjusted variants when the model has been adjusted via `adjust_frm()`. Coloring highlights points within 1 minute of the identity line and simple outliers.

Usage

```
plot_frm(frm = train_frm(verbose = 1), type = "scatter.cv", trafo = "identity")
```

Arguments

`frm`	An object of class `frm` as returned by `train_frm()`.
`type`	Plot type. One of: • "scatter.cv": CV predictions for the training set • "scatter.cv.adj": CV predictions for the adjustment set (requires `frm$adj`) • "scatter.train": Model predictions for the training set • "scatter.train.adj": Adjusted model predictions for the adjustment set (requires `frm$adj`)
`trafo`	Transformation applied for display. One of: • "identity": no transformation • "log2": apply log2 transform to axes (metrics are computed on raw values)

Value

NULL, called for its side effect of plotting.

Examples

```
frm <- read_rp_lasso_model_rds()
plot_frm(frm, type = "scatter.cv")
```

predict.frm *Predict retention times using a FastRet Model*

Description

Predict retention times for new data using a FastRet Model (FRM).

Usage

```
## S3 method for class 'frm'
predict(
  object = train_frm(),
  df = object$df,
  adjust = NULL,
  verbose = 0,
  clip = TRUE,
  impute = TRUE,
  cache = TRUE,
  ...
)
```

Arguments

object	An object of class frm as returned by train_frm().
df	A data.frame with the same columns as the training data.
adjust	If object was adjusted using adjust_frm(), it will contain a property object$adj. If adjust is TRUE, object$adj will be used to adjust predictions obtained from object$model. If FALSE object$adj will be ignored. If NULL, object$model will be used, if available.
verbose	A logical value indicating whether to print progress messages.
clip	Clip predictions to be within RT range of training data?
impute	Impute missing predictor values using column means of training data?
cache	Cache chemical descriptors on disk (TRUE, default) or bypass the cache and recompute every descriptor (FALSE). Passed to getCDs().
...	Not used. Required to match the generic signature of predict().

Value

A numeric vector with the predicted retention times.

See Also

train_frm(), adjust_frm()

Examples

```
object <- read_rp_lasso_model_rds()
df <- head(RP)
yhat <- predict(object, df)
```

`preprocess_data` *Preprocess data*

Description

Preprocess data so they can be used as input for `train_frm()`.

Usage

```
preprocess_data(
  data,
  degree_polynomial = 1,
  interaction_terms = FALSE,
  verbose = 1,
  nw = 1,
  rm_near_zero_var = TRUE,
  rm_na = TRUE,
  add_cds = TRUE,
  rm_ucs = TRUE,
  rt_terms = 1,
  mandatory = c("NAME", "RT", "SMILES"),
  cache = TRUE
)
```

Arguments

`data` Dataframe with following columns:

- Mandatory: NAME, RT and SMILES.
- Recommmended: INCHIKEY.
- Optional: Any of the chemical descriptors listed in CDFeatures. All other columns will be removed. See 'Details'.

`degree_polynomial`
Add predictors with polynomial terms up to the specified degree, e.g. 2 means "add squares", 3 means "add squares and cubes". Set to 1 to leave descriptors unchanged.

`interaction_terms`
Add interaction terms? Polynomial terms are not included in the generation of interaction terms.

`verbose` 0: no output, 1: show progress, 2: progress and warnings.

`nw` Number of workers to use for parallel processing.

`rm_near_zero_var` Remove near zero variance predictors?

`rm_na` Remove NA values?

`add_cds` Add chemical descriptors using `getCDs()`? See 'Details'.

`rm_ucs` Remove unsupported columns?

`rt_terms` Which retention-time transformations to append as extra predictors. Supply a numeric vector referencing predefined rt_terms (1=RT, 2=I(RT^2), 3=I(RT^3), 4=log(RT), 5=exp(RT), 6=sqrt(RT)) or a character vector with the explicit transformation terms. Character values are passed to `model.frame()`, so they must use valid formula syntax (e.g. "I(RT^2)" rather than "RT^2").

`mandatory` Character vector of mandatory columns that must be present in `data`. If any of these columns are missing, an error is raised.

`cache` Passed to `getCDs()`: cache chemical descriptors on disk (TRUE, default) or bypass the cache and recompute every descriptor (FALSE).

Details

If `add_cds = TRUE`, chemical descriptors are added using `getCDs()`. If **all** chemical descriptors listed in CDFeatures are already present in the input `data` object, `getCDs()` will leave them unchanged. If one or more chemical descriptors are missing, **all** chemical descriptors will be recalculated and existing ones will be overwritten.

