Package: hydrorecipes (via r-universe)

May 23, 2026

Type Package

Title Hydrogeology steps

Version 0.0.6

Author Jonathan Kennel

Maintainer Jonathan Kennel <jkennel@uoguelph.ca>

Description This package is an implementation of some common steps of the `recipes` package using `R6` classes. It also provides some additional steps that may be useful for the geosciences and signal processing. Two goals of this package are to provide a higher performance package (memory and computation time), and as a learning experience.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 3.5.0), Bessel

Imports R6 (>= 2.3.0), Rcpp, data.table, earthtide, collapse, gslnls, plotly

LinkingTo Rcpp, RcppArmadillo, RcppThread, RcppEigen, splines2, BH

Suggests rmarkdown, tinytest, RcppRoll, recipes, tibble, knitr, bench, scales, ggplot2, splines2

SystemRequirements fftw3 (libfftw3-dev (deb), or fftw-devel (rpm)), C++17

RoxygenNote 7.3.2

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

VignetteBuilder knitr

Config/pak/sysreqs cmake libfftw3-dev libgmp3-dev make libgsl0-dev libicu-dev libuv1-dev libmpfr-dev libssl-dev

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

Date/Publication 2025-04-24 19:35:14 UTC

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

RemoteRef HEAD

RemoteSha 9ba076606cce51daddff7d24c7f7d7988bf5f29b

Contents

`areal_rojstaczer_semiconfined`
areal_rojstaczer_semiconfined

Description

areal_rojstaczer_semiconfined

Usage

```
areal_rojstaczer_semiconfined(
  frequency,
  radius_well,
  transmissivity,
  storage_confining,
  storage_aquifer,
  diffusivity_confining,
  diffusivity_vadose,
  thickness_confining,
  thickness_vadose,
  loading_efficiency,
  attenuation
)
```

Arguments

`frequency`	the frequency of the response
`radius_well`	well radius
`transmissivity`	aquifer transmissivity (L*L/t)
`storage_confining`	confining layer storativity (L/L)
`storage_aquifer`	aquifer storativity (L/L)
`diffusivity_confining`	confining layer diffusivity
`diffusivity_vadose`	air diffusivity of vadose zone
`thickness_confining`	confining layer thickness
`thickness_vadose`	vadose thickness
`loading_efficiency`	the loading efficiency of the aquifer
`attenuation`	an attenuation factor

Value

complex response vector in frequency domain

`areal_rojstaczer_unconfined` *areal_rojstaczer_unconfined*

Description

areal_rojstaczer_unconfined

Usage

```
areal_rojstaczer_unconfined(
  frequency,
  radius_well,
  storage_aquifer,
  specific_yield,
  k_vertical,
  diffusivity_vertical,
  diffusivity_vadose,
  thickness_saturated_well,
  thickness_vadose,
```

```
  thickness_aquifer,
  loading_efficiency,
  attenuation
)
```

Arguments

`frequency`	the frequency of the response
`radius_well`	well radius
`storage_aquifer`	aquifer storativity (L/L)
`specific_yield`	specific yield of unconfined aquifer
`k_vertical`	vertical hydraulic conductivity of vadose zone
`diffusivity_vertical`	unconfined layer diffusivity
`diffusivity_vadose`	air diffusivity of vadose zone
`thickness_saturated_well`	saturated well thickness
`thickness_vadose`	vadose thickness
`thickness_aquifer`	aquifer thickness
`loading_efficiency`	the loading efficiency of the aquifer
`attenuation`	an attenuation factor

Value

complex response vector in frequency domain

`bake` *bake*

Description

Evaluate the steps and store the recipe results

Usage

```
bake(.rec, data = NULL)
```

Arguments

.rec	the R6 recipe object.
data	an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which lm is called.

Value

an updated recipe

Examples

```
rec <- recipe(y~x, data = list(x = rnorm(10), y = rnorm(10))) |>
       step_scale(x) |>
       prep() |>
       bake()
```

---

be_clark_cpp *be_clark_cpp*

Description

Clark 1967 solution for calculating barometric efficiency (Algorithm from Batu 1998, pg 76)

Usage

```
be_clark_cpp(dep, ind, lag_space, inverse)
```

Arguments

dep	numeric vector of the dependent variable (ie:water level)
ind	numeric vector of the independent variable (ie:barometric pressure)
lag_space	integer spacing for lags, useful for higher frequency monitoring
inverse	logical whether the barometric relationship is inverse

Value

barometric efficiency using Clark's method

Examples

```
n <- 1000
baro <- sin(seq(0, 2 * pi, length.out = 1000))
wl <- -0.4 * baro + rnorm(1000, sd = 0.02)
be_clark_cpp(wl, 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

`be`	`numeric` value of the barometric efficiency
`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_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

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
```

bessel_k_cplx *bessel_k_cplx*

Description

Modified Bessel function of first kind order 1

Usage

```
bessel_k_cplx(x, nu, expon_scaled, n_seq)
```

Arguments

x	numeric value to evaluate
nu	numeric value to evaluate
expon_scaled	boolean value to evaluate
n_seq	nseq value to evaluate

Value

bessel function result

---

bouwer *bouwer*

Description

Keeth elementary dataset 1997-08-14 (DATA sheet Slug_Bouwer-Rice.xls) https://pubs.usgs.gov/of/2002/ofr02197
Rice.xls

Usage

```
data(bouwer)
```

Format

data.table

Examples

```
data(bouwer)
```

`bouwer_abc` *bouwer_abc*

Description

a, b and c values for Bouwer and Rice, 1976 solution, for testing https://pubs.usgs.gov/of/2002/ofr02197/spreadshee
Rice.xls

Usage

```
data(bouwer_abc)
```

Format

data.table

Examples

```
data(bouwer_abc)
```

`bouwer_rice` *Calculate transmissivity with Bouwer-Rice solution*

Description

Calculate transmissivity with Bouwer-Rice solution

Usage

```
bouwer_rice(time, drawdown, radius_screen, radius_casing, Le, Lw, H)
```

Arguments

`time`	the elapsed time
`drawdown`	the drawdown
`radius_screen`	radius of the screen
`radius_casing`	radius of the casing where the water level is
`Le`	Effecive screen length
`Lw`	height of water from bottom of well
`H`	height from bottom of aquifer

Value

transmissivity from bouwer_rice

`bouwer_rice_abc` *Calculate equations 4 and 5 from bouwer, 1989*

Description

Calculate equations 4 and 5 from bouwer, 1989

Usage

```
bouwer_rice_abc(rw, Le, Lw, H)
```

Arguments

`rw`	radius of well
`Le`	Effecive screen length
`Lw`	height of water from bottom of well
`H`	height from bottom of aquifer

Value

ln(Re/rw)

`convert_for_rojstaczer` *convert_for_rojstaczer*

Description

convert_for_rojstaczer

Usage

```
convert_for_rojstaczer(tf)
```

Arguments

`tf`	the transfer function

Value

conjugate of tf

`convert_le_to_be` *convert_le_to_be*

Description

convert_le_to_be

Usage

```
convert_le_to_be(tf)
```

Arguments

`tf`	complex transfer function

Value

the loading efficiency

`convolve_filter` *convolve_filter*

Description

convolution of vector with matrix

Usage

```
convolve_filter(x, y, remove_partial, reverse)
```

Arguments

`x`	vector to convolve with y (numeric vector)
`y`	numeric matrix to convolve with x (column by column convolution) (numeric matrix)
`remove_partial`	keep the end values or fill with NA (boolean)
`reverse`	should x be reversed before convolution (boolean)

Value

numeric matrix of convolved values

Examples

```
a <- convolve_filter(x = 1:100,
                     y = c(1:10, rep(0, 90)),
                     remove_partial = FALSE,
                     reverse = TRUE)

b <- stats::convolve(1:100, rev(1:10), type = 'filter')
```

---

convolve_matrix *convolve_matrix*

Description

convolution of vector with matrix

Usage

```
convolve_matrix(x, y, remove_partial, reverse)
```

Arguments

x	vector to convolve with y (numeric vector)
y	numeric matrix to convolve with x (column by column convolution) (numeric matrix)
remove_partial	keep the end values or fill with NA (boolean)
reverse	should x be reversed before convolution (boolean)

Value

numeric matrix of convolved values

Examples

```
a <- convolve_matrix(x = 1:100,
                     y = as.matrix(1:10),
                     remove_partial = FALSE,
                     reverse = TRUE)

b <- stats::convolve(1:100, rev(1:10), type = 'filter')
```

