Package: waterlevel (via r-universe)

May 28, 2026

Type Package

Title Well Water Level Analysis Tools

Version 0.0.1

Maintainer Jonathan Kennel <jkennel@uoguelph.ca>

Description Well water level analysis tools for R. The current focus is on barometric efficiency.

License GPL-3

Depends R (>= 3.1.0), recipes (>= 0.1.4)

Imports Rcpp (>= 1.0.1), RcppArmadillo, RcppParallel, data.table, tidyselect, earthtide(>= 0.0.7), splines, tibble, plotly, purrr, dplyr, rlang, fftw, stats

LinkingTo Rcpp, RcppArmadillo, RcppParallel

RoxygenNote 7.3.2

Encoding UTF-8

Suggests testthat (>= 2.1.0), knitr, rmarkdown

VignetteBuilder knitr

Config/pak/sysreqs cmake libfftw3-dev make libicu-dev libuv1-dev libssl-dev

Repository https://rpkgs.r-universe.dev

Date/Publication 2025-09-07 10:04:43 UTC

RemoteUrl https://github.com/rpkgs/waterlevel

RemoteRef HEAD

RemoteSha 6f4071798a608c62ed86e9e7cfb404f4c2670967

Contents

waterlevel-package 3
add_level_shifts 4
AIC_mlm 5
baro_elev 5
be_acworth 6

be_acworth_eq_4 . . . 7
be_clark . . . 8
be_correct . . . 9
be_high_low . . . 10
be_least_squares . . . 11
be_least_squares_diff . . . 12
be_rahi . . . 13
be_ratio . . . 14
be_storage . . . 16
be_visual . . . 16
be_visual_data . . . 18
be_visual_plot . . . 19
brf_from_frf . . . 19
coh_phase . . . 20
convolve_fft . . . 20
cross_basis_fft . . . 21
det_parallel . . . 21
diff_shift . . . 22
distributed_lag . . . 22
distributed_lag_parallel . . . 23
energy_density_wang . . . 24
fftw_convolve . . . 24
find_gaps . . . 25
find_level_shift . . . 26
fit_gaps . . . 26
formula_from_recipe . . . 27
frf_predict_terms . . . 27
gap_fill . . . 28
gap_fill2 . . . 29
get_fit_summary . . . 29
get_intercept_stats . . . 30
get_level_shift_coef . . . 30
get_shift . . . 31
harmonic . . . 31
konno_ohmachi_parallel . . . 32
konno_ohmachi_serial . . . 32
lag_difference . . . 33
lag_matrix . . . 34
level_shift_to_datetime . . . 34
log_lags . . . 35
pad_kernel . . . 35
portion . . . 36
predict_gaps . . . 36
predict_terms . . . 37
resample_fft_parallel . . . 37
resample_mvfft . . . 38
response_from_fit . . . 39
set_level_shift . . . 40

shift_subset . . . 40
solve_tf_parallel . . . 41
spec_from_pgram . . . 41
spec_pgram . . . 42
spec_welch . . . 43
spec_welch_tf_error . . . 43
step_distributed_lag . . . 44
step_earthtide . . . 46
step_harmonic2 . . . 48
step_lag_earthtide . . . 50
step_lag_matrix . . . 52
stretch_interp . . . 54
summarize_coef . . . 54
summarize_lm . . . 55
synthetic . . . 55
synthetic_wl . . . 56
transducer . . . 57
transfer_fun . . . 57
window_blackman . . . 58
window_blackman_harris . . . 58
window_blackman_nuttall . . . 59
window_first_deriv . . . 59
window_gaussian . . . 60
window_hamming . . . 60
window_hann . . . 61
window_nuttall . . . 61
window_rectangular . . . 62
wipp30 . . . 62

Index **64**

`waterlevel-package` *A short title line describing what the package does*

Description

A more detailed description of what the package does. A length of about one to five lines is recommended.

Details

This section should provide a more detailed overview of how to use the package, including the most important functions.

Author(s)

Your Name, email optional.

Maintainer: Your Name <your@email.com>

References

This optional section can contain literature or other references for background information.

See Also

Optional links to other man pages

Examples

```
  ## Not run:
     ## Optional simple examples of the most important functions
     ## These can be in \dontrun{} and \donttest{} blocks.

## End(Not run)
```

`add_level_shifts` *add_level_shifts*

Description

add_level_shifts

Usage

```
add_level_shifts(x, y)
```

Arguments

x	datetimes
y	shift vector

Value

vector of shifts

AIC_mlm *Computation of AIC for mlm objects*

Description

Extends the `extractAIC` method from the **stats** package to handle multi-predictand linear models (objects of class mlm). FROM paleocar

Usage

```
AIC_mlm(fit, scale = 0, k = 2)
```

Arguments

`fit`	An object of class mlm.
`scale`	The estimate of the error variance. `scale = 0` indicates that it is to be estimated by maximum likelihood.
`k`	Numeric, the penalty per parameter to be used; the default k = 2 is the classical AIC.

Value

A list of length 2 giving

- `df` The 'equivalent degrees of freedom' for the fitted model `fit`.
- `AIC` A vector of the (generalized) Akaike Information Criterion for the fits.

baro_elev *baro_elev*

Description

Calculate the effect of elevation on barometric pressure.

Usage

```
baro_elev(
  elevation = 0,
  pressure_sea = 101325,
  temperature = 20,
  gravity = 9.80665
)
```

Arguments

elevation	vector elevations (numeric)
pressure_sea	standard pressure (numeric)
temperature	temperature in celcius (numeric)
gravity	gravity (numeric)

Value

barometric pressure dependence on elevation

Examples

```
plot(baro_elev(0:10000), type='l', xlab = "elev (m)", ylab = 'pressure')
```

---

be_acworth *be_acworth_eq_4*

Description

Implementation of Acworth et. al. 2016. This is basically the transfer function value for the semi-diurnal band.

Usage

```
be_acworth(
  dat,
  wl = "wl",
  ba = "ba",
  et = "et",
  method = "spec_pgram",
  inverse = TRUE,
  ...
)
```

Arguments

dat	data that has the independent and dependent variables (data.table)
wl	name of the water level column (character).
ba	name of the barometric pressure column (character).
et	name of the Earth tide column (character).
method	either spec_pgram or spec_welch
inverse	logical whether the barometric relationship is inverse (TRUE means that when the barometric pressure goes up the measured water level goes down (vented transducer, depth to water), FALSE means that when the barometric pressure goes up so does the measured pressure (non-vented transducer)).
...	arguments to pass to transfer_acworth, spec_welch, spec_pgram

Value

barometric efficiency

References

Acworth, R. I., Halloran, L. J. S., Rau, G. C., Cuthbert, M. O., & Bernardi, T. L. (2016). An objective frequency-domain method for quantifying confined aquifer compressible storage using Earth and atmospheric tides. Geophysical Research Letters, 43 (November). doi:10.1002/2016GL071328

Acworth, R. I., G. C. Rau, L. J. S. Halloran, and W. A. Timms (2017), Vertical groundwater stor1077 age properties and changes in confinement determined using hydraulic head response to atmospheric tides, Water Resources Research, 53(4), 2983–2997, doi:10.1002/2016WR020311.

`be_acworth_eq_4` *be_acworth_eq_4*

Description

Implementation of Acworth et. al. 2016 equation 4.