Value

A dataframe with the preprocessed data.

Examples

```
data <- head(RP, 3)
pre <- preprocess_data(data, verbose = 0)
```

`read_retip_hilic_data` *Read the HILIC dataset from the Retip package*

Description

Reads the `Retip::HILIC` dataset (CC BY 4.0) from the Retip package or, if Retip is not installed, downloads the dataset directly from the Retip GitHub repository. Before returning the dataset, SMILES strings are canonicalized and the original `tibble` object is converted to a base R `data.frame`.

Usage

```
read_retip_hilic_data(verbose = 1)
```

Arguments

verbose	Verbosity. 1 for messages, 0 to suppress them.

Details

Attribution as required by CC BY 4.0:

- Original dataset by: Paolo Bonini, Tobias Kind, Hiroshi Tsugawa, Dinesh Kumar Barupal, and Oliver Fiehn as part of the Retip project.
- Source repository: https://github.com/oloBion/Retip
- Original file: https://github.com/oloBion/Retip/raw/master/data/HILIC.RData
- License: CC BY 4.0 (https://creativecommons.org/licenses/by/4.0/)
- Modifications in FastRet:
  - converted tibble to data.frame
  - canonicalized SMILES using as_canonical()
  - renamed column 'INCHKEY' to 'INCHIKEY'

Value

A data frame with 970 rows and the following columns:

- NAME: Molecule name
- INCHIKEY: InChIKey
- SMILES: Canonical SMILES string
- RT: Retention time in Minutes

Source

https://github.com/oloBion/Retip/raw/master/data/HILIC.RData

References

Retip: Retention Time Prediction for Compound Annotation in Untargeted Metabolomics
Paolo Bonini, Tobias Kind, Hiroshi Tsugawa, Dinesh Kumar Barupal, and Oliver Fiehn
Analytical Chemistry 2020 92 (11), 7515-7522 DOI: 10.1021/acs.analchem.9b05765

Examples

```
df <- read_retip_hilic_data(verbose = 0)
```

read_rpadj_xlsx *Retention times re-measured under a steeper gradient*

Description

25 metabolites from the RP dataset, re-measured on the same reverse-phase column under a **steeper gradient** (real data, dataset `RP_Steep` in the FastRet paper). Used to demonstrate model adjustment via `adjust_frm()`.

Usage

```
read_rpadj_xlsx()
```

Value

A dataframe with 25 rows (metabolites) and 4 columns: RT, NAME, SMILES and INCHIKEY.

Examples

```
x <- read_rpadj_xlsx()
```

read_rp_lasso_model_rds *LASSO Model trained on RP dataset*

Description

Read a LASSO model trained on the RP dataset using `train_frm()`.

Usage

```
read_rp_lasso_model_rds()
```

Value

A `frm` object.

Examples

```
frm <- read_rp_lasso_model_rds()
```

read_rp_xlsx *Read retention times (RT) measured on a reverse phase (RP) column*

Description

Reads retention times from a reverse phase liquid chromatography experiment, performed at 35°C and a flow rate of 0.3 mL/min. The data is also available as a dataframe in the package; to access it directly, use RP.

Usage

```
read_rp_xlsx()
```

Value

A dataframe of 458 metabolites with columns `RT`, `SMILES`, `NAME` and `INCHIKEY`.

Source

Measured by the Institute of Functional Genomics at the University of Regensburg.

See Also

RP

Examples

```
x <- read_rp_xlsx()
all.equal(x, RP)
```

RP *Retention Times (RT) measured on a Reverse Phase (RP) Column*

Description

Retention time data from a reverse phase liquid chromatography measured with a temperature of 35°C and a flowrate of 0.3ml/min. The same data is available as an xlsx file in the package. To read it into R use `read_rp_xlsx()`. @format A dataframe of 458 metabolites with the following columns:

RT Retention time

SMILES SMILES notation of the metabolite

NAME Name of the metabolite

INCHIKEY InChIKey of the metabolite

Usage

```
RP
```

Format

An object of class `data.frame` with 458 rows and 4 columns.

Source

Measured by the Institute of Functional Genomics at the University of Regensburg.