`distributed_lag_list` *distributed_lag_list*

Description

Create distributed lag terms

Usage

```
distributed_lag_list(
  x,
  n_lag,
  lag_max,
  df,
  degree,
  internal_knots,
  boundary_knots,
  complete_basis,
  periodic,
  derivs,
  integral
)
```

Arguments

`x`	numeric vector to lag
`n_lag`	number of lag terms
`lag_max`	integer the maximum lag
`df`	Degree of freedom that equals to the column number of the returned matrix. One can specify `df` rather than `knots`, then the function chooses `df - degree - as.integer(intercept)` internal knots at suitable quantiles of `x` ignoring missing values and those `x` outside of the boundary. For periodic splines, `df - as.integer(intercept)` internal knots will be chosen at suitable quantiles of `x` relative to the beginning of the cyclic intervals they belong to (see Examples) and the number of internal knots must be greater or equal to the specified `degree - 1`. If internal knots are specified via `knots`, the specified `df` will be ignored.
`degree`	A nonnegative integer specifying the degree of the piecewise polynomial. The default value is 3 for cubic splines. Zero degree is allowed for piecewise constant basis functions.
`internal_knots`	location of internal knots
`boundary_knots`	location of boundary knots
`complete_basis`	logical intercept?

periodic — A logical value. If TRUE, the periodic splines will be returned. The default value is FALSE.

derivs — A nonnegative integer specifying the order of derivatives of splines basis function. The default value is 0.

integral — A logical value. If TRUE, the corresponding integrals of spline basis functions will be returned. The default value is FALSE. For periodic splines, the integral of each basis is integrated from the left boundary knot.

Value

List of distributed lags

---

get_formula_vars *get_formula_vars*

Description

get_formula_vars

Usage

```
get_formula_vars(formula, data)
```

Arguments

formula — an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under 'Details'.

data — an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which lm is called.

Value

character vector of variable names

`grf_grid` *grf_grid*

Description

Parallel convolution of GRF well function and flow rates in the time domain. Time series needs to be regularily spaced and so are the flow rates. Some performance gains can be achieved if the number of flow rate does not change for each time.

Usage

```
grf_grid(
  grid,
  well_locations,
  flow_rate,
  time,
  specific_storage,
  hydraulic_conductivity,
  thickness,
  flow_dimension
)
```

Arguments

`grid`	locations of grid points (x,y)
`well_locations`	locations of wells (x,y)
`flow_rate`	well flow rates
`time`	prediction times
`specific_storage`	aquifer specific storage
`hydraulic_conductivity`	aquifer hydraulic conductivity
`thickness`	aquifer thickness
`flow_dimension`	flow dimension

Value

theis solution for multiple pumping scenario

`grf_time` *grf_time*

Description

Convolution of GRF well function and flow rates in the time domain. Time series needs to be regularily spaced and so are the flow rates. Some performance gains can be achieved if the number of flow rate does not change for each time.

Usage

```
grf_time(
  radius,
  specific_storage,
  hydraulic_conductivity,
  thickness,
  time,
  flow_rate,
  flow_dimension
)
```

Arguments

`radius`	distance to monitoring interval
`specific_storage`	aquifer storativity
`hydraulic_conductivity`	aquifer hydraulic conductivity
`thickness`	aquifer thickness
`time`	prediction times
`flow_rate`	well flow rates
`flow_dimension`	flow dimension

Value

theis solution for multiple pumping scenario

`hantush_jacob` *hantush_jacob*

Description

Convolution of hantush well function and flow rates in the time domain. Time series needs to be regularly spaced.

Usage

```
hantush_jacob(
  time,
  flow_rate,
  radius,
  storativity,
  transmissivity,
  leakage,
  precision
)
```

Arguments

`time`	prediction times
`flow_rate`	well flow rates
`radius`	distance to monitoring interval
`storativity`	aquifer storativity
`transmissivity`	aquifer transmissivity
`leakage`	hantush leakage
`precision`	how precise should the solution be. More is more precise but slower.

Value

hantush jacob solution for multiple pumping scenario

`hantush_well` *hantush_well*

Description

Result of the hantush well function

Prodanoff, J.H.A., Mansur, W.J. and Mascarenhas, F.C.B., 2006. Numerical evaluation of Theis and Hantush-Jacob well functions. Journal of hydrology, 318(1-4), pp.173-183.

Usage

```
hantush_well(u, b, precision)
```

Arguments

u	value of the Theis u
b	the leakance
precision	how precise should the solution be. More is more precise but slower.

Value

hantush well function

---

harmonic_list *harmonic_list*

Description

Create sin and cosine terms for harmonic analysis

Usage

```
harmonic_list(time, frequency, start, cycle_size)
```

Arguments

time	numeric vector of times
frequency	numeric vector of frequencies
start	time the cycle starts
cycle_size	size of the cycle in number of measurements

Value

List of cosines and sines

`hsieh_1987_fig_2_3` *Hsieh (1987) Figures 2 and 3 Digitized*

Description

Amplitude and phase response as a function of S.

Usage

```
hsieh_1987_fig_2_3
```

Format

A `data.table` The columns are:

`dimensionless_frequency` dimensionless frequency

`response` gain and phase of response

`S` storativity

`variable` gain or phase

Examples

```
utils::data(hsieh_1987_fig_2_3)
```

`hussein_gain` *hussein_gain*

Description

Hussein 2013, Figure 5 gain

Usage

```
data(hussein_gain)
```

Format

data.table

Examples

```
data(hussein_gain)
```

`hussein_phase` *hussein_phase*

Description

Hussein 2013, Figure 5 phase

Usage

```
data(hussein_phase)
```

Format

data.table

Examples

```
data(hussein_phase)
```

`kdr` *kdr*

Description

measured drawdown from kruseman and de ridder 2000 (figure 3.3). Oude Korendijk pumping test

Usage

```
data(kdr)
```

Format

data.table

Examples

```
data(kdr)
```

`kelvin` *kelvin Kelvin functions of the second kind ker and kei and order 0 to 1.*

Description

kelvin Kelvin functions of the second kind ker and kei and order 0 to 1.

Usage

```
kelvin(z, nSeq = 2)
```

Arguments

`z`	value to evaluate the kelvin functions
`nSeq`	positive integer; if > 1, computes the result for a whole *sequence* of `nu` values; if `nu >= 0`,`nu, nu+1, ..., nu+nSeq-1`, if `nu < 0`, `nu, nu-1, ..., nu-nSeq+1`.

Value

data.table of real and imaginary kelvin functions

`kennel_2020` *kennel_2020*

Description

SSFL 2016-08-18 00:00:00 to 2016-10-13 12:00:00 barometric pressure and water pressure data from the same hole. 1 minute interval, pressures in dbar, RBRsoloD 20dbar

Usage

```
data(kennel_2020)
```

Format

data.table

Examples

```
data(kennel_2020)
```

`lag_list` *lag_list*

Description

Create lagged terms

Usage

```
lag_list(x, lags, n_subset, n_shift)
```

Arguments

`x`	numeric vector - variable to lag
`lags`	integer vector - amount to lag
`n_subset`	take every n_subset rows
`n_shift`	shift values from starting on first row. Should be less than n_subset

Value

List of lagged terms

`liu_1989_fig_8` *Liu (1989) Figure 8 Digitized*

Description

Amplitude comparison of Cooper and Liu.

Usage

```
liu_1989_fig_8
```

Format

A `data.table` The columns are:

`period` dimensionless frequency

`response` gain of response

`aquifer_thickness` thickness of the aquifer (m)

`method` cooper or liu method

Examples

```
utils::data(liu_1989_fig_8)
```

`log_lags` *log_lags*

Description

Generate logarithmically spaced lags

Usage

```
log_lags(n, lag_max)
```

Arguments

`n`	integer number of lag terms
`lag_max`	integer the maximum lag

Value

vector of logarithmically spaced lags

`pad_num` *pad_num*

Description

This function creates a sequence of numbers with 0 padding based on the series length.

Usage

```
pad_num(n, pad = "0")
```

Arguments

`n`	the number of columns
`pad`	the character to use for the padding.

Value

a character string padded by ”0”

`plate` *plate*

Description

Get the results from the recipe. If the recipe hasn't been prepped and baked, this will do those steps and return the result.