Usage

```
be_acworth_eq_4(s2_gw, s2_et, s2_at, m2_gw, m2_et, d_phase, inverse = TRUE)
```

Arguments

`s2_gw`	`numeric` s2 component in the groundwater levels
`s2_et`	`numeric` s2 component in the earth tides
`s2_at`	`numeric` s2 component for atmospheric pressure
`m2_gw`	`numeric` m2 component in the groundwater levels
`m2_et`	`numeric` m2 component in the earth tides
`d_phase`	`numeric` phase difference between Earth tide and atmospheric drivers s2_et and s2_at
`inverse`	`logical` whether the barometric relationship is inverse (TRUE means that when the barometric pressure goes up the measured water level goes down (vented transducer, depth to water), FALSE means that when the barometric pressure goes up so does the measured pressure (non-vented transducer))

Value

barometric efficiency

References

Acworth, R. I., Halloran, L. J. S., Rau, G. C., Cuthbert, M. O., & Bernardi, T. L. (2016). An objective frequency-domain method for quantifying confined aquifer compressible storage using Earth and atmospheric tides. Geophysical Research Letters, 43 (November). doi:10.1002/2016GL071328

Examples

```
be_acworth_eq_4(s2_at = 7.461,
           s2_et=224.640,
           s2_gw=4.086,
           m2_gw = 0.471,
           m2_et = 492.526,
           d_phase=-56.709)
be_acworth_eq_4(s2_at = 6.164,
           s2_et=270.463,
           s2_gw=0.329,
           m2_gw = 0.225,
           m2_et = 551.572,
           d_phase=-71.726)
be_acworth_eq_4(s2_at = 5.897,
           s2_et=234.478,
           s2_gw=5.536,
           m2_gw = 0.773,
           m2_et = 558.075,
           d_phase=-70.393)
```

---

be_clark *be_clark*

Description

Clark 1967 solution for calculating barometric efficiency.

Usage

```
be_clark(
  dat,
  dep = "wl",
  ind = "baro",
  lag_space = 1,
  inverse = TRUE,
  return_model = FALSE
)
```

Arguments

dat	data that has the independent and dependent variables (data.table)
dep	name of the dependent variable column (character). This is typically the name for the column holding your water level data.
ind	name of the independent variable column (character). This is typically the name for the column holding your barometric pressure data.
lag_space	space between difference calculation in number of observations

inverse	whether the barometric relationship is inverse (TRUE means that when the barometric pressure goes up the measured water level goes down (vented transducer, depth to water), FALSE means that when the barometric pressure goes up so does the measured pressure (non-vented transducer)) (logical).
return_model	whether to return the lm model or just the barometric/loading efficiency (logical).#'

Value

lm linear model for Clark's method. The coefficient is the BE/LE

References

Clark, W., 1967. Computing the Barometric Efficiency of a well. Proc. Am. Soc. Civ. Eng. J. Hydraul. Div. V. 93, 93–98.

Examples

```
library(data.table)
datetime <- seq.POSIXt(as.POSIXct("2016-01-01 12:00:00"),
                       as.POSIXct("2016-01-05 12:00:00"), by='sec' )
baro <- sin(seq(0, 2*pi, length.out = length(datetime)))
wl   <- -0.4 * baro + rnorm(length(datetime), sd = 0.02)
dat  <- data.table(baro, wl, datetime)
be_clark(dat, dep='wl', ind='baro', lag_space=1, inverse=TRUE)
```

---

be_correct *be_correct*

Description

Adjust values based on the barometric efficiency

Usage

```
be_correct(
  dat,
  dep = "wl",
  ind = "baro",
  be = 0,
  inverse = TRUE,
  known_mean = NULL
)
```

Arguments

dat	data that has the independent and dependent variables (data.table)
dep	name of the dependent variable column (character). This is typically the name for the column holding your water level data.
ind	name of the independent variable column (character). This is typically the name for the column holding your barometric pressure data.
be	numeric value of the barometric efficiency
inverse	whether the barometric relationship is inverse (TRUE means that when the barometric pressure goes up the measured water level goes down (vented transducer, depth to water), FALSE means that when the barometric pressure goes up so does the measured pressure (non-vented transducer)) (logical).
known_mean	numeric explicitly enter the mean if known. Otherwise estimate from data.

Value

numeric vector of corrected values

Examples

```
library(data.table)
baro = rnorm(1000, sd = 0.01) + 9
wl <- baro * 0.4 + 18
dat <- data.table(baro, wl)
dat$wl + be_correct(dat, be=0.4, inverse = FALSE)
dat$wl + be_correct(dat, be=0.4, inverse = FALSE, known_mean = 9)
# should return ~21.6 = 18 + 0.4 * 9
```

---

be_high_low *be_high_low*

Description

Uses the maximum and minimum values for a time period to calculate be.

Usage

```
be_high_low(
  dat,
  dep = "wl",
  ind = "baro",
  time = "datetime",
  time_group = "%Y-%m-%d"
)
```

Arguments

dat	data that has the independent and dependent variables (data.table)
dep	name of the dependent variable column (character). This is typically the name for the column holding your water level data.
ind	name of the independent variable column (character). This is typically the name for the column holding your barometric pressure data.
time	name of the column containing the time (character)
time_group	format for grouping time (ie: "%Y-%m-%d") (character)

Value

barometric efficiency based on high and low values

Examples

```
library(data.table)
datetime <- seq.POSIXt(as.POSIXct("2016-01-01 00:00:00", tz = 'UTC'),
                       as.POSIXct("2016-01-06 00:00:00", tz = 'UTC'), by='hour')
baro  <- sin(seq(0, 2*pi, length.out = length(datetime)))
wl    <- 0.4 * baro + rnorm(length(datetime), sd = 0.01)
dat   <- data.table(baro, wl, datetime)
be_hl <- be_high_low(dat)
be_hl <- be_high_low(dat, time_group = 86400)
```

---

be_least_squares *be_least_squares*

Description

Calculate the barometric efficiency by using least squares.

Usage

```
be_least_squares(
  dat,
  dep = "wl",
  ind = "baro",
  inverse = TRUE,
  return_model = FALSE
)
```

Arguments

dat	data that has the independent and dependent variables (data.table)
dep	name of the dependent variable column (character). This is typically the name for the column holding your water level data.
ind	name of the independent variable column (character). This is typically the name for the column holding your barometric pressure data.
inverse	whether the barometric relationship is inverse (TRUE means that when the barometric pressure goes up the measured water level goes down (vented transducer, depth to water), FALSE means that when the barometric pressure goes up so does the measured pressure (non-vented transducer)) (logical).
return_model	whether to return the lm model or just the barometric/loading efficiency (logical).#'

Value

barometric efficiency calculated by least squares

Examples

```
library(data.table)
datetime <- seq.POSIXt(as.POSIXct("2016-01-01 12:00:00"),
                       as.POSIXct("2016-01-05 12:00:00"), by='hour' )
baro <- sin(seq(0, 2*pi, length.out = length(datetime)))
wl <- -0.4 * baro
dat <- data.table(baro, wl, datetime)

be_least_squares(dat)
```

---

be_least_squares_diff *be_least_squares_diff*

Description

Calculate the barometric efficiency by using the least squares with differences

Usage

```
be_least_squares_diff(
  dat,
  dep = "wl",
  ind = "baro",
  lag_space = 1,
  inverse = TRUE,
  return_model = FALSE
)
```

Arguments

dat	data that has the independent and dependent variables (data.table)
dep	name of the dependent variable column (character). This is typically the name for the column holding your water level data.
ind	name of the independent variable column (character). This is typically the name for the column holding your barometric pressure data.
lag_space	space between difference calculation in number of observations
inverse	whether the barometric relationship is inverse (TRUE means that when the barometric pressure goes up the measured water level goes down (vented transducer, depth to water), FALSE means that when the barometric pressure goes up so does the measured pressure (non-vented transducer)) (logical).
return_model	whether to return the lm model or just the barometric/loading efficiency (logical).#'

Value

barometric efficiency calculated by least squares

Examples

```
library(data.table)
datetime <- seq.POSIXt(as.POSIXct("2016-01-01 12:00:00"),
                       as.POSIXct("2016-01-05 12:00:00"), by='hour' )
baro <- sin(seq(0, 2*pi, length.out = length(datetime)))
wl <- -0.4 * baro
dat <- data.table(baro, wl, datetime)
be_least_squares_diff(dat, lag_space = 1)
```

---

be_rahi *be_rahi ## need to check for correctness*

Description

Rahi 2010 solution for calculating barometric efficiency.

Usage

```
be_rahi(dat, dep = "wl", ind = "baro", lag_space = 1, inverse = TRUE)
```

Arguments

dat	data that has the independent and dependent variables (data.table)
dep	name of the dependent variable column (character). This is typically the name for the column holding your water level data.
ind	name of the independent variable column (character). This is typically the name for the column holding your barometric pressure data.
lag_space	space between difference calculation in number of observations
inverse	whether the barometric relationship is inverse (TRUE means that when the barometric pressure goes up the measured water level goes down (vented transducer, depth to water), FALSE means that when the barometric pressure goes up so does the measured pressure (non-vented transducer)) (logical).

Value

barometric efficiency using Rahi's method

References

Rahi, K. A. (2010). Estimating the hydraulic parameters of the Arbuckle-Simpson aquifer by analysis of naturally-induced stresses (Doctoral dissertation, Oklahoma State University).

Examples

```
library(data.table)
datetime <- seq.POSIXt(as.POSIXct("2016-01-01 12:00:00"),
                       as.POSIXct("2016-01-05 12:00:00"), by='hour' )
baro <- sin(seq(0, 2 * pi, length.out = length(datetime)))
noise <- rnorm(length(datetime), sd = 0.01)
wl <- -0.4 * baro + noise
dat <- data.table(baro, wl, datetime)
be_rahi(dat, dep = 'wl', ind = 'baro', lag_space = 1)
```

---

be_ratio *be_ratio*

Description

Calculate the barometric efficiency using the mean (or other statistic) of the water level change divided by the barometric pressure change . There is the option to only include responses that are large.

Usage

```
be_ratio(
  dat,
  dep = "wl",
  ind = "baro",
  lag_space = 1,
  inverse = TRUE,
  quant = 0.9,
  stat = mean,
  ...
)
```

Arguments

dat	data that has the independent and dependent variables (data.table)
dep	name of the dependent variable column (character). This is typically the name for the column holding your water level data.
ind	name of the independent variable column (character). This is typically the name for the column holding your barometric pressure data.
lag_space	space between difference calculation in number of observations
inverse	whether the barometric relationship is inverse (TRUE means that when the barometric pressure goes up the measured water level goes down (vented transducer, depth to water), FALSE means that when the barometric pressure goes up so does the measured pressure (non-vented transducer)) (logical).
quant	quantile cutoff value that differences need to be this large (numeric)
stat	the result to return (mean, median, quantile) (function)
...	other arguments to pass to "stat"

Value

barometric efficiency calculated by using ratio method

Examples

```
library(data.table)
datetime <- seq.POSIXt(as.POSIXct("2016-01-01 12:00:00"),
                       as.POSIXct("2016-01-05 12:00:00"), by='hour' )
baro <- sin(seq(0, 2*pi, length.out = length(datetime)))
noise <- rnorm(length(datetime), sd = 0.01)
wl <- -0.4 * baro + noise
dat <- data.table(baro, wl, datetime)
be_ratio(dat, quant = 0.5, stat=median)
be_ratio(dat, quant = 0.9, stat=mean)
```

be_storage *be_storage*

Description

This function estimates the storage coefficient from physical properties and the barometric efficiency The equation used is from Batu 1998, eq 2-101, pg. 72

Usage

```
be_storage(be, n, b, gamma = 9799.74, beta = 4.786e-10)
```

Arguments

be	the barometric efficiency (numeric)
n	the porosity 0 to 1 (numeric)
b	aquifer thickness (m) (numeric)
gamma	the specific weight of water (N/m^3) (numeric)
beta	the compressibility of water (4.786e-10 m^2/N) (numeric)

Value

storage calculated from barometric efficiency

References

Batu, V. (1998). Aquifer hydraulics: a comprehensive guide to hydrogeologic data analysis. John Wiley & Sons.

Examples

```
be_storage(0.5, 0.32, 45)
```

be_visual *be_visual*

Description

Generate dataset for comparing barometric efficiency

Usage

```
be_visual(
  dat,
  dep = "wl",
  ind = "baro",
  time = "datetime",
  be_tests = seq(0, 1, 0.1),
  inverse = TRUE,
  subsample = TRUE
)
```

Arguments

dat	data that has the independent and dependent variables (data.table)
dep	name of the dependent variable column (character). This is typically the name for the column holding your water level data.
ind	name of the independent variable column (character). This is typically the name for the column holding your barometric pressure data.
time	name of the column containing the time (character)
be_tests	vector of barometric efficiencies to test (between 0 and 1) (numeric)
inverse	whether the barometric relationship is inverse (TRUE means that when the barometric pressure goes up the measured water level goes down (vented transducer, depth to water), FALSE means that when the barometric pressure goes up so does the measured pressure (non-vented transducer)) (logical).
subsample	should the data be subsampled for plotting? (logical)

Value

data.table of barometric efficiency compensated datasets

References

Smith, L. A., van der Kamp, G., & Hendry, M. J. (2013). A new technique for obtaining high-resolution pore pressure records in thick claystone aquitards and its use to determine in situ compressibility. Water Resources Research, 49(2), 732-743. doi:10.1002/wrcr.20084

Examples

```
library(data.table)
be <- 0.43
x <- seq(0, 28*pi, pi / (12*12))

baro <- sin(x) + rnorm(length(x), sd = 0.04)
wl <- -sin(x) * be + rnorm(length(x), sd = 0.04)
dat <- data.table(datetime = as.POSIXct(x * 86400 / (2 * pi),
                                        origin = '1970-01-01', tz = 'UTC'),
                  wl = wl, baro = baro)
be_visual(dat)
```

be_visual_data *be_visual_data*

Description

Generate dataset for comparing barometric efficiency

Usage

```
be_visual_data(
  dat,
  dep = "wl",
  ind = "baro",
  be_tests = seq(0, 1, 0.1),
  inverse = TRUE
)
```

Arguments

dat	data that has the independent and dependent variables (data.table)
dep	name of the dependent variable column (character). This is typically the name for the column holding your water level data.
ind	name of the independent variable column (character). This is typically the name for the column holding your barometric pressure data.
be_tests	vector of barometric efficiencies to test (between 0 and 1) (numeric)
inverse	whether the barometric relationship is inverse (TRUE means that when the barometric pressure goes up the measured water level goes down (vented transducer, depth to water), FALSE means that when the barometric pressure goes up so does the measured pressure (non-vented transducer)) (logical).

Value

data.table of barometric efficiency compensated datasets

Examples

```
library(data.table)
be <- 0.43
x <- seq(0, 28*pi, pi / (12*12))

baro <- sin(x) + rnorm(length(x), sd = 0.04)
wl <- -sin(x) * be + rnorm(length(x), sd = 0.04)
dat <- data.table(datetime = as.POSIXct(x * 86400 / (2 * pi),
                                        origin = '1970-01-01', tz = 'UTC'),
                  wl = wl, baro = baro)
be_visual_data(dat)
```

be_visual_plot *be_visual_plot*

Description

Plot to compare barometric efficiency. Large datasets may take a long time to plot. Subsample should be set to TRUE

Usage

```
be_visual_plot(dat, time = "datetime", subsample = TRUE)
```

Arguments

dat	data that has the independent and dependent variables (data.table)
time	name of the column containing the time (character)
subsample	should the data be subsampled for plotting? (logical)

Value

plotly graph for barometric efficiency estimation with Smith method

Examples

```
library(plotly)
library(data.table)
be <- 0.43
x <- seq(0, 28 * pi, pi / (12 * 12))

baro <- sin(x) + rnorm(length(x), sd = 0.04)
wl <- -sin(x) * be + rnorm(length(x), sd = 0.04)
dat <- data.table(datetime = as.POSIXct(x * 86400 / (2 * pi),
                                        origin = '1970-01-01', tz = 'UTC'),
                  wl = wl, baro = baro)
dat_be <- be_visual_data(dat)
#be_visual_plot(dat_be) #not run
```

brf_from_frf *brf_from_frf*

Description

Calculate the barometric response function from the transfer function

Usage

```
brf_from_frf(x)
```

Arguments

x	the complex transfer function vector

Value

vector of the barometric response function

---

coh_phase *coh_phase*

Description

calculate phase and coherency from the spectra and cross-spectra

Usage

```
coh_phase(pgram)
```

Arguments

pgram	numeric array must be multivariate

Value

list of coherency and phase

---

convolve_fft *convolve_fft*

Description

convolve_fft

Usage

```
convolve_fft(x, y)
```

Arguments

x	numeric vector
y	convolution kernel

Value

numeric vector result of convolution

cross_basis_fft *cross_basis_fft*

Description

cross_basis_fft

Usage

```
cross_basis_fft(basisvar, basislag)
```

Arguments

basisvar	numeric vector
basislag	lagging matrix

Value

numeric vector result of convolution

det_parallel *det_parallel*

Description

Determinant for an array in parallel

Usage

```
det_parallel(a)
```

Arguments

a	numeric array values to evaluate

Value

vector of determinants

diff_shift *diff_shift*

Description

Calculate lagged differences padded with NA values

Usage

```
diff_shift(x, lag_space = 1)
```

Arguments

x	vector to difference (numeric)
lag_space	spacing for lags, useful for higher frequency monitoring (integer).

Value

lagged differences

Examples

```
diff_shift(1:100, 2)
```

distributed_lag *distributed_lag*

Description

Depending on the vector to lag and the maximum knot, distributed_lag will either use an FFT (no NA and large maximum knot), or a parallel method (NA, or small maximum knot).

Usage

```
distributed_lag(
  x,
  knots,
  spline_fun = ns,
  lag_name = "",
  n_subset = 1,
  n_shift = 0
)
```

Arguments

x	numeric vector to lag
knots	specific knots for the lagging process
spline_fun	spline function to use i.e. splines::ns, splines::bs
lag_name	name of variable to lag
n_subset	for distributed_lag parallel
n_shift	for distributed_lag parallel

Value

matrix with distributed lag terms

---

distributed_lag_parallel
distributed_lag_parallel

Description

This method calculates the basis for a distributed lag in parallel. It is currently slow.

Usage

```
distributed_lag_parallel(x, bl, lag_max, n_subset = 1L, n_shift = 0L)
```

Arguments

x	matrix value of lag
bl	matrix the basis lags
lag_max	integer maximum number of lags
n_subset	integer subset the data
n_shift	integer to shift

Value

distributed lag basis

`energy_density_wang` *energy_density_wang*

Description

energy_density_wang

Usage

```
energy_density_wang(magnitude, distance)
```

Arguments

`magnitude`	Richter magnitude
`distance`	epicentral distance

Value

value of the energy density

Examples

```
energy_density_wang(5, 9)
```

`fftw_convolve` *fftw_convolve*

Description

Do convolution for a data series and matrix of equal length filters. For long data series this function will be much faster than non-fft methods.

Usage

```
fftw_convolve(x, y, normalize = TRUE, align = "center")
```

Arguments

`x`	`numeric` vector or single column matrix the data series
`y`	`numeric` matrix of filter(s)
`normalize`	`logical` do you want the values normalized
`align`	`character` what alignment for the convolution

Value

convolution of data series and filter

Examples

```
library(splines)

knots <- round(10^seq(0.5, 4.635484, length.out = 9))
n_lags <- 86400 * 2

bla <- ns(0:(n_lags),
  knots = knots,
  Boundary.knots = c(0, n_lags)
)
bv <- rnorm(86400 * 2 + 3600 * 24 * 4)
system.time({
  dl <- fftw_convolve(bv, bla, normalize = FALSE)
})
```

`find_gaps` *find_gaps*

Description

find_gaps

Usage

```
find_gaps(x, dep_var = "val", time_var = "datetime", time_interval = 1)
```

Arguments

`x`	the data set (data.table)
`dep_var`	the dependent variable name (character)
`time_var`	the time variable name (character)
`time_interval`	the delta t (numeric)

Value

data.table of gaps

find_level_shift *find_level_shift*

Description

find_level_shift

Usage

```
find_level_shift(x, dep_var = "val", time_var = "datetime", time_interval = 1)
```

Arguments

x	the data set (data.table)
dep_var	the dependent variable name (character)
time_var	the time variable name (character)
time_interval	the delta t (numeric)

Value

data.table of gaps

fit_gaps *fit_gaps*

Description

fit_gaps

Usage

```
fit_gaps(x, recipe, time_var = "datetime")
```

Arguments

x	the data set (data.table)
recipe	the recipe to apply
time_var	the time variable name (character)

Value

data.table of fit results

formula_from_recipe *formula_from_recipe guess formula from recipe*

Description

formula_from_recipe guess formula from recipe

Usage

```
formula_from_recipe(recipe)
```

Arguments

recipe	recipe to apply

Value

formula

frf_predict_terms *frf_predict_terms*

Description

frf_predict_terms

Usage

```
frf_predict_terms(
  frf,
  x,
  time_col = "datetime",
  vars = c("wl", "baro"),
  coherency_cut = 0.3,
  limit = 2
)
```

Arguments

frf	frequency response function
x	input variable dataset
time_col	datetime
vars	variables used in the transfer function call
coherency_cut	values must have at least this coherency
limit	gain must be lower than this value. Often useful for Earth tides.

Value

data.table of results

`gap_fill` *gap_fill*

Description

gap_fill

Usage

```
gap_fill(
  x,
  gaps,
  recipe,
  dep_var = "wl",
  time_var = "datetime",
  time_interval = 1L,
  buffer_start = 86400 * 4,
  buffer_end = 86400 * 4,
  max_interp = 86400 * 7,
  include_level_shift = TRUE
)
```

Arguments

`x`	data.table of water levels
`gaps`	data.table of gaps
`recipe`	recipe to apply
`dep_var`	the time variable name (character)
`time_var`	the time variable name (character)
`time_interval`	the delta t (numeric)
`buffer_start`	how much buffer on each side of gap
`buffer_end`	how much buffer on each side of gap
`max_interp`	largest gap to interpolate
`include_level_shift`	include level shift in adjustment

Value

data.table of predictions

gap_fill2 *gap_fill2*

Description

gap_fill2

Usage

```
gap_fill2(x, y)
```

Arguments

x	gap data.table
y	interpolated values

Value

data.table with gap values adjusted for range

get_fit_summary *get_fit_summary*

Description

get_fit_summary

Usage

```
get_fit_summary(x, recipe, start_reg, end_reg, start_gap, end_gap)
```

Arguments

x	data.table of water levels
recipe	recipe to apply
start_reg	start subset time
end_reg	end subset time
start_gap	start subset time
end_gap	end subset time

Value

data.table summary

get_intercept_stats *get_intercept_stats*

Description

get_intercept_stats

Usage

```
get_intercept_stats(x)
```

Arguments

x	data.table of level shifts from regression

Value

data.table of shifts

get_level_shift_coef *get_level_shift_coef*

Description

get_level_shift_coef

Usage

```
get_level_shift_coef(x)
```

Arguments

x	data.table of level shifts from regression

Value

data.table of shifts

get_shift *get_shift*

Description

get_shift

Usage

```
get_shift(x, recipe, start, end)
```

Arguments

x	data.table of water levels
recipe	recipe to apply
start	start subset time
end	end subset time

Value

data.table of predictions

harmonic *harmonic*

Description

calculate sin and cos curves from POSIXct times (serial)

Usage

```
harmonic(times, freq)
```

Arguments

times	numeric vector times to calculate sin and cos at
freq	numeric vector frequencies for sin and cos

Value

sin and cos curves

konno_ohmachi_parallel *konno_ohmachi_parallel*

Description

This method does konno ohmachi smoothing

Usage

```
konno_ohmachi_parallel(x, f, b = 10L)
```

Arguments

x	vector to smooth
f	vector of frequencies
b	integer even magnitude to smooth

Value

konno ohmachi smoothed results

konno_ohmachi_serial *konno_ohmachi_serial*

Description

This method does konno ohmachi smoothing

Usage

```
konno_ohmachi_serial(x, f, b = 10L)
```

Arguments

x	`numeric vector` to smooth
f	`numeric vector` of frequencies
b	`integer` smoothing coefficient

Value

vector of konno ohmachi smoothed results

lag_difference *lag_difference*

Description

Calculate differences for the dependent and independent variables and remove any NA values. This modifies the data.table in place.

Usage

```
lag_difference(
  dat,
  var_name = "wl",
  lag_space = 1,
  inverse = FALSE,
  remove_na = TRUE
)
```

Arguments

dat	data that has the independent and dependent variables (data.table)
var_name	name of the column to lag (character)
lag_space	space between difference calculation in number of observations
inverse	whether the barometric relationship is inverse (TRUE means that when the barometric pressure goes up the measured water level goes down (vented transducer, depth to water), FALSE means that when the barometric pressure goes up so does the measured pressure (non-vented transducer)) (logical).
remove_na	remove NA values (logical)

Value

data.table with lagged differences

Examples

```
library(data.table)
datetime <- seq.POSIXt(as.POSIXct("2016-01-01 12:00:00"),
                       as.POSIXct("2016-01-05 12:00:00"), by='hour' )
baro <- sin(seq(0, 2*pi, length.out = length(datetime)))
wl <- 0.4 * baro
dat <- data.table(baro, wl, datetime)
lag_difference(dat, lag_space = 1, remove_na = FALSE)
```

`lag_matrix` *lag_matrix*

Description

lag data and subset the results

Usage

```
lag_matrix(x, lags, n_subset = 1L, n_shift = 0L, var_name = "lag")
```

Arguments

`x`	`numeric vector` to lag
`lags`	`integer vector` with the lags
`n_subset`	`integer` subset every n_subset values
`n_shift`	`integer` amount to shift results
`var_name`	`character` name for the generated matrix columns

Value

matrix with lagged values

`level_shift_to_datetime` *level_shift_to_datetime*

Description

level_shift_to_datetime

Usage

```
level_shift_to_datetime(x)
```

Arguments

`x`	character string that includes the date

Value

datetime

log_lags *log_lags*

Description

Generate logarithmically spacing for lags. Note: the number of lags will not exactly equal n unless max_time_lag is large or n is very small

Usage

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

Arguments

n	number of lags (integer)
max_time_lag	maximum lag (integer)

Value

integer vector of lags

Examples

```
log_lags(12, 86401)
```

pad_kernel *pad_kernel*

Description

Pad a smoothing kernel

Usage

```
pad_kernel(k, n)
```

Arguments

k	tskernel smoothing "tskernel" object.
n	integer the total length

Value

vector of the padded kernel

`portion` *Extract Finalized Training Set*

Description

As steps are estimated by ‘prep‘, these operations are applied to the training set. Rather than running ‘bake‘ to duplicate this processing, this function will return variables from the processed training set.

Usage

```
portion(object, ...)
```

Arguments

`object`	A ‘recipe‘ object that has been prepared with the option ‘retain = TRUE‘.
`...`	One or more selector functions to choose which variables will be returned by the function. See `selections()` for more details. If no selectors are given, the default is to use `dplyr::everything()`.

Details

When preparing a recipe, if the training data set is retained using ‘retain = TRUE‘, there is no need to ‘bake‘ the recipe to get the preprocessed training set.

‘portion‘ will return the results of a recipes where _all steps_ have been applied to the data, irrespective of the value of the step’s ‘skip‘ argument.

See Also

[recipe()] [prep.recipe()] [bake.recipe()] [juice.recipe()]

`predict_gaps` *predict_gaps*

Description

predict_gaps

Usage

```
predict_gaps(x, fits, term_labels = NULL)
```

Arguments

x	the data set (data.table)
fits	data.table of fits
term_labels	names of the group (character)

Value

data.table of predictions

---

predict_terms *predict_terms*

Description

predict by term for 'mlm' or 'lm' object

Usage

```
predict_terms(object, ...)
```

Arguments

object	model fit from lm object
...	currently not used

Value

return matrix of predictions

---

resample_fft_parallel *Resample an fft using varying numbers of sine tapers*

Description

Produce an un-normalized psd based on an fft and a vector of optimal sine tapers

Usage

```
resample_fft_parallel(
  fftz,
  tapers,
  verbose = TRUE,
  dbl = TRUE,
  tapcap = 10000L
)
```

Arguments

fftz	complex; a matrix representing the dual-length fft; see also the dbl argument
tapers	integer; a vector of tapers
verbose	logical; should messages be given?
dbl	logical; should the code assume fftz is dual-length or single-length?
tapcap	integer; the maximum number of tapers which can be applied; note that the length is automatically limited by the length of the series.

Details

To produce a psd estimate with our adaptive spectrum estimation method, we need only make one fft calculation initially and then apply the weighting factors given by parabolic_weights_rcpp, which this function does.

See Also

riedsid

Examples

```
fftz <- complex(real=1:8, imaginary = 1:8)
taps <- 1:4
try(resample_fft_rcpp2(fftz, taps))
```

---

resample_mvfft *Resample an fft using varying numbers of sine tapers*

Description

Produce an un-normalized psd based on an fft and a vector of optimal sine tapers

Usage

```
resample_mvfft(fftz, tapers, verbose = TRUE, dbl = TRUE, tapcap = 10000L)
```

Arguments

fftz	complex; a matrix representing the dual-length fft; see also the dbl argument
tapers	integer; a vector of tapers
verbose	logical; should messages be given?
dbl	logical; should the code assume fftz is dual-length or single-length?
tapcap	integer; the maximum number of tapers which can be applied; note that the length is automatically limited by the length of the series.

Details

To produce a psd estimate with our adaptive spectrum estimation method, we need only make one fft calculation initially and then apply the weighting factors given by parabolic_weights_field, which this function does.

See Also

riedsid

Examples

```
fftz <- complex(real=1:8, imaginary = 1:8)
taps <- 1:4
try(resample_fft_rcpp2(fftz, taps))
```

---

response_from_fit *response_from_fit*

Description

response_from_fit

Usage

```
response_from_fit(x, ...)
```

Arguments

x	model fit object, commonly lm
...	additional arguments

Value

tibble of model fit

set_level_shift *set_level_shift*

Description

Create a factor column with the data subseted according to break points.

Usage

```
set_level_shift(x, level_breaks = NULL)
```

Arguments

x	vector of numeric
level_breaks	break points in the data

Value

vector of type factor

Examples

```
set_level_shift(1:100, c(2, 40, 86))
```

shift_subset *shift_subset*

Description

lag data and subset the results

Usage

```
shift_subset(x, lag = 0L, n_subset = 1L, n_shift = 0L)
```

Arguments

x	numeric vector to lag
lag	integer vector with the lags
n_subset	integer subset every n_subset values
n_shift	integer amount to shift results

Value

vector with lagged values

`solve_tf_parallel` *solve_tf_parallel*

Description

Calculate the transfer functions for an array in parallel

Usage

```
solve_tf_parallel(a)
```

Arguments

`a`	`numeric array` values to evaluate

Value

vector of transfer functions

`spec_from_pgram` *spec_from_pgram*

Description

spec_from_pgram

Usage

```
spec_from_pgram(pgram, method = "spec_pgram", ...)
```

Arguments

`pgram`	the periodogram
`method`	either spec_pgram or spec_welch
`...`	additional arguments

Value

spectrum from the pgram

spec_pgram *spec_pgram*

Description

calculate the spectra and cross-spectra

Usage

```
spec_pgram(
  x,
  spans = NULL,
  kernel = NULL,
  taper = 0.1,
  pad = 0,
  fast = TRUE,
  demean = FALSE,
  detrend = TRUE,
  plot = TRUE,
  na.action = na.fail,
  ...
)
```

Arguments

x	univariate or multivariate time series.
spans	vector of odd integers giving the widths of modified Daniell smoothers to be used to smooth the periodogram.
kernel	alternatively, a kernel smoother of class "tskernel".
taper	specifies the proportion of data to taper. A split cosine bell taper is applied to this proportion of the data at the beginning and end of the series.
pad	proportion of data to pad. Zeros are added to the end of the series to increase its length by the proportion pad.
fast	logical; if TRUE, pad the series to a highly composite length.
demean	logical. If TRUE, subtract the mean of the series.
detrend	logical. If TRUE, remove a linear trend from the series. This will also remove the mean.
plot	plot the periodogram?
na.action	NA action function.
...	graphical arguments passed to plot.spec.

Value

array of spectra

spec_welch *spec_welch*

Description

calculate the spectra and cross-spectra. The scaling of individual series requires a closer look.

Usage

```
spec_welch(
  x,
  n_subsets = 10,
  overlap = 0.5,
  window = window_hann,
  demean = TRUE,
  detrend = TRUE,
  ...
)
```

Arguments

x	numeric matrix univariate or multivariate time series
n_subsets	number of subsets (integer)
overlap	amount of overlap 0.0 < overlap < 1.0 (numeric)
window	window function to apply (function). The options are window_hann, window_hamming, window_blackman, window_nuttall, window_blackman_nuttall, window_blackman_harris
demean	should the mean be removed from x prior to calculation (logical)
detrend	should the x be detrended to calculation (logical)
...	not used

Value

array of spectra

spec_welch_tf_error *spec_welch_tf_error*

Description

spec_welch_tf_error

Usage

```
spec_welch_tf_error(coh, amp, overlap = 0.5, n_subsets = 10, return = "amp")
```

Arguments

coh	coherency
amp	amplitude
overlap	overlap fraction from spec_welch
n_subsets	number of subsets in spec welch
return	either 'amp', or 'phase'

Value

error

---

step_distributed_lag *Distributed lag response*

Description

'step_distributed_lag' creates a *specification* of a recipe step that are distributed lag versions of a particular variable. Uses FFT for fast calculation with a large maximum lag and many observations

Usage

```
step_distributed_lag(
  recipe,
  ...,
  role = "distributed_lag",
  trained = FALSE,
  knots = 1,
  spline_fun = splines::ns,
  n_subset = 1,
  n_shift = 0,
  prefix = "distributed_lag_",
  default = NA,
  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 which variables are affected by the step. See [selections()] for more details. For the 'tidy' method, these are not currently used.
role	Defaults to "distributed_lag"

trained	A logical to indicate if the quantities for preprocessing have been estimated.
knots	specific knots for the lagging process
spline_fun	spline function to use i.e. splines::ns, splines::bs
n_subset	for distributed_lag parallel
n_shift	for distributed_lag parallel
prefix	A prefix for generated column names, default to "distributed_lag_".
default	Passed to dplyr::lag(), determines what fills empty rows left by lagging (defaults to NA).
columns	A character string of the selected variable names. This field is a placeholder and will be populated once prep() is used.
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

'step_distributed_lag' calculates the earthtide response and then lags (leads) the terms.

Value

An updated version of 'recipe' with the new step added to the sequence of existing steps (if any). For the 'tidy' method, a tibble with columns 'terms' which is the columns that will be affected and 'holiday'.

See Also

[step_lag_matrix()] [recipe()] [prep.recipe()] [bake.recipe()]

Examples

```
data(transducer)

rec <- recipe(wl ~ .,
  data = transducer[1:1000, list(datetime, wl, baro)]
)

with_et <- rec %>%
  step_distributed_lag(baro, knots = c(0, 10, 100)) %>%
  step_naomit(everything()) %>%
  prep() %>%
  juice()
```

step_earthtide *Lagged Earth tide response*

Description

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

Usage

```
step_earthtide(
  recipe,
  ...,
  role = "earthtide",
  trained = 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_",
  default = NA,
  columns = NULL,
  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.

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	How often to update astro parameters in number of samples. This speeds up code but may make it slightly less accurate.
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_".
default	Passed to dplyr::lag(), determines what fills empty rows left by lagging (defaults to NA).
columns	A character string of the selected variable names. This field is a placeholder and will be populated once prep() is used.
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

'step_earthtide' calculates the Earth tide harmonics

Value

An updated version of ‘recipe‘ with the new step added to the sequence of existing steps (if any). For the ‘tidy‘ method, a tibble with columns ‘terms‘ which is the columns that will be affected and ‘holiday‘.

See Also

[step_earthtide()] [recipe()] [prep.recipe()] [bake.recipe()]

Examples

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

rec <- recipe(wl ~ .,
              data = transducer[1:1000, list(datetime, wl)])

wg <- na.omit(eterna_wavegroups[eterna_wavegroups$time == '1 month',])
with_et <- rec %>%
  step_earthtide(datetime,
                 latitude = 34,
                 longitude = -118.5,
                 wave_groups = wg) %>%
  prep() %>%
  juice()
```

---

step_harmonic2 *Generate harmonics*

Description

‘step_harmonic2‘ creates a *specification* of a recipe step for harmonics (sin and cos terms).

Usage

```
step_harmonic2(
  recipe,
  ...,
  role = "harmonic",
  trained = FALSE,
  freq = 1,
  prefix = "harmonic_",
  default = NA,
  columns = NULL,
  skip = FALSE,
  id = rand_id("harmonic")
)
```

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 "harmonic"
trained	A logical to indicate if the quantities for preprocessing have been estimated.
freq	numeric vector frequencies for sin and cos
prefix	A prefix for generated column names, default to "harmonic_".
default	Passed to dplyr::lag(), determines what fills empty rows left by lagging (defaults to NA).
columns	A character string of the selected variable names. This field is a placeholder and will be populated once prep() is used.
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

'step_harmonic2' calculates the Earth tide harmonics

Value

An updated version of 'recipe' with the new step added to the sequence of existing steps (if any). For the 'tidy' method, a tibble with columns 'terms' which is the columns that will be affected and 'holiday'.

See Also

[step_earthtide()] [recipe()] [prep.recipe()] [bake.recipe()]

Examples

```
data(transducer)

rec <- recipe(wl ~ .,
              data = transducer[1:1000, list(datetime, wl)])

with_et <- rec %>%
  step_harmonic2(datetime, freq = c(1, 1.93, 2)) %>%
  prep() %>%
  juice()
```

step_lag_earthtide *Lagged Earth tide response*

Description

‘step_lag_earthtide‘ creates a *specification* of a recipe step that are lagged versions of the Earth tide response at a particular location. Requires the *earthtide* package

Usage

```
step_lag_earthtide(
  recipe,
  ...,
  role = "lag_earthtide",
  trained = FALSE,
  method = "gravity",
  lag = 0,
  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,
  prefix = "lag_earthtide_",
  default = NA,
  columns = NULL,
  skip = FALSE,
  id = rand_id("lag_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 "lag_earthtide"
trained	A logical to indicate if the quantities for preprocessing have been estimated.

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.
lag	A vector of positive integers. Each specified column will be lagged for each value in the vector.
astro_update	How often to update astro parameters in number of samples. This speeds up code but may make it slightly less accurate.
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
prefix	A prefix for generated column names, default to "lag_earthtide_".
default	Passed to `dplyr::lag()`, determines what fills empty rows left by lagging (defaults to NA).
columns	A character string of the selected variable names. This field is a placeholder and will be populated once `prep()` is used.
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

'step_lag_earthtide' calculates the earthtide response and then lags (leads) the terms.

Value

An updated version of ‘recipe‘ with the new step added to the sequence of existing steps (if any). For the ‘tidy‘ method, a tibble with columns ‘terms‘ which is the columns that will be affected and ‘holiday‘.

See Also

[step_lag_earthtide()] [recipe()] [prep.recipe()] [bake.recipe()]

Examples

```
data(transducer)

rec <- recipe(wl ~ .,
              data = transducer[1:1000, list(datetime, wl)])

with_et <- rec %>%
  step_lag_earthtide(datetime, latitude = 34, longitude = -118.5, lag = -1:1) %>%
  prep() %>%
  juice()
```

`step_lag_matrix` *Create a lagged (or lead) predictor*

Description

‘step_lag_matrix‘ creates a *specification* of a recipe step that will add new columns of lagged data. Lagged data will by default include NA values where the lag was induced. These can be removed with [step_naomit()], or you may specify an alternative filler value with the ‘default‘ argument. This method is faster than [step_lag()] and allows for negative values.

Usage

```
step_lag_matrix(
  recipe,
  ...,
  role = "lag_matrix",
  trained = FALSE,
  lag = 1,
  n_subset = 1,
  n_shift = 0,
  prefix = "lag_matrix_",
  default = NA,
  columns = NULL,
  skip = FALSE,
  id = rand_id("lag_matrix")
)
```

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.
role	Defaults to "predictor"
trained	A logical to indicate if the quantities for preprocessing have been estimated.
lag	A vector of integers. They can be positive, negative or zero. Each specified column will be lagged for each value in the vector.
n_subset	subset every n_subset values
n_shift	shift the data n_shift values
prefix	A prefix for generated column names, default to "lag_".
default	Passed to dplyr::lag(), determines what fills empty rows left by lagging (defaults to NA).
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.recipe()]? While all operations are baked when [prep.recipe()] 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

The step assumes that the data are already _in the proper sequential order_ for lagging.

Value

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

See Also

[recipe()] [step_lag()] [prep.recipe()] [bake.recipe()] [step_naomit()]

Examples

```
data(transducer)

rec <- recipe(wl ~ .,
              data = transducer[1:1000, list(datetime, wl, baro)])

with_et <- rec %>%
  step_lag_matrix(baro, lag = -1:1) %>%
  step_naomit(everything()) %>%
  prep() %>%
  juice()
```

stretch_interp *stretch_interp*

Description

stretch_interp

Usage

```
stretch_interp(start_val = NA, end_val = NA, values)
```

Arguments

start_val	starting value to match
end_val	end value to match
values	set of values

Value

shifted vector of values

summarize_coef *summarize_coef*

Description

summarize_coef

Usage

```
summarize_coef(fit)
```

Arguments

fit	model fit from lm object

Value

data.table with coefficient results

summarize_lm *summarize_lm*

Description

summarize_lm

Usage

```
summarize_lm(fit, ...)
```

Arguments

fit	model fit from lm object
...	additional arguments

Value

data.table with fit results

synthetic *synthetic*

Description

This function is used for testing purposes

Usage

```
synthetic(
  sd_noise = 2e-04,
  sd_noise_trend = 3e-05,
  linear_trend = 1e-07,
  n = 14 * 86400,
  seed = NULL,
  scale = 0.5,
  baro_kernel = NULL
)
```

Arguments

sd_noise	standard deviation of random noise to add (numeric)
sd_noise_trend	standard deviation of noise to add to generate a trend (numeric)
linear_trend	magnitude of linear trend in time (numeric)
n	length of time series in seconds (integer)
seed	random number seed for reproducibility (numeric)
scale	multiplier for barometric pressure (numeric)
baro_kernel	vector values to convolve with barometric pressure

Value

data.table of synthetic water levels and barometric pressure

---

synthetic_wl *synthetic_wl*

Description

This function is used for testing purposes

Usage

```
synthetic_wl(
  baro,
  datetime,
  sd_noise = 0,
  linear_trend = 0,
  intercept = 0,
  seed = NULL,
  kernel = NULL
)
```

Arguments

baro	barometric pressure (numeric vector)
datetime	POSIXct dates
sd_noise	standard deviation of random noise to add (numeric)
linear_trend	magnitude of linear trend in time (numeric)
intercept	
seed	random number seed for reproducibility (numeric)
kernel	

Value

data.table of synthetic water levels and barometric pressure

transducer *transducer*

Description

This data.frame contains the water levels, barometric pressure, and synthtetic earthtides from 2016-08-25 to 2016-10-15 (subsampled to every 2 minutes).

Usage

```
transducer
```

Format

A `data.frame` The columns are:

`datetime` POSIXct date and time

`wl` transducer pressure of water column(dbar)

`baro` barometric pressure (dbar)

`et` synthetic gravity

Examples

```
utils::data(transducer)
```

transfer_fun *transfer_fun*

Description

transfer_fun

Usage

```
transfer_fun(dat, vars, time = "datetime", method = "spec_pgram", ...)
```

Arguments

`dat`	data that has the independent and dependent variables (data.table)
`vars`	variables to include for transfer function calculation
`time`	the name of the column that contains the POSIXct date and time
`method`	either spec_pgram, spec_welch, or spec_multitaper
`...`	arguments to pass to method

Value

transfer function, gain and phase, coherency

`window_blackman` *window_blackman*

Description

Blackman window for FFT

Usage

```
window_blackman(n)
```

Arguments

`n`	length of signal (integer)

Value

window

Examples

```
window_blackman(100)
```

`window_blackman_harris` *window_blackman_harris*

Description

Blackman-Harris window for FFT

Usage

```
window_blackman_harris(n)
```

Arguments

`n`	length of signal (integer)

Value

window

Examples

```
window_blackman_harris(100)
```

`window_blackman_nuttall` *window_blackman_nuttall*

Description

Blackman-Nuttall window for FFT

Usage

```
window_blackman_nuttall(n)
```

Arguments

`n`	length of signal (integer)

Value

window

Examples

```
window_blackman_nuttall(100)
```

`window_first_deriv` *window_first_deriv*

Description

First derivative window for FFT

Usage

```
window_first_deriv(n, a0, a1, a2, a3)
```

Arguments

`n`	length of signal (integer)
`a0`	`numeric` coefficient
`a1`	`numeric` coefficient
`a2`	`numeric` coefficient
`a3`	`numeric` coefficient

Value

window

Examples

```
# nuttall window
window_first_deriv(100, 0.355768, 0.487396, 0.144232, 0.012604)
```

`window_gaussian` *window_gaussian*

Description

Gaussian window for FFT https://en.wikipedia.org/wiki/Window_function

Usage

```
window_gaussian(n, sigma)
```

Arguments

`n`	length of signal (integer)
`sigma`	the standard deviation in periods (numeric)

Value

window

Examples

```
window_gaussian(100, 0.3)
```

`window_hamming` *window_hamming*

Description

Hamming window for FFT

Usage

```
window_hamming(n)
```

Arguments

`n`	length of signal (integer)

Value

window

Examples

```
window_hamming(100)
```

`window_hann` *window_hann*

Description

Hann window for FFT

Usage

```
window_hann(n)
```

Arguments

n	length of signal (integer)

Value

window

Examples

```
window_hann(100)
```

`window_nuttall` *window_nuttall*

Description

Nuttall window for FFT

Usage

```
window_nuttall(n)
```

Arguments

n	length of signal (integer)

Value

window

Examples

```
window_nuttall(100)
```

`window_rectangular` *window_rectangular*

Description

Rectangular window for FFT

Usage

```
window_rectangular(n)
```

Arguments

n	length of signal (integer)

Value

window

Examples

```
window_rectangular(100)
```

`wipp30` *wipp30*

Description

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

Usage

```
wipp30
```

Format

A `data.frame` The columns are:

`time` time in hours

`wl` transducer pressure of water column

`baro` barometric pressure

`et` synthetic gravity

Examples

```
utils::data(wipp30)
```

Index

∗ **datagen**
    step_distributed_lag, 44
    step_earthtide, 46
    step_harmonic2, 48
    step_lag_earthtide, 50
∗ **datasets**
    transducer, 57
    wipp30, 62
∗ **generate Earth tide harmonics**
    step_earthtide, 46
∗ **generate a distributed lag response**
    step_distributed_lag, 44
∗ **generate a lagged form of the Earth tide response**
    step_lag_earthtide, 50
∗ **generate harmonics of frequency in cycles per day**
    step_harmonic2, 48
∗ **package**
    waterlevel-package, 3

add_level_shifts, 4
AIC_mlm, 5

bake(), *45*, *47*, *49*, *51*
baro_elev, 5
be_acworth, 6
be_acworth_eq_4, 7
be_clark, 8
be_correct, 9
be_high_low, 10
be_least_squares, 11
be_least_squares_diff, 12
be_rahi, 13
be_ratio, 14
be_storage, 16
be_visual, 16
be_visual_data, 18
be_visual_plot, 19
brf_from_frf, 19

coh_phase, 20
convolve_fft, 20
cross_basis_fft, 21

det_parallel, 21
diff_shift, 22
distributed_lag, 22
distributed_lag_parallel, 23
dplyr::everything(), *36*
dplyr::lag(), *45*, *47*, *49*, *51*, *53*

energy_density_wang, 24

fft, *38*
fftw_convolve, 24
find_gaps, 25
find_level_shift, 26
fit_gaps, 26
formula_from_recipe, 27
frf_predict_terms, 27

gap_fill, 28
gap_fill2, 29
get_fit_summary, 29
get_intercept_stats, 30
get_level_shift_coef, 30
get_shift, 31

harmonic, 31

konno_ohmachi_parallel, 32
konno_ohmachi_serial, 32

lag_difference, 33
lag_matrix, 34
level_shift_to_datetime, 34
log_lags, 35

pad_kernel, 35
parabolic_weights_field, *39*
parabolic_weights_rcpp, *38*

portion, 36
predict_gaps, 36
predict_terms, 37
prep(), *45*, *47*, *49*, *51*

resample_fft_parallel, 37
resample_mvfft, 38
response_from_fit, 39
riedsid, *38*, *39*

selections(), *36*
set_level_shift, 40
shift_subset, 40
solve_tf_parallel, 41
spec_from_pgram, 41
spec_pgram, 42
spec_welch, 43
spec_welch_tf_error, 43
step_distributed_lag, 44
step_earthtide, 46
step_harmonic2, 48
step_lag_earthtide, 50
step_lag_matrix, 52
stretch_interp, 54
summarize_coef, 54
summarize_lm, 55
synthetic, 55
synthetic_wl, 56

transducer, 57
transfer_fun, 57

waterlevel *(*waterlevel-package*)*, 3
waterlevel-package, 3
window_blackman, 58
window_blackman_harris, 58
window_blackman_nuttall, 59
window_first_deriv, 59
window_gaussian, 60
window_hamming, 60
window_hann, 61
window_nuttall, 61
window_rectangular, 62
wipp30, 62