See Also

read_rp_xlsx

`selective_measuring` *Selective Measuring*

Description

The function `adjust_frm()` is used to modify existing FastRet models based on changes in chromatographic conditions. It requires a set of molecules with measured retention times on both the original and new column. This function selects a sensible subset of molecules from the original dataset for re-measurement. The selection process includes:

1. Generating chemical descriptors from the SMILES strings of the molecules. These are the features used by `train_frm()` and `adjust_frm()`.″
2. Standardizing chemical descriptors to have zero mean and unit variance.
3. Training a Ridge Regression model with the standardized chemical descriptors as features and the retention times as the target variable.
4. Scaling the chemical descriptors by coefficients of the Ridge Regression model.
5. Clustering the entire dataset, which includes the scaled chemical descriptors and the retention times.
6. Returning the clustering results, which include the cluster assignments, the medoid indicators, and the raw data.

Usage

```
selective_measuring(
  raw_data,
  k_cluster = 25,
  verbose = 1,
  seed = NULL,
  rt_coef = "max_ridge_coef"
)
```

Arguments

raw_data	The raw data to be processed. Must be a dataframe with columns NAME, RT and SMILES.
k_cluster	The number of clusters for PAM clustering.
verbose	The level of verbosity.
seed	An optional random seed for reproducibility, set at the beginning of the function.
rt_coef	Which coefficient to use for scaling RT before clustering. Options are: • 0: exclude RT from the clustering. • 'max' / 'max_ridge_coef': scale with the maximum absolute coefficient obtained in ridge regression. I.e., RT will have approximately the same weight as the most important chemical descriptor. • 1: do not scale RT any further, i.e., use standardized RT. The effect of leaving RT unscaled is kind of unpredictable, as the ridge coefficients depend on the dataset. If the maximum absolute coefficient is much smaller than 1, RT will dominate the clustering. If it is much larger than 1, RT will have little influence on the clustering. • 'inf': set all chemical descriptor values to zero, i.e., RT is "infinitely" more important than any chemical descriptor.

Value

A list containing the following elements:

- clustering: A data frame with columns RT, SMILES, NAME, CLUSTER and IS_MEDOID.
- clobj: The clustering object. The object returned by the clustering function. Depends on the method parameter.
- coefs: The coefficients from the Ridge Regression model.
- model: The Ridge Regression model.
- df: The preprocessed data.
- dfz: The standardized features.
- dfzb: The features scaled by the coefficients (betas) of the Ridge Regression model.

Examples

```
x <- selective_measuring(RP[1:50, ], k = 5, verbose = 0)
# For the sake of a short runtime, only the first 50 rows of the RP dataset
# were used in this example. In practice, you should always use the entire
# dataset to find the optimal subset for re-measurement.
```

`start_gui` *Start the FastRet GUI*

Description

Starts the FastRet GUI

Usage

```
start_gui(port = 8080, host = "0.0.0.0", reload = FALSE, nw = 2, nsw = 1)
```

Arguments

`port`	The port the application should listen on
`host`	The address the application should listen on
`reload`	Whether to reload the application when the source code changes
`nw`	The number of worker processes started. The first worker always listens for user input from the GUI. The other workers are used for handling long running tasks like model fitting or clustering. If `nw` is 1, the same process is used for both tasks, which means that the GUI will become unresponsive during long running tasks.
`nsw`	The number of subworkers each worker is allowed to start. The higher this number, the faster individual tasks like model fitting can be processed. A value of 1 means that all subprocesses will run sequentially.

Details

If you set `nw = 3` and `nsw = 4`, you should have at least 16 cores, one core for the shiny main process. Three cores for the three worker processes and 12 cores (3 * 4) for the subworkers. For the default case, `nworkers = 2` and `nsw = 1`, you only need 3 cores, as `nsw = 1` means that all subprocesses will run sequentially.

Value

A shiny app. This function returns a shiny app that can be run to interact with the model.

Examples

```
if (interactive()) start_gui()
```

train_frm *Train a new FastRet model (FRM) for retention time prediction*

Description

Trains a new model from molecule SMILES to predict retention times (RT) using the specified method.