Usage

```
plate(.rec, type = "dt", ...)
```

Arguments

`.rec`	the R6 recipe object.
`type`	the return type for the recipe (dt = 'data.table', df = 'data.frame', tbl = 'tibble', list = 'list', m = 'matrix')
`...`	additional arguments

Value

an updated recipe

Examples

```
dat <- data.frame(x = rnorm(10), y = rnorm(10))

rec <- recipe(y~x, data = dat) |>
       step_scale(x) |>
       prep() |>
       bake() |>
       plate()

rec <- recipe(y~x, data = dat) |>
       step_scale(x) |>
       plate()
```

`prep` *prep*

Description

prep a recipe

Usage

```
prep(.rec, retain = TRUE)
```

Arguments

`.rec`	the R6 recipe object.
`retain`	logical - currently not implemented

Value

an updated recipe

Examples

```
rec <- recipe(y~x, data = list(x = rnorm(10), y = rnorm(10))) |>
       step_scale(x) |>
       prep()
```

`recipe` *Create a new R6 recipe. This is analogous to the the list structure that the *recipes* package uses.*

Description

Create a new recipe object.

Usage

```
recipe(formula, data, ...)
```

Arguments

`formula`	The model formula. It cannot contain operations.
`data`	list, data.frame, data.table, tibble of data. They will all be treated as lists.
`...`	additional arguments to pass to Recipe$new(). This is currently not used.

Value

A new R6 'Recipe' object.

Examples

```
dat <- data.frame(x = rnorm(10), y = rnorm(10))

rec <- recipe(y~x, data = dat)
```

---

rojstaczer_1988a_fig_3

Rojstaczer (1988a) Figure 3 Digitized

Description

Barometric efficiency and phase as a function of dimensionless frequency

Usage

```
rojstaczer_1988a_fig_3
```

Format

A `data.table` The columns are:

`W` dimensionless frequency

`response` gain and phase of response

`variable` gain or phase

Examples

```
utils::data(rojstaczer_1988a_fig_3)
```

---

rojstaczer_1988b_fig_3

Rojstaczer (1988b) Figure 3 Digitized

Description

Barometric efficiency and phase as a function of Q/W. Static confined barometric efficiency is 0.5 and R « Q. S and S' are 0.0001.

Usage

```
rojstaczer_1988b_fig_3
```

Format

A `data.table` The columns are:

`W` dimensionless frequency

`response` gain and phase of response

`Q_div_W` ratio comparing water table drainage and borehole storage

`variable` gain or phase

Examples

```
utils::data(rojstaczer_1988b_fig_3)
```

---

rojstaczer_1988b_fig_5

Rojstaczer (1988b) Figure 5 Digitized

Description

Amplitude and phase response as a function of S.

Usage

```
rojstaczer_1988b_fig_5
```

Format

A `data.table` The columns are:

`W` dimensionless frequency

`response` gain and phase of response

`S` storativity

`variable` gain or phase

Examples

```
utils::data(rojstaczer_1988b_fig_5)
```

`rojstaczer_1988b_fig_6`

Rojstaczer (1988b) Figure 6 Digitized

Description

Amplitude and phase response as a function of R_div_Q and dimesionless frequency.

Usage

```
rojstaczer_1988b_fig_6
```

Format

A `data.table` The columns are:

`dimensionless_frequency` dimensionless frequency

`response` gain and phase of response

`R_div_Q` R div Q

`variable` gain or phase

Examples

```
utils::data(rojstaczer_1988b_fig_6)
```

`rojstaczer_1990_fig_2`

Rojstaczer and Riley (1990) Figure 2 Digitized

Description

Amplitude of water table response to Earth tides as a function of Ω'

Usage

```
rojstaczer_1990_fig_2
```

Format

A `data.table` The columns are:

`ohm` water table parameter

`response` amplitude of the response

Examples

```
utils::data(rojstaczer_1990_fig_2)
plot(response~ohm, rojstaczer_1990_fig_2, log = 'x')
```

`rojstaczer_1990_fig_3`

Rojstaczer and Riley (1990) Figure 3 Digitized

Description

Areal strain sensitivity and phase of phreatic well to Earth tides as a function of dimensionless frequency Qu and partial penetration. Strain sensitivity is 0.5 (I think this is wrong in the paper where it is listed as 0.05)

Usage

```
rojstaczer_1990_fig_3
```

Format

A `data.table` The columns are:

`Qu` dimensionless frequency

`response` gain and phase of response

`b_div_d` partial penetration saturated thickness / aquifer thickness

`variable` gain or phase

Examples

```
utils::data(rojstaczer_1990_fig_3)
```

`rojstaczer_1990_fig_4`

Rojstaczer and Riley (1990) Figure 4 Digitized

Description

Barometric efficiency and phase of phreatic well to atmospheric loading as a function of dimensionless frequency Qu and R/Qu and fully penetrating. Static barometic efficiency is 0.5.

Usage

```
rojstaczer_1990_fig_4
```

Format

A `data.table` The columns are:

`Qu` dimensionless frequency

`response` gain and phase of response

`R_div_Qu` ratio comparing water table drainage and unsaturated zone influences

`variable` gain or phase

Examples

```
utils::data(rojstaczer_1990_fig_4)
```

---

step_add_noise *step_add_noise*

Description

Add noise.

Usage

```
step_add_noise(
  .rec,
  terms,
  sd = 1,
  mean = 0,
  fun = rnorm,
  role = "predictor",
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`terms`	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
`sd`	the standard deviation of the noise to add
`mean`	the mean of the noise to add
`fun`	the random noise generating function
`role`	character - the name of the role
`...`	additional arguments

Value

add noise to a variable

Examples

```
dat <- data.frame(x = rnorm(100), y = rnorm(100))
rec <- recipe(y~x, data = dat) |>
       step_add_noise(x) |> plate()
```

`step_add_vars` *step_add_vars*

Description

Add a variable from the initial (template) data not included in the recipe creation.

Usage

```
step_add_vars(.rec, terms, role = "predictor", ...)
```

Arguments

`.rec`	the R6 recipe object.
`terms`	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
`role`	character - the name of the role
`...`	additional arguments

Value

recipe with added variables

Examples

```
dat <- data.frame(x = rnorm(10), y = rnorm(10), z = rnorm(10))

rec <- recipe(y~x, data = dat) |>
       step_add_vars(z) |>
       plate()
```

`step_aquifer_constant_drawdown` *step_aquifer_constant_drawdown*

Description

Estimates the flows of a constant drawdown test using Jacob-Lohman 1952.

Usage

```
step_aquifer_constant_drawdown(
  .rec,
  time,
  drawdown = 1,
  thickness = 1,
  radius_well = 0.15,
  specific_storage = 1e-06,
  hydraulic_conductivity = 1e-04,
  n_terms = 16L,
  role = "predictor",
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`time`	the time for evaluation (t)
`drawdown`	drawdown at the well (L)
`thickness`	the aquifer thickness (L)
`radius_well`	the radius of the well (L)
`specific_storage`	specific storage of aquifer (L/L)
`hydraulic_conductivity`	the hydraulic conductivity (L/t)
`n_terms`	number of terms for laplace solution inversion
`role`	character - the name of the role
`...`	additional arguments

Value

an updated recipe

References

Jacob, C.E. and S.W. Lohman, 1952. Nonsteady flow to a well of constant drawdown in an extensive aquifer, Trans. Am. Geophys. Union, vol. 33, pp. 559-569.

See Also

Other aquifer: `step_aquifer_grf()`, `step_aquifer_leaky()`, `step_aquifer_patch()`, `step_aquifer_theis()`, `step_aquifer_theis_aniso()`, `step_aquifer_wellbore_storage()`

Examples

```
time <- 10^seq(-5, 2, 0.1)
form <- formula(time~.)
dat <- data.frame(time = time)

jl = recipe(formula = form, data = dat) |>
  step_aquifer_constant_drawdown(time = time,
                                 drawdown = 10,
                                 thickness = 10,
                                 radius_well = 0.15,
                                 specific_storage = 1e-6,
                                 hydraulic_conductivity = 1,
                                 n_terms = 12L) |>
  plate()
```

---

step_aquifer_grf *step_aquifer_grf*

Description

Generates the drawdown using the Generalized Radial Flow (GRF) model. This method defaults to a fast FFT convolution so many rates can be included, but requires a regular time series.

Usage

```
step_aquifer_grf(
  .rec,
  time,
  flow_rate,
  thickness = 1,
  radius = 100,
  specific_storage = 1e-06,
  hydraulic_conductivity = 1e-04,
  flow_dimension = 2,
  role = "predictor",
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`time`	the time for evaluation (t)
`flow_rate`	the flow rate from the well (L^3/t)
`thickness`	the aquifer thickness (L)
`radius`	the distance to the observation location (L)
`specific_storage`	specific storage of aquifer (L/L)
`hydraulic_conductivity`	the hydraulic conductivity (L/t)
`flow_dimension`	aquifer flow dimension (1 = linear, 2 = radial, 3 = spherical)
`role`	character - the name of the role
`...`	additional arguments

Value

The drawdown using the GRF model

References

Barker, J.A., A generalized radial flow model for hydraulic tests in fractured rock. Water Resour. Res., 24 (1988), pp. 1796-1804, 10.1029/WR024i010p01796

See Also

Other aquifer: `step_aquifer_constant_drawdown()`, `step_aquifer_leaky()`, `step_aquifer_patch()`, `step_aquifer_theis()`, `step_aquifer_theis_aniso()`, `step_aquifer_wellbore_storage()`

Examples

```
time <- 1:2000
flow_rate  <- c(rep(0.001, 500),
                rep(0.002, 500),
                rep(0.0, 1000))

dat <- data.frame(time, flow_rate)

# radial (flow_dimension = 2 Theis)
dd_rad <- recipe(time~flow_rate, dat) |>
  step_aquifer_grf(time = time,
                   flow_rate = flow_rate,
                   thickness = 1.0,
                   radius = 20,
                   specific_storage = 1e-5,
                   hydraulic_conductivity = 1e-3,
                   flow_dimension = 2) |>
  plate()
```

```
# linear (flow_dimension = 1)

dd_lin <- recipe(time~flow_rate, dat) |>
  step_aquifer_grf(time = time,
                   flow_rate = flow_rate,
                   thickness = 1.0,
                   radius = 20,
                   specific_storage = 1e-5,
                   hydraulic_conductivity = 1e-3,
                   flow_dimension = 1) |>
  plate()

# spherical (flow_dimension = 3)
dd_sph <- recipe(time~flow_rate, dat) |>
  step_aquifer_grf(time = time,
                   flow_rate = flow_rate,
                   thickness = 1.0,
                   radius = 20,
                   specific_storage = 1e-5,
                   hydraulic_conductivity = 1e-3,
                   flow_dimension = 3) |>
  plate()
```

`step_aquifer_leaky` *step_aquifer_leaky*

Description

hantush_jacob 1955 solution for a leaky aquifer

Usage

```
step_aquifer_leaky(
  .rec,
  time,
  flow_rate,
  thickness = 1,
  leakage = 100,
  radius = 100,
  specific_storage = 1e-06,
  hydraulic_conductivity = 1e-04,
  precision = 1e-10,
  role = "predictor",
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`time`	the time for evaluation (t)
`flow_rate`	the flow rate from the well (L^3/t)
`thickness`	the aquifer thickness (L)
`leakage`	the leakage defined by hantush (smaller indicates more leaky)
`radius`	the distance to the observation location (L)
`specific_storage`	specific storage of aquifer (L/L)
`hydraulic_conductivity`	the hydraulic conductivity (L/t)
`precision`	the precision of the solution (default 1e-10)
`role`	character - the name of the role
`...`	additional arguments

Value

The drawdown using the Hantush and Jacob 1955 model

References

Prodanoff, J.H.A., Mansur, W.J. and Mascarenhas, F.C.B., 2006. Numerical evaluation of Theis and Hantush-Jacob well functions. Journal of hydrology, 318(1-4), pp.173-183. eq: 10, 11, 12

Hantush, M.S. and C.E. Jacob, 1955. Non-steady radial flow in an infinite leaky aquifer, Am. Geophys. Union Trans., vol. 36, no. 1, pp. 95-100.

See Also

Other aquifer: `step_aquifer_constant_drawdown()`, `step_aquifer_grf()`, `step_aquifer_patch()`, `step_aquifer_theis()`, `step_aquifer_theis_aniso()`, `step_aquifer_wellbore_storage()`

Examples

```
time <- 1:2000
flow_rate <- c(rep(0.001, 500),
               rep(0.002, 500),
               rep(0.0, 1000))

# high
dat <- data.frame(time = 1:2000, flow_rate = flow_rate)
hj_100 <- recipe(flow_rate~time, dat) |>
  step_aquifer_leaky(time,
                     flow_rate,
                     leakage = 100,
                     radius = 100,
                     storativity = 1e-6,
```

```
                      transmissivity = 1e-4) |>
  plate()

# medium
hj_200 <- recipe(flow_rate~time, dat) |>
  step_aquifer_leaky(time,
                     flow_rate,
                     leakage = 200,
                     radius = 100,
                     storativity = 1e-6,
                     transmissivity = 1e-4) |>
  plate()

# low
hj_1000 <- recipe(flow_rate~time, dat) |>
  step_aquifer_leaky(time,
                     flow_rate,
                     leakage = 1000,
                     radius = 100,
                     storativity = 1e-6,
                     transmissivity = 1e-4) |>
  plate()
```

`step_aquifer_patch` *step_aquifer_patch*

Description

barker_herbert 1982 solution for radial patches.

Usage

```
step_aquifer_patch(
  .rec,
  time,
  flow_rate = 0.01,
  thickness = 1,
  radius = 200,
  radius_patch = 100,
  specific_storage_inner = 1e-06,
  specific_storage_outer = 1e-05,
  hydraulic_conductivity_inner = 1e-04,
  hydraulic_conductivity_outer = 1e-06,
  n_stehfest = 12L,
  role = "predictor",
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`time`	the time for evaluation (t)
`flow_rate`	the flow rate from the well (L^3/t)
`thickness`	the aquifer thickness (L)
`radius`	the distance to the observation location (L)
`radius_patch`	the radius of the cylindrical patch (L)
`specific_storage_inner`	specific storage of inner patch (L/L)
`specific_storage_outer`	specific storage of outer patch (L/L)
`hydraulic_conductivity_inner`	the hydraulic conductivity of the inner patch (L/t)
`hydraulic_conductivity_outer`	the hydraulic conductivity of the outer patch (L/t)
`n_stehfest`	integer number of terms to use in Stehfest method (typically < 18)
`role`	character - the name of the role
`...`	additional arguments

Value

The drawdown using the Theis model

References

Barker, J.A., and R. Herbert, 1982: Pumping tests in patchy aquifers, Ground Water, vol. 20, No. 2, pp. 150-155.

Butler, J.J., 1988: Pumping tests in nonuniform aquifers – The radially symmetric case, Journal of Hydrology, Vol. 101, pp. 15-30.

See Also

Other aquifer: `step_aquifer_constant_drawdown()`, `step_aquifer_grf()`, `step_aquifer_leaky()`, `step_aquifer_theis()`, `step_aquifer_theis_aniso()`, `step_aquifer_wellbore_storage()`

Examples

```
dat <- data.frame(time = as.numeric(1:100))
formula <- as.formula(time~.)

frec = recipe(formula = formula, data = dat) |>
  step_aquifer_patch(time = time, flow_rate = 0.01) |>
  plate()
```

`step_aquifer_theis` *step_aquifer_theis*

Description

Radial Flow (GRF) model with flow dimension equal to 2. This method defaults to a fast FFT convolution so many rates can be included, but requires a regular time series.

Usage

```
step_aquifer_theis(
  .rec,
  time,
  flow_rate,
  thickness = 1,
  radius = 100,
  specific_storage = 1e-06,
  hydraulic_conductivity = 1e-04,
  role = "predictor",
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`time`	the time for evaluation (t)
`flow_rate`	the flow rate from the well (L^3/t)
`thickness`	the aquifer thickness (L)
`radius`	the distance to the observation location (L)
`specific_storage`	specific storage of aquifer (L/L)
`hydraulic_conductivity`	the hydraulic conductivity (L/t)
`role`	character - the name of the role
`...`	additional arguments

Value

The drawdown using the Theis model

References

Barker, J.A., A generalized radial flow model for hydraulic tests in fractured rock. Water Resour. Res., 24 (1988), pp. 1796-1804, 10.1029/WR024i010p01796

Theis, C.V., 1935: The relation between the lowering of the piezometric surface and the rate and duration of discharge of a well using ground-water storage, Transactions of the American Geophysical Union, 16th Annual Meeting, Part 2, pp. 519-524.

See Also

Other aquifer: step_aquifer_constant_drawdown(), step_aquifer_grf(), step_aquifer_leaky(), step_aquifer_patch(), step_aquifer_theis_aniso(), step_aquifer_wellbore_storage()

Examples

```
dat <- data.frame(x = as.numeric(1:100),
                  y = rep(0.01, 100))
formula <- as.formula(y~x)

frec = recipe(formula = formula, data = dat) |>
  step_aquifer_theis(time = x, flow_rate = y) |>
  prep() |>
  bake()
```

---

step_aquifer_theis_aniso

step_aquifer_theis_aniso

Description

Generates the drawdown using the Papadopulos 1965 model. Papadopulos, I.S., 1965. Nonsteady flow to a well in an infinite anisotropic aquifer.

Usage

```
step_aquifer_theis_aniso(
  .rec,
  time,
  flow_rate,
  thickness = 1,
  distance_x = 100,
  distance_y = 100,
  specific_storage = 1e-06,
  hydraulic_conductivity_major = 1e-04,
  hydraulic_conductivity_minor = 1e-05,
  major_axis_angle = 0,
  role = "predictor",
  ...
)
```

Arguments

.rec	the R6 recipe object.
time	the time for evaluation (t)
flow_rate	the flow rate from the well (L^3/t)

`thickness` the aquifer thickness (L)

`distance_x` distance in the x direction

`distance_y` distance in the y direction

`specific_storage`
specific storage of aquifer (L/L)

`hydraulic_conductivity_major`
hydraulic conductivity in the major principal direction

`hydraulic_conductivity_minor`
hydraulic conductivity in the minor principal direction

`major_axis_angle`
the orientation of the major principal axis (angle between the x axis and major axis)

`role` character - the name of the role

`...` additional arguments

Value

The drawdown using the Papadopulos 1965 model

References

Papadopulos, I.S., 1965. Nonsteady flow to a well in an infinite anisotropic aquifer. Heilweil, V.M. and Hsieh, P.A., 2006. Determining anisotropic transmissivity using a simplified Papadopulos method. Groundwater, 44(5), pp.749-753.

See Also

Other aquifer: `step_aquifer_constant_drawdown()`, `step_aquifer_grf()`, `step_aquifer_leaky()`, `step_aquifer_patch()`, `step_aquifer_theis()`, `step_aquifer_wellbore_storage()`

Examples

```
dat <- data.frame(x = as.numeric(1:100),
                  y = rep(0.01, 100))
formula <- as.formula(y~x)

frec = recipe(formula = formula, data = dat) |>
  step_aquifer_theis_aniso(time = x, flow_rate = y) |>
  prep() |>
  bake()
```

`step_aquifer_wellbore_storage` *step_aquifer_patch*

Description

Papadopulos-Cooper 1967 solution for wellbore storage.

Usage

```
step_aquifer_wellbore_storage(
  .rec,
  time,
  flow_rate = 1,
  radius = 0.15,
  radius_casing = 0.15,
  radius_well = 0.15,
  thickness = 1,
  specific_storage = 1e-06,
  hydraulic_conductivity = 1e-04,
  n_terms = 12L,
  role = "predictor",
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`time`	the time for evaluation (t)
`flow_rate`	the flow rate from the well (L^3/t)
`radius`	the distance to the observation location (L)
`radius_casing`	radius of casing in the interval over which the water level declines (L)
`radius_well`	effective radius of well screen or open hole (L)
`thickness`	the aquifer thickness (L)
`specific_storage`	specific storage of aquifer (L/L)
`hydraulic_conductivity`	the hydraulic conductivity (L/t)
`n_terms`	number of terms for laplace solution inversion
`role`	character - the name of the role
`...`	additional arguments

Value

The drawdown using the Papadopulos-Cooper model

References

Papadopulos, I.S. and H.H. Cooper, 1967. Drawdown in a well of large diameter, Water Resources Research, vol. 3, no. 1, pp. 241-244.

See Also

Other aquifer: step_aquifer_constant_drawdown(), step_aquifer_grf(), step_aquifer_leaky(), step_aquifer_patch(), step_aquifer_theis(), step_aquifer_theis_aniso()

Examples

```
dat <- data.frame(x = 10^seq(-5, 2, length.out = 100),
                  y = rep(0.01, 100))
formula <- as.formula(y~x)

frec = recipe(formula = formula, data = dat) |>
  step_aquifer_wellbore_storage(time = x,
                                flow_rate = 10.0,
                                hydraulic_conductivity = 10.0,
                                specific_storage = 1e-4) |>
  prep() |>
  bake()
```

---

step_baro_clark *step_baro_clark*

Description

Clark 1967 solution for calculating barometric efficiency (Algorithm from Batu 1998, pg 76)

Usage

```
step_baro_clark(
  .rec,
  water_level,
  barometric_pressure,
  lag_space = 1L,
  inverse = FALSE,
  role = "augment",
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`water_level`	`numeric vector` of the dependent variable (ie:water level)
`barometric_pressure`	`numeric vector` of the independent variable (ie:barometric pressure)
`lag_space`	`integer` spacing for lags, useful for higher frequency monitoring
`inverse`	`logical` whether the barometric relationship is inverse
`role`	character - the name of the role
`...`	additional arguments

Value

barometric efficiency using Clark's method

References

Clark, W.E., 1967. Computing the barometric efficiency of a well. Journal of the Hydraulics Division, 93(4), pp.93-98.

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

See Also

Other barometric: `step_baro_harmonic()`, `step_baro_least_squares()`

Examples

```
data(kennel_2020)

clarks <- recipe(wl~., kennel_2020) |>
  step_baro_clark(wl, baro, lag_space = 1) |> # 1 minutes (every minute differences)
  step_baro_clark(wl, baro, lag_space = 60) |> # 60 minutes (hourly differences)
  step_baro_clark(wl, baro, lag_space = 1440) |> # 1440 minutes (daily differences)
  prep() |>
  bake()

clarks$get_step_data("barometric_efficiency")
```

`step_baro_frequency_semi_confined` *step_baro_frequency_semi_confined*

Description

Work in Progress: Do not use. Rojstaczer 1988 solution

step_baro_frequency_semi_confined

Usage

```
step_baro_frequency_semi_confined(
  .rec,
  frequency,
  radius_well,
  transmissivity,
  storage_confining,
  storage_aquifer,
  diffusivity_confining,
  diffusivity_vadose,
  thickness_confining,
  thickness_vadose,
  loading_efficiency,
  attenuation,
  role = "predictor",
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`frequency`	the frequency of the response
`radius_well`	well radius
`transmissivity`	aquifer transmissivity (L*L/t)
`storage_confining`	confining layer storativity (L/L)
`storage_aquifer`	aquifer storativity (L/L)
`diffusivity_confining`	confining layer diffusivity
`diffusivity_vadose`	air diffusivity of vadose zone
`thickness_confining`	confining layer thickness

`thickness_vadose`
: vadose thickness

`loading_efficiency`
: the loading efficiency of the aquifer

`attenuation`
: an attenuation factor

`role`
: character - the name of the role

`...`
: additional arguments

Value

complex response vector in frequency domain

`step_baro_frequency_unconfined` *step_baro_frequency_semi_confined*

Description

Work in Progress: Do not use. Rojstaczer 1988 solution

step_baro_frequency_unconfined

Usage

```
step_baro_frequency_unconfined(
  .rec,
  frequency,
  radius_well,
  storage_aquifer,
  specific_yield,
  k_vertical,
  diffusivity_vertical,
  diffusivity_vadose,
  thickness_saturated_well,
  thickness_vadose,
  thickness_aquifer,
  loading_efficiency,
  attenuation,
  role = "predictor",
  ...
)
```

Arguments

`.rec`
: the R6 recipe object.

`frequency`
: the frequency of the response

`radius_well`
: well radius

`storage_aquifer`
: aquifer storativity (L/L)

`specific_yield`
: the specific yeild of of the unconfined system

`k_vertical`
: vertical hydraulic conductivity of unconfined aquifer

`diffusivity_vertical`
: vertical diffusivity of unconfined aquifer

`diffusivity_vadose`
: air diffusivity of vadose zone

`thickness_saturated_well`
: length of saturated well

`thickness_vadose`
: vadose thickness

`thickness_aquifer`
: aquifer thickness

`loading_efficiency`
: the loading efficiency of the aquifer

`attenuation`
: an attenuation factor

`role`
: character - the name of the role

`...`
: additional arguments

Value

complex response vector in frequency domain

`step_baro_harmonic` *step_baro_harmonic*

Description

Estimate the static barometric efficiency using harmonic methods (Ratio, Acworth, Rau, Transfer)

Usage

```
step_baro_harmonic(
  .rec,
  time,
  water_level,
  barometric_pressure,
  earth_tide,
  frequency = c(1.9324, 2),
  cycle_size = 86400,
  starting_value = 0,
  inverse = FALSE,
  role = "augment",
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`time`	name of column that holds the time information
`water_level`	`numeric vector` of the dependent variable (ie:water level)
`barometric_pressure`	`numeric vector` of the independent variable (ie:barometric pressure)
`earth_tide`	`variable` unquoted Earth tide column name
`frequency`	numeric vector - the frequencies of the sin and cos curves
`cycle_size`	numeric - the period of the sin and cos curves
`starting_value`	numeric - the starting position of the sin and cos curves. This may be specified to have more control over the signal phase.
`inverse`	`logical` whether the barometric relationship is inverse
`role`	character - the name of the role
`...`	additional arguments

Value

`double` barometric efficiency using different methods

References

Acworth, R.I., Halloran, L.J., Rau, G.C., Cuthbert, M.O. and Bernardi, T.L., 2016. An objective frequency domain method for quantifying confined aquifer compressible storage using Earth and atmospheric tides. Geophysical Research Letters, 43(22), pp.11-671.

See Also

Other barometric: `step_baro_clark()`, `step_baro_least_squares()`

Examples

```
data("kennel_2020")
library(data.table)
library(collapse)

formula <- as.formula(wl~.)
frec = recipe(formula = formula, data = kennel_2020) |>
 step_baro_harmonic(datetime,
                    wl,
                    baro,
                    et,
                    inverse = FALSE)
```

`step_baro_least_squares` *step_baro_least_squares*

Description

Least squares solution for calculating barometric efficiency

Usage

```
step_baro_least_squares(
  .rec,
  water_level,
  barometric_pressure,
  lag_space = 1L,
  inverse = FALSE,
  differences = FALSE,
  role = "augment",
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`water_level`	`numeric vector` of the dependent variable (ie:water level)
`barometric_pressure`	`numeric vector` of the independent variable (ie:barometric pressure)
`lag_space`	`integer` spacing for lags, useful for higher frequency monitoring
`inverse`	`logical` whether the barometric relationship is inverse
`differences`	`numeric vector` number of samples between differences
`role`	character - the name of the role
`...`	additional arguments

Value

barometric efficiency using least squares

See Also

Other barometric: `step_baro_clark()`, `step_baro_harmonic()`

Examples

```
data(kennel_2020)

least_squares <- recipe(wl~., kennel_2020) |>
  step_baro_least_squares(wl, baro) |> # 1 minutes (every minute differences)
  step_baro_least_squares(wl, baro, lag_space = 1440, differences = TRUE) |>
  prep() |>
  bake()

least_squares$get_step_data("barometric_efficiency")
```

`step_center` *step_center*

Description

Adds a step to center a data column(s)

Usage

```
step_center(
  .rec,
  terms,
  role = "predictor",
  skip = FALSE,
  na_rm = TRUE,
  fun = collapse::fmean,
  keep_original_cols = FALSE,
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`terms`	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
`role`	character - the name of the role
`skip`	logical - should the step be skipped
`na_rm`	logical - should NA values be removed from calculations
`fun`	function - the function that is applied to a list or columns of a data.frame like object.
`keep_original_cols`	logical - keep the original columns or replace them
`...`	additional arguments

Value

an updated recipe

Examples

```
dat <- data.frame(x = rnorm(10), y = rnorm(10))

rec <- recipe(y~x, data = dat) |>
       step_center(x) |>
       prep() |>
       bake()
```

`step_check_na` *step_check_na*

Description

Check columns for NA

Usage

```
step_check_na(.rec, terms, role = "check", ...)
```

Arguments

`.rec`	the R6 recipe object.
`terms`	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
`role`	character - the name of the role
`...`	additional arguments

Value

an updated recipe

Examples

```
dat <- data.frame(x = rnorm(10), y = rnorm(10))

rec <- recipe(y~x, data = dat) |>
       step_check_na(x) |>
       prep() |>
       bake()
```

step_check_spacing *step_check_spacing*

Description

Check the spacing of a variable

Usage

```
step_check_spacing(.rec, terms, role = "check", ...)
```

Arguments

.rec	the R6 recipe object.
terms	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
role	character - the name of the role
...	additional arguments

Value

an updated recipe

Examples

```
dat <- data.frame(x = rnorm(10), y = rnorm(10))

rec <- recipe(y~x, data = dat) |>
       step_check_spacing(x) |>
       prep() |>
       bake()
```

step_compare_columns *step_check_spacing*

Description

Check the spacing of a variable

Usage

```
step_compare_columns(
  .rec,
  data,
  compare,
  role = "add",
  n_sd = 4,
  na_rm = TRUE,
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`data`	`variable` unquoted data column name
`compare`	`variable` unquoted column name for comparison
`role`	character - the name of the role
`n_sd`	numeric - number of standard deviations for the scaling
`na_rm`	logical - should NA values be removed from calculations
`...`	additional arguments

Value

an updated recipe

Examples

```
data("kennel_2020")

kennel_2020[1e4, wl := 13.36]

frec = recipe(wl~baro, data = kennel_2020) |>
 step_compare_columns(data = wl, compare = baro, n_sd = 15) |>
 prep() |>
 bake()
```

---

step_convolve_exponential

step_convolve_exponential

Description

linearly convolve a gamma kernel with a data series.

Usage

```
step_convolve_exponential(
  .rec,
  terms,
  amplitude,
  theta,
  align = "right",
  max_length = Inf,
  role = "predictor",
  ...
)
```

Arguments

.rec	the R6 recipe object.
terms	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
amplitude	amplitude
theta	scale
align	character center, left or right align the convolution
max_length	the maximum length of the kernel
role	character - the name of the role
...	additional arguments

Value

an updated recipe

Examples

```
formula <- as.formula(x~y+z)
rows <- 1e4

dat <- data.frame(x = rep(1, rows),
                  y = 1:rows,
                  z = cumsum(rnorm(rows)))

frec = recipe(formula = formula, data = dat) |>
  step_convolve_gamma(z, amplitude = 1, theta = 1, k = 1) |>
  plate("tbl")
```

`step_convolve_gamma` *step_convolve_gamma*

Description

linearly convolve a gamma kernel with a data series.

Usage

```
step_convolve_gamma(
  .rec,
  terms,
  amplitude,
  k,
  theta,
  align = "right",
  max_length = Inf,
  role = "predictor",
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`terms`	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
`amplitude`	amplitude
`k`	shape
`theta`	scale
`align`	character center, left or right align the convolution
`max_length`	the maximum length of the kernel
`role`	character - the name of the role
`...`	additional arguments

Value

an updated recipe

Examples

```
formula <- as.formula(x~y+z)
rows <- 1e4

dat <- data.frame(x = rep(1, rows),
                  y = 1:rows,
                  z = cumsum(rnorm(rows)))

frec = recipe(formula = formula, data = dat) |>
  step_convolve_gamma(z, amplitude = 1, theta = 1, k = 1) |>
  plate("tbl")
```

`step_cross_correlation` *step_cross_correlation*

Description

Calculate the autocorrelation function or cross-correlation

Usage

```
step_cross_correlation(.rec, terms, lag_max = 100, role = "predictor", ...)
```

Arguments

`.rec`	the R6 recipe object.
`terms`	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
`lag_max`	maximum lag to calculate
`role`	character - the name of the role
`...`	additional arguments

Value

an updated recipe

Examples

```
formula <- as.formula(y~x)
rows <- 1e4

dat <- data.frame(x = rnorm(rows),
                  y = as.numeric(1:rows))

frec = recipe(formula = formula, data = dat) |>
 step_cross_correlation(x)

frec = recipe(formula = formula, data = dat) |>
 step_cross_correlation(c(x, y))
```

---

step_distributed_lag *step_distributed_lag*

Description

Generates distributed lag vectors. For regular spaced lags this uses an FFT based method which is faster and more memory efficient.

Usage

```
step_distributed_lag(
  .rec,
  terms,
  n_lag = 12L,
  lag_max = 86400L,
  knots = NA_real_,
  basis_matrix = NA_real_,
  intercept = FALSE,
  role = "predictor",
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`terms`	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
`n_lag`	number of lags to calculate (enter either lag_max and n_lag or knots)
`lag_max`	maximum lag to calculate (enter either lag_max and n_lag or knots)
`knots`	specify the knot locations

basis_matrix	user specified basis_matrix
intercept	include intercept in basis matrix
role	character - the name of the role
...	additional arguments

Value

an updated recipe

References

Gasparrini, A., 2011. Distributed Lag Linear and Non-Linear Models in R: The Package dlnm. Journal of statistical software 465 43, 1–20.

Examples

```
formula <- as.formula(y~x)
rows <- 1e4

dat <- data.frame(x = rnorm(rows),
                  y = as.numeric(1:rows),
                  z = rnorm(rows))

frec = recipe(formula = formula, data = dat) |>
 step_distributed_lag(x, knots = hydrorecipes:::log_lags_arma(6, 800))
```

---

step_drop_columns *step_drop_columns*

Description

Removes regressors from recipe.

Usage

```
step_drop_columns(.rec, terms, role = "modify", ...)
```

Arguments

.rec	the R6 recipe object.
terms	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
role	character - the name of the role
...	additional arguments

Value

an updated recipe

Examples

```
rows <- 20
formula <- as.formula(y~x)

dat <- data.frame(x = rnorm(rows),
                  y = as.numeric(1:rows),
                  z = rnorm(rows))


frec = recipe(formula = formula, data = dat) |>
  step_drop_columns(x)
```

`step_dummy` *step_dummy*

Description

dummy encode a factor or factor like variable.

Usage

```
step_dummy(
  .rec,
  terms,
  one_hot = FALSE,
  role = "predictor",
  skip = FALSE,
  keep_original_cols = FALSE,
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`terms`	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
`one_hot`	logical - use one hot encoding.
`role`	character - the name of the role
`skip`	logical - should the step be skipped
`keep_original_cols`	logical - keep the original columns or replace them
`...`	additional arguments

Value

an updated recipe

Examples

```
dat <- data.frame(x = factor(sample(1:10, 100, replace = TRUE)),
                  y = rnorm(100))

rec <- recipe(y~x, data = dat) |>
       step_dummy(x, one_hot = FALSE)
rec <- recipe(y~x, data = dat) |>
       step_dummy(x, one_hot = TRUE)
```

---

step_earthtide *step_earthtide*

Description

Generate synthetic Earth tide waves and wave groups.

Usage

```
step_earthtide(
  .rec,
  terms,
  do_predict = TRUE,
  method = "gravity",
  latitude = 0,
  longitude = 0,
  elevation = 0,
  azimuth = 0,
  gravity = 0,
  earth_radius = 6378136.3,
  earth_eccen = 0.0066943979514,
  cutoff = 1e-06,
  catalog = "ksm04",
  eop = NULL,
  scale = TRUE,
  n_thread = 1L,
  astro_update = 1L,
  interp_factor = 1L,
  role = "predictor",
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`terms`	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
`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.
`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.
`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
`n_thread`	Number of threads to use for parallel processing (integer).
`astro_update`	How often to update astro parameters in number of samples. This speeds up code but may make it slightly less accurate.
`interp_factor`	calculate for every `interp_factor` samples. This may be faster while being more accurate than adjusting the cutoff.
`role`	character - the name of the role
`...`	additional arguments

Value

an updated recipe

Examples

```
library(data.table)

data(kennel_2020)
latitude     <- 34.23411                          # latitude
longitude    <- -118.678                          # longitude
elevation    <- 500                               # elevation
cutoff       <- 1e-5                              # cutoff
catalog      <- 'ksm04'                           # hartmann wenzel catalog
astro_update <- 300                               # how often to update astro parameters
method       <- 'volume_strain'                   # which potential to calculate

wave_groups_dl <- data.table::as.data.table(earthtide::eterna_wavegroups)
wave_groups_dl <- na.omit(wave_groups_dl[time == "1 month"])
wave_groups_dl <- wave_groups_dl[wave_groups_dl$start > 0.5,]
wave_groups_dl <- wave_groups_dl[, list(start, end)]
ngr <- nrow(wave_groups_dl)

rec <- recipe(wl~baro+datetime, data = kennel_2020) |>
  step_earthtide(datetime,
                 wave_groups = wave_groups_dl,
                 latitude = latitude,
                 longitude = longitude,
                 elevation = elevation,
                 cutoff = cutoff,
                 catalog = catalog)
```

---

step_fft_coherence *step_fft_coherence*

Description

estimates the coherence between terms.

Usage

```
step_fft_coherence(.rec, terms, role = "augment", ...)
```

Arguments

.rec	the R6 recipe object.
terms	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
role	character - the name of the role
...	additional arguments

Value

an updated recipe

Examples

```
data(kennel_2020)

form <- as.formula("wl~.")

formula <- as.formula(wl~baro + et)
frec = recipe(formula = formula, data = kennel_2020) |>
  step_fft_pgram(c(wl, baro, et), spans = 7) |>
  step_fft_coherence() |>
  prep() |>
  bake()
```

---

step_fft_pgram *step_fft_pgram*

Description

Periodgrams and cross-periodograms using a method similar to `stats::spec.pgram`.

Usage

```
step_fft_pgram(
  .rec,
  terms,
  spans = 3,
  detrend = TRUE,
  demean = TRUE,
  lst = TRUE,
  taper = 0.1,
  pad_fft = TRUE,
  time_step = 1,
  role = "predictor",
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`terms`	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.

spans	vector of odd integers giving the widths of modified Daniell smoothers to be used to smooth the periodogram.
detrend	logical. If TRUE, remove a linear trend from the series. This will also remove the mean.
demean	logical. If TRUE, subtract the mean of the series.
lst	logical return a list?
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_fft	logical Zero pad the list for faster FFT calculation?
time_step	numeric monitoring interval size
role	character - the name of the role
...	additional arguments

Value

an updated recipe

Examples

```
formula <- as.formula(y~.)

dat <- data.frame(x = rnorm(200),
                  y = rnorm(200))

frec = recipe(formula = formula, data = dat) |>
  step_fft_pgram(c(x,y))
```

---

step_fft_transfer_experimental

step_fft_transfer_experimental

Description

Calculates the transfer function with pgram results.

Usage

```
step_fft_transfer_experimental(
  .rec,
  terms,
  spans = 3,
  detrend = TRUE,
  demean = TRUE,
  taper = 0.1,
```

```
  n_groups = 200,
  time_step = 1,
  formula = NULL,
  role = "augment",
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`terms`	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
`spans`	vector of odd integers giving the widths of modified Daniell smoothers to be used to smooth the periodogram.
`detrend`	logical. If `TRUE`, remove a linear trend from the series. This will also remove the mean.
`demean`	logical. If `TRUE`, subtract the mean of the series.
`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.
`n_groups`	number of results
`time_step`	`numeric` monitoring interval size
`formula`	formula notation to specify inputs and outputs
`role`	character - the name of the role
`...`	additional arguments

Value

an updated recipe

Examples

```
data(kennel_2020)

form <- as.formula("wl~.")

rec <- recipe(form, kennel_2020) |>
       step_fft_transfer_experimental(c(wl, baro, et), spans = 3) |>
       plate()
```

`step_fft_transfer_pgram` *step_fft_transfer_pgram*

Description

Calculates the transfer function with pgram results.

Usage

```
step_fft_transfer_pgram(
  .rec,
  terms,
  spans = 3,
  detrend = TRUE,
  demean = TRUE,
  taper = 0.1,
  time_step = 1,
  formula = NULL,
  role = "augment",
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`terms`	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
`spans`	vector of odd integers giving the widths of modified Daniell smoothers to be used to smooth the periodogram.
`detrend`	logical. If `TRUE`, remove a linear trend from the series. This will also remove the mean.
`demean`	logical. If `TRUE`, subtract the mean of the series.
`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.
`time_step`	`numeric` monitoring interval size
`formula`	formula notation to specify inputs and outputs
`role`	character - the name of the role
`...`	additional arguments

Value

an updated recipe

Examples

```
data(kennel_2020)

form <- as.formula("wl~.")

rec <- recipe(form, kennel_2020) |>
       step_fft_transfer_pgram(c(wl, baro, et), spans = 3) |>
       plate()
```

---

step_fft_transfer_welch

step_fft_transfer_welch

Description

calculates the transfer function with Welch's results.

Usage

```
step_fft_transfer_welch(
  .rec,
  terms,
  length_subset,
  overlap = 0.8,
  window,
  time_step = 1,
  formula = NULL,
  role = "augment",
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`terms`	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
`length_subset`	length of fft section
`overlap`	amount of overlap
`window`	window weights
`time_step`	`numeric` monitoring interval size
`formula`	formula notation to specify inputs and outputs
`role`	character - the name of the role
`...`	additional arguments

Value

an updated recipe

Examples

```
data(kennel_2020)
form <- as.formula("wl~.")

  rec <- recipe(form, kennel_2020) |>
  step_fft_transfer_welch(c(wl, baro, et),
                          length_subset = 1440*8 + 1,
                          overlap = 0.6,
                          window = window_nuttall(1440*8+1)) |>
  plate()
```

---

step_fft_welch *step_fft_welch*

Description

calculates the periodogram (estimate of spectral density) using Welch's method.

Usage

```
step_fft_welch(
  .rec,
  terms,
  length_subset,
  overlap = 0.8,
  window,
  time_step = 1,
  role = "augment",
  ...
)
```

Arguments

.rec	the R6 recipe object.
terms	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
length_subset	length of fft section
overlap	amount of overlap
window	window weights

`time_step`	`numeric` monitoring interval size
`role`	character - the name of the role
`...`	additional arguments

Value

an updated recipe

Examples

```
formula <- as.formula(y~.)

dat <- data.frame(x = rnorm(200),
                  y = rnorm(200))

frec = recipe(formula = formula, data = dat) |>
  step_fft_welch(c(x,y), length_subset = 10, window = window_rectangle(10))
```

`step_find_interval` *step_find_interval*

Description

divides a series into intervals and then performs dummy encoding.

Usage

```
step_find_interval(.rec, terms, vec, role = "augment", ...)
```

Arguments

`.rec`	the R6 recipe object.
`terms`	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
`vec`	a vector of break points
`role`	character - the name of the role
`...`	additional arguments

Value

an updated recipe

Examples

```
formula <- as.formula(y~x)

rows = 200
dat <- data.frame(x = rnorm(rows),
                  y = 1:rows,
                  z = rnorm(rows))

frec1 = recipe(formula = formula, data = dat) |>
  step_find_interval(x, vec = c(-0.1, 0.0, 0.1)) |>
  plate("tbl")
```

---

step_harmonic *step_harmonic*

Description

Add sin and cos terms for harmonic analysis

Usage

```
step_harmonic(
  .rec,
  terms,
  frequency = NA_real_,
  cycle_size = NA_real_,
  starting_value = 0,
  role = "predictor",
  skip = FALSE,
  keep_original_cols = FALSE,
  ...
)
```

Arguments

.rec	the R6 recipe object.
terms	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
frequency	numeric vector - the frequencies of the sin and cos curves
cycle_size	numeric - the period of the sin and cos curves
starting_value	numeric - the starting position of the sin and cos curves. This may be specified to have more control over the signal phase.

`role`	character - the name of the role
`skip`	logical - should the step be skipped
`keep_original_cols`	logical - keep the original columns or replace them
`...`	additional arguments

Value

an updated recipe

Examples

```
dat <- data.frame(x = 1:10, y = rnorm(10))

rec <- recipe(y~x, data = dat) |>
       step_harmonic(x,
                     frequency = 2.0,
                     cycle_size = 4.0,
                     starting_value = 0.0)
```

`step_intercept` *step_intercept*

Description

Add an intercept term

Usage

```
step_intercept(.rec, terms, value = 1, role = "predictor", ...)
```

Arguments

`.rec`	the R6 recipe object.
`terms`	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
`value`	what value to use (typically 1.0)
`role`	character - the name of the role
`...`	additional arguments

Value

an updated recipe

Examples

```
dat <- data.frame(x = 1:10, y = rnorm(10))

rec <- recipe(y~x, data = dat) |>
       step_intercept()
```

`step_kernel_divide_naive` *step_kernel_divide_naive*

Description

Divide a signal by a kernel in the frequency domain.

Usage

```
step_kernel_divide_naive(.rec, terms, kernel, role = "predictor", ...)
```

Arguments

`.rec`	the R6 recipe object.
`terms`	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
`kernel`	the convolution kernel
`role`	character - the name of the role
`...`	additional arguments

Value

an updated recipe

`step_kernel_filter` *step_kernel_filter*

Description

linearly convolve a kernel with a data series.

Usage

```
step_kernel_filter(
  .rec,
  terms,
  kernel,
  align = "center",
  role = "predictor",
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`terms`	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
`kernel`	the convolution kernel
`align`	character center, left or right align the convolution
`role`	character - the name of the role
`...`	additional arguments

Value

an updated recipe

Examples

```
formula <- as.formula(x~y+z)
rows <- 1e4

dat <- data.frame(x = rep(1, rows),
                  y = 1:rows,
                  z = cumsum(rnorm(rows)))

frec = recipe(formula = formula, data = dat) |>
  step_kernel_filter(z, kernel = list(rep(1, 1001)/1001), align = "center") |>
  plate("tbl")
```

step_lead_lag *step_lead_lag*

Description

Lag or lead a column or columns. This requires a sorted and regular time series.

Usage

```
step_lead_lag(
  .rec,
  terms,
  lag,
  n_shift = 0L,
  n_subset = 1L,
  role = "predictor",
  skip = FALSE,
  keep_original_cols = FALSE,
  ...
)
```

Arguments

.rec	the R6 recipe object.
terms	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
lag	integer vector - number of samples to lag or lead. Negative numbers indicate leading a vector.
n_shift	integer - number of values to shift the starting position when n_subset is not equal to 0. The value of n_shift has to be less than 'n_subset'.
n_subset	integer - spacing between adjacent samples in the result.
role	character - the name of the role
skip	logical - should the step be skipped
keep_original_cols	logical - keep the original columns or replace them
...	additional arguments

Value

an updated recipe

Examples

```
dat <- data.frame(x = rnorm(10), y = rnorm(10))

rec <- recipe(y~x, data = dat) |>
       step_lead_lag(x, lag = 1)

rec <- recipe(y~x, data = dat) |>
       step_lead_lag(x, lag = 1, n_subset = 5)

rec <- recipe(y~x, data = dat) |>
       step_lead_lag(x, lag = 1, n_shift = 2, n_subset = 5)
```

---

step_multiply *step_multiply*

Description

step_multiply

Usage

```
step_multiply(
  .rec,
  terms,
  values = 1,
  role = "predictor",
  skip = FALSE,
  keep_original_cols = FALSE,
  ...
)
```

Arguments

.rec	the R6 recipe object.
terms	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
values	multiply column(s) by these values
role	character - the name of the role
skip	logical - should the step be skipped
keep_original_cols	logical - keep the original columns or replace them
...	additional arguments

Value

an updated recipe

Examples

```
dat <- data.frame(x = rnorm(10), y = rnorm(10))

rec <- recipe(y~x, data = dat) |>
       step_multiply(x, value = 4)
```

`step_nls` *step_nls*

Description

Uses the Eigen C++ library fast versions to generate predictions and coefficients from a recipe.

Usage

```
step_nls(
  .rec,
  formula,
  algorithm = "lm",
  n_subset = 1L,
  n_shift = 0L,
  range = c(-Inf, Inf),
  control = gsl_nls_control(xtol = 1e-08),
  trace = FALSE,
  role = "predictor",
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`formula`	formula for the regression
`algorithm`	character string specifying the algorithm to use. The following choices are supported: • `"lm"` Levenberg-Marquardt algorithm (default). • `"lmaccel"` Levenberg-Marquardt algorithm with geodesic acceleration. Stability is controlled by the `avmax` parameter in `control`, setting `avmax` to zero is analogous to not using geodesic acceleration. • `"dogleg"` Powell's dogleg algorithm.

- `"ddogleg"` Double dogleg algorithm, an improvement over `"dogleg"` by including information about the Gauss-Newton step while the iteration is still far from the minimum.
- `"subspace2D"` 2D generalization of the dogleg algorithm. This method searches a larger subspace for a solution, it can converge more quickly than `"dogleg"` on some problems.

`n_subset` integer - spacing between adjacent samples in the result.

`n_shift` integer - number of values to shift the starting position when n_subset is not equal to 0. The value of n_shift has to be less than 'n_subset'.

`range` limit the fitting range to observations between range[1] and range[2]

`control` an optional list of control parameters to tune the least squares iterations and multistart algorithm. See `gsl_nls_control` for the available control parameters and their default values.

`trace` logical value indicating if a trace of the iteration progