Package 'hydrorecipes'

June 27, 2022

Type Package

Title Hydrogeology Steps for the 'recipes' Package

Version 0.0.3

Date 2022-06-24

Maintainer Jonathan Kennel <jkennel@uoguelph.ca>

Description Additional steps to be used with the 'recipes' package. New steps were designed for regression deconvolution on datasets with millions of rows with applications to signal decomposition and response characterization. The methods in this package were developed as part of PhD thesis titled High Frequency Water Level Responses to Natural Signals <http://hdl.handle.net/10214/17890> by Jonathan Kennel in 2020.

BugReports https://github.com/jkennel/hydrorecipes/issues

URL https://github.com/jkennel/hydrorecipes

License GPL-3

Depends R (>= 4.1.0), recipes (>= 0.1.15)

Imports Rcpp (>= 1.0.7), earthtide, generics, splines, tibble, dplyr, rlang, tidyr, fftw

LinkingTo Rcpp, RcppArmadillo, RcppParallel

RoxygenNote 7.2.0

Encoding UTF-8

Suggests knitr, rmarkdown, ggplot2, scales, broom, glmnet, testthat (>= 3.0.0), covr, splines2

VignetteBuilder knitr

LazyData true

NeedsCompilation yes

SystemRequirements C++11

Config/testthat/edition 3

Author Jonathan Kennel [aut, cre], Beth Parker [ths]

Repository CRAN

Date/Publication 2022-06-27 07:30:04 UTC

R topics documented:

`log_lags` *log_lags*

Description

Generate lags or knots with logarithmic spacing. Lags/knots start at 0. Lags are in terms of samples. For example, if samples are taken every 60, a lag of 1 corresponds to 60 seconds, a lag of 4 would correspond to 240 seconds.

Usage

```
log_lags(n, max_time_lag)
```

Arguments

`n`	The total number of lags (integer)
`max_time_lag`	The maximum lag in number of samples (integer)

Value

An integer vector of lags

Examples

```
log_lags(12, 86400)
```

predict_terms *predict_terms*

Description

Predict the contribution for each step.

Usage

```
predict_terms(fit, rec, data, ...)

## S3 method for class 'lm'
predict_terms(fit, rec, data, ...)

## S3 method for class 'cv.glmnet'
predict_terms(fit, rec, data, ...)

## S3 method for class 'numeric'
predict_terms(fit, rec, data, ...)
```

Arguments

fit	A model object that has a coefficients method (e.g. lm)
rec	A prepped recipe
data	A data.frame with feature columns
...	Currently not used

Value

A data.frame of predicted values for each step (component contribution).

Examples

```
data(transducer)
transducer$datetime_num <- as.numeric(transducer$datetime)

rec_toll_rasmussen <- recipe(wl ~ baro + et + datetime_num, transducer) |>
   step_lead_lag(baro, lag = log_lags(100, 86400 * 2 / 120)) |>
   step_ns(datetime_num, deg_free = 10) |>
   prep()

input_toll_rasmussen <- rec_toll_rasmussen |> bake(new_data = NULL)

fit_toll_rasmussen <- lm(wl ~ ., input_toll_rasmussen)
pred <- predict_terms(fit_toll_rasmussen,
                      rec_toll_rasmussen,
                      input_toll_rasmussen)
```

response *response*

Description

This function takes a model object and extracts the responses from `step_distributed_lag`, `step_lead_lag`, `step_harmonic` and `step_earthtide`.

Usage

```
response(fit, rec, verbose = FALSE, ...)

## S3 method for class 'lm'
response(fit, rec, verbose = FALSE, ...)

## S3 method for class 'cv.glmnet'
response(fit, rec, verbose = FALSE, ...)

## S3 method for class 'numeric'
response(fit, rec, verbose = FALSE, ...)
```

Arguments

`fit`	A model object that has a `coefficients` method (e.g. `lm`)
`rec`	A prepped `recipe`
`verbose`	Print names of steps with no response methods
`...`	Currently not used

Details

`step_distributed_lag` and `step_lead_lag` result in impulse response functions and `step_harmonic` and `step_earthtide` result in harmonic components (amplitude and phase for each *main* frequency).