Usage

```
train_frm(
  df,
  method = "lasso",
  verbose = 1,
  nfolds = 5,
  nw = 1,
  degree_polynomial = 1,
  interaction_terms = FALSE,
  rm_near_zero_var = TRUE,
  rm_na = TRUE,
  rm_ns = FALSE,
  seed = NULL,
  do_cv = TRUE,
  cache = TRUE,
  tune_grid = NULL
)
```

Arguments

df	A dataframe with columns "NAME", "RT", "SMILES" and optionally a set of chemical descriptors. If no chemical descriptors are provided, they are calculated using the function preprocess_data().
method	A string representing the prediction algorithm. Either "lasso", "ridge", "gbtree", "gbtreeDefault" or "gbtreeRP". Method "gbtree" is an alias for "gbtreeDefault".
verbose	A logical value indicating whether to print progress messages.
nfolds	An integer representing the number of folds for cross validation.
nw	An integer representing the number of workers for parallel processing.
degree_polynomial	An integer representing the degree of the polynomial. Polynomials up to the specified degree are included in the model.
interaction_terms	A logical value indicating whether to include interaction terms in the model.
rm_near_zero_var	A logical value indicating whether to remove near zero variance predictors.

rm_na — A logical value indicating whether to remove NA values before training. Highly recommended to avoid issues during model fitting. Setting this to FALSE with method = "lasso" will most likely lead to errors.

rm_ns — A logical value indicating whether to remove chemical descriptors that were considered as not suitable for linear regression based on a previous analysis of an independent dataset. Currently not used.

seed — An integer value to set the seed for random number generation to allow for reproducible results.

do_cv — A logical value indicating whether to perform cross-validation. If FALSE, the cv element in the returned object will be NULL.

cache — Cache chemical descriptors on disk (TRUE, default) or bypass the cache and recompute every descriptor (FALSE). Passed to getCDs().

tune_grid — Optional hyperparameter grid for gbtree/BRT models. Either a data.frame of xgboost hyperparameters (one combination per row, e.g. columns max_depth, eta, gamma, colsample_bytree, subsample, min_child_weight) or one of the built-in grid-size keywords "tiny", "small" or "large" (see get_param_grid(); "small" is a fast 8-combination Retip-style grid). When supplied, the fixed default parameters are replaced by a cross-validated grid search over the grid (via find_params_best()); the number of boosting rounds is still chosen by early stopping. Ignored (with a warning) for non-BRT methods. NULL (default) keeps the fast stock parameters.

Value

A 'FastRet Model', i.e., an object of class frm. Components are:

- model: The fitted base model. This can be an object of class glmnet (for Lasso or Ridge regression) or xgb.Booster (for GBTree models).
- df: The data frame used for training the model. The data frame contains all user-provided columns (including mandatory columns RT, SMILES and NAME) as well the calculated chemical descriptors. (But no interaction terms or polynomial features, as these can be recreated within a few milliseconds).
- cv: A named list containing the cross validation results, or NULL if do_cv = FALSE. When not NULL, elements are:
  - folds: A list of integer vectors specifying the samples in each fold.
  - models: A list of models trained on each fold.
  - stats: A list of vectors with RMSE, Rsquared, MAE, pBelow1Min per fold.
  - preds: Retention time predictions obtained in CV as numeric vector.
- seed: The seed used for random number generation.
- version: The version of the FastRet package used to train the model.
- args: The value of function arguments besides df as named list.

Examples

```
m <- train_frm(df = RP[1:40, ], method = "lasso", nfolds = 2, verbose = 0)
# For the sake of a short runtime, only the first 40 rows of the RP dataset
# are used in this example. In practice, you should always use the entire
# training dataset for model training.
```

Index

∗ **dataset**
    read_retip_hilic_data, 11
    read_rp_lasso_model_rds, 13
    read_rp_xlsx, 14
    read_rpadj_xlsx, 13
    RP, 14
∗ **public**
    adjust_frm, 2
    clip_predictions, 5
    fastret_app, 6
    getCDs, 7
    plot_frm, 8
    predict.frm, 9
    preprocess_data, 10
    selective_measuring, 15
    start_gui, 17
    train_frm, 18

adjust_frm, 2
adjust_frm(), *8, 9*, *13*, *15*
as_canonical(), *12*

CDFeatures, *10, 11*
clip_predictions, 5

fastret_app, 6

getCDs, 7
getCDs(), *4*, *9*, *11*, *19*

model.frame(), *11*

plot_frm, 8
predict.frm, 9
preprocess_data, 10
preprocess_data(), *18*

read_retip_hilic_data, 11
read_rp_lasso_model_rds, 13
read_rp_xlsx, 14
read_rpadj_xlsx, 13
RP, *13, 14*, 14

selective_measuring, 15
start_gui, 17

tools::R_user_dir, *7*
train_frm, 18
train_frm(), *3, 4*, *8–10*, *13*, *15*