Value

A data.frame of impulse response functions, or harmonic components corresponding to each step.

Examples

```
data(transducer)
transducer$datetime_num <- as.numeric(transducer$datetime)

rec_toll_rasmussen <- recipe(wl~baro + datetime_num, transducer) |>
   step_lead_lag(baro, lag = log_lags(100, 86400 * 2 / 120)) |>
   step_ns(datetime_num, deg_free = 10) |>
   prep()
```

```
input_toll_rasmussen <- rec_toll_rasmussen |> bake(new_data = NULL)

fit_toll_rasmussen <- lm(wl~., input_toll_rasmussen)
resp <- response(fit_toll_rasmussen,
                 rec_toll_rasmussen)
plot(value~x, resp[resp$name == 'cumulative',], type = 'l')
```

`step_distributed_lag` *Create a distributed lagged predictor*

Description

`step_distributed_lag` creates a *specification* of a recipe step that will add new basis lag columns. The new data will include NA values up to the maximum lag. These can be removed with `recipes::step_naomit()`. The inspiration for this step comes from the dlnm package. For large datasets with large maximum time lags, convolution is done in the frequency domain for efficiency. Samples should be ordered and have regular spacing (i.e. regular time series, regular spatial sampling).

Usage

```
step_distributed_lag(
  recipe,
  ...,
  role = "predictor",
  trained = FALSE,
  knots = NULL,
  basis_mat = NULL,
  spline_fun = splines::ns,
  options = list(intercept = TRUE),
  prefix = "distributed_lag_",
  keep_original_cols = FALSE,
  columns = NULL,
  skip = FALSE,
  id = rand_id("distributed_lag")
)
```

Arguments

`recipe`	A recipe object. The step will be added to the sequence of operations for this recipe.
`...`	One or more selector functions to choose variables for this step. See `selections()` for more details.
`role`	For model terms created by this step, what analysis role should they be assigned? By default, the new columns created by this step from the original variables will be used as *predictors* in a model.
`trained`	A logical to indicate if the quantities for preprocessing have been estimated.

knots — An integer vector of breakpoints to define the spline. These should include the Boundary.knots. See splines for more info.

basis_mat — The matrix of basis kernels to convolve. This is NULL until computed by prep.recipe(). This can also be specified as an object generated from the splines or splines2 packages having attributes for knots and Boundary.knots. If specified like this knots will be obtained from the basis_mat and not from the knots parameter.

spline_fun — Function used for calculating basis_mat. This should return an object having knots and Boundary.knots attributes.

options — The arguments to pass to spline_fun.

prefix — A prefix for generated column names, default to "distributed_lag_".

keep_original_cols — A logical to keep the original variables in the output. Defaults to FALSE.

columns — A character string of variable names that will be populated elsewhere.

skip — A logical. Should the step be skipped when the recipe is baked by bake()? While all operations are baked when prep() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations.

id — A character string that is unique to this step to identify it.

Details

This step assumes that the data are already *in the proper sequential order* for lagging. The input should be sampled at a regular interval (time, space, etc.). When the recipe is baked a set of vectors resulting from the convolution of a vector and a basis matrix is returned. Distributed lags can be used to model a delayed response to a input in a flexible manner with fewer regressor terms. The method achieves this by convolving a input stress with a basis lag matrix (commonly spline function) which leads to a set of regressors with fewer terms but still capable of describing both fast and slow responses.

Value

An updated version of recipe with the new step added to the sequence of any existing operations.

References

Almon, S (1965). The Distributed Lag Between Capital Appropriations and Expenditures. Econometrica 33(1), 178.

Gasparrini A. Distributed lag linear and non-linear models in R: the package dlnm. Journal of Statistical Software. 2011; 43(8):1-20. https://doi.org/10.18637/jss.v043.i08

See Also

step_lead_lag() recipes::step_lag()

Other row operation steps: step_lead_lag()

Examples

```
data(wipp30)

rec_base <- recipe(wl~baro, data = wipp30)

# default uses splines::ns
rec <- rec_base |>
  step_distributed_lag(baro,
                       knots = log_lags(4, 72)) |>
  prep()

# use different spline function
rec <- rec_base |>
  step_distributed_lag(baro,
                       spline_fun = splines::bs,
                       options = list(intercept = TRUE,
                                      degree = 4L),
                       knots = log_lags(4, 72)) |>
  prep()

# specify basis_mat
basis_mat <- splines2::mSpline(0:72, knots = c(3,16))
rec <- rec_base |>
  step_distributed_lag(baro,
                       basis_mat = basis_mat) |>
  prep()
```

---

step_earthtide *Earth tide response*

Description

`step_earthtide` creates a *specification* of a recipe step that are the Earth tide harmonics for a particular location. This step requires the earthtide package.

Usage

```
step_earthtide(
  recipe,
  ...,
  role = "predictor",
  trained = FALSE,
  do_predict = FALSE,
  method = "gravity",
  astro_update = 1L,
  latitude = 0,
  longitude = 0,
  elevation = 0,
```

```
  azimuth = 0,
  gravity = 0,
  earth_radius = 6378136.3,
  earth_eccen = 0.0066943979514,
  cutoff = 1e-06,
  wave_groups = NULL,
  catalog = "ksm04",
  eop = NULL,
  scale = TRUE,
  prefix = "earthtide_",
  columns = NULL,
  keep_original_cols = FALSE,
  skip = FALSE,
  id = rand_id("earthtide")
)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose which variables are affected by the step. See selections() for more details. For the tidy method, these are not currently used.
role	Defaults to "earthtide"
trained	A logical to indicate if the quantities for preprocessing have been estimated.
do_predict	run in predict or analyze mode
method	One or more of "gravity", "tidal_potential", "tidal_tilt", "vertical_displacement", "horizontal_displacement", "n_s_displacement", "e_w_displacement", "vertical_strain", "areal_strain", "volume_strain", "horizontal_strain", or "ocean_tides", "pole_tide", "lod_tide". The pole tide and lod_tide are used in predict mode even if do_predict is FALSE. More than one value can only be used if do_predict == TRUE.
astro_update	Integer that determines how often to phases are updated in number of samples. Defaults to 1 (every sample), but speed gains are realized with larger values. Typically updating every hour will have speed gains and keep precision (ie 3600 for one second data, 60 for minute data, 1 for hourly data).
latitude	The station latitude (numeric) defaults to 0.
longitude	The station longitude (numeric) defaults to 0.
elevation	The station elevation (m) (numeric) defaults to 0.
azimuth	Earth azimuth (numeric) defaults to 0.
gravity	Gravity at the station (m/s^2) (numeric) 0 to estimate gravity from elevation and latitude.
earth_radius	Radius of earth (m) (numeric) defaults to 6378136.3
earth_eccen	Eccentricity of earth (numeric) defaults to 6.69439795140e-3
cutoff	Cutoff amplitude for constituents (numeric) defaults to 1e-6.

wave_groups	Two column data.frame having start and end of frequency groups (data.frame). This data.frame must have two columns with the names 'start', and 'end' signifying the start and end of the wave groupings. An optional third column 'multiplier' can be provided to scale the particular wave group. If column names do no match, the inferred column positions are start, end, multiplier.
catalog	Use the "hw95s" catalog or "ksm04" catalog (character).
eop	User defined Earth Orientation Parameter (EOP) data.frame with the following columns: datetime, ddt, ut1_utc, lod, x, y, dx, dy
scale	Scale results when do_predict is FALSE
prefix	A prefix for generated column names, default to "earthtide_".
columns	A character string of variable names that will be populated (eventually) by the terms argument.
keep_original_cols	A logical to keep the original variables in the output. Defaults to FALSE.
skip	A logical. Should the step be skipped when the recipe is baked by bake()? While all operations are baked when prep() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations.
id	A character string that is unique to this step to identify it.

Details

There are many waves (thousands) that make up a tidal signal. step_earthtide calculates the Earth tide signal for a time and location. The tidal signal can be estimated as a single summed curve when do_predict = TRUE or as a set of wave groups when do_predict = FALSE. Wave groups are ranges of frequencies identified by start and end frequencies. For example, if you have one month of data the M2 signal could be described as the sum of all the waves between 1.914129 and 1.950419 cycles per day. The regressors for each wave group have a sin and cos component. See recipes::step_harmonic() for a simplified version of this where each sin and cos curve corresponds to a single wave.

Value

An updated version of recipe with the new step added to the sequence of existing steps (if any).

See Also

step_earthtide() earthtide::calc_earthtide() recipes::step_harmonic()

Examples

```
library(earthtide)
data(eterna_wavegroups)
data(transducer)

transducer <- transducer[, c('datetime', 'wl'),]
```

```
t_sub <- transducer[(as.numeric(transducer$datetime) %% 14400) == 0, ]
wg <- na.omit(eterna_wavegroups[eterna_wavegroups$time == '1 month',])

recipe(wl ~ ., data = t_sub) |>
  step_earthtide(datetime,
                 latitude = 34,
                 longitude = -118.5,
                 wave_groups = wg,
                 do_predict = FALSE) |>
  prep()

recipe(wl ~ ., data = t_sub) |>
  step_earthtide(datetime,
                 latitude = 34,
                 longitude = -118.5,
                 wave_groups = wg,
                 do_predict = TRUE) |>
  prep()
```

`step_lead_lag` *Create a lead predictor*

Description

`step_lead_lag` creates a *specification* of a recipe step that will add new columns that are shifted forward (lag) or backward (lead). Data will by default include NA values where the shift was induced. These can be removed with `recipes::step_naomit()`. Samples should be ordered and have regular spacing (i.e. regular time series, regular spatial sampling).

Usage

```
step_lead_lag(
  recipe,
  ...,
  role = "predictor",
  trained = FALSE,
  lag = 1,
  n_subset = 1,
  n_shift = 0,
  prefix = "lead_lag_",
  keep_original_cols = FALSE,
  columns = NULL,
  skip = FALSE,
  id = rand_id("lead_lag")
)
```

Arguments

recipe	A recipe object. The step will be added to the sequence of operations for this recipe.
...	One or more selector functions to choose variables for this step. See selections() for more details.
role	For model terms created by this step, what analysis role should they be assigned? By default, the new columns created by this step from the original variables will be used as *predictors* in a model.
trained	A logical to indicate if the quantities for preprocessing have been estimated.
lag	A vector of integers. Each specified column will be lagged for each value in the vector. Negative values are accepted and indicate leading the vector (i.e. the reverse of lagging)
n_subset	A single integer. Subset every n_subset values.
n_shift	A single integer amount to shift results in number of observations.
prefix	A prefix for generated column names, default to "lag_lead_".
keep_original_cols	A logical to keep the original variables in the output. Defaults to FALSE.
columns	A character string of variable names that will be populated (eventually) by the terms argument.
skip	A logical. Should the step be skipped when the recipe is baked by bake()? While all operations are baked when prep() is run, some operations may not be able to be conducted on new data (e.g. processing the outcome variable(s)). Care should be taken when using skip = TRUE as it may affect the computations for subsequent operations.
id	A character string that is unique to this step to identify it.

Details

This step assumes that the data are already *in the proper sequential order* for lagging. This step allows a vector to be shifted forward (lag) or backward (lead). While forward shifts are commonly used for lagged responses, there are cases where a backward shift may be useful. This can arise when there are unknown clock errors between two sensors making the response appear to occur before the input. Another situation where a backward shift may be useful is in cyclical signals where alignment is unknown. The data can also efficiently be subsetted during the lag/leading process resulting in smaller model inputs while still utilizing the entire lag/lead history.

Value

An updated version of recipe with the new step added to the sequence of any existing operations.

See Also

recipes::step_lag()

Other row operation steps: step_distributed_lag()

Examples

```
data(wipp30)

recipe(wl~., data = wipp30) |>
  step_lead_lag(baro, lag = -2:2, n_subset = 1, n_shift = 0) |>
  prep()

recipe(wl~ ., data = wipp30) |>
  step_lead_lag(baro, lag = -2:2, n_subset = 2, n_shift = 0) |>
  prep()

recipe(wl~ ., data = wipp30) |>
  step_lead_lag(baro, lag = -2:2, n_subset = 2, n_shift = 1) |>
  prep()
```

---

tidy2 *tidy2*

Description

Turn an object into a tidy2 tibble.

Usage

```
tidy2(x, ...)
```

Arguments

x	An object to be converted into a tidy `tibble::tibble()`.
...	Additional arguments to tidying method.

Value

A `tibble::tibble()` with information about model components.

---

tidy2.step_distributed_lag *tidy2.recipe*

Description

`tidy2` will return a data frame that contains information regarding a recipe or operation within the recipe (when a `tidy2` method for the operation exists). This method ensures that relevant data for `predict_terms` and `response` can be easily accessed from a recipe formulation.

Usage

```
## S3 method for class 'step_distributed_lag'
tidy2(x, ...)

## S3 method for class 'step_earthtide'
tidy2(x, ...)

## S3 method for class 'step_lead_lag'
tidy2(x, ...)

## S3 method for class 'recipe'
tidy2(x, number = NA, id = NA, ...)

## S3 method for class 'step'
tidy2(x, ...)

## S3 method for class 'check'
tidy2(x, ...)

## S3 method for class 'step_ns'
tidy2(x, ...)

## S3 method for class 'step_intercept'
tidy2(x, ...)
```

Arguments

x	A recipe object, step, or check (trained or otherwise).
...	Not currently used.
number	An integer or NA. If missing and id is not provided, the return value is a list of the operations in the recipe. If a number is given, a tidy method is executed for that operation in the recipe (if it exists). number must not be provided if id is.
id	A character string or NA. If missing and number is not provided, the return value is a list of the operations in the recipe. If a character string is given, a tidy method is executed for that operation in the recipe (if it exists). id must not be provided if number is.

Value

A tibble with columns that vary depending on what tidy2 method is executed. When number and id are NA, a tibble with columns number (the operation iteration), operation (either "step" or "check"), type (the method, e.g. 'lead_lag', 'distributed_lag'), a logical column called trained for whether the operation has been estimated using prep, a logical for skip, and a character column id.

transducer *transducer*

Description

This data.frame contains the water levels, barometric pressure, and synthetic Earth tides from 2016-08-25 to 2016-10-15 (sub-sampled to every 2 minutes).

Usage

```
transducer
```

Format

A data.frame The columns are:

datetime POSIXct date and time

wl transducer pressure of water column and air (dbar)

baro barometric pressure (dbar)

et synthetic gravity

Examples

```
data(transducer)
```

wipp30 *wipp30*

Description

This data.frame contains the water levels, barometric pressure, and Earth tides for wipp30. This is the dataset included in BETCO.

Usage

```
wipp30
```

Format

A data.frame The columns are:

time elapsed time in hours

wl water level

baro barometric pressure

et synthetic gravity

References

Toll NJ, Rasmussen TC, 2007, "Removal of barometric pressure effects and Earth tides from observed water levels", Ground Water 45(1):101-105

Examples

```
data(wipp30)
```

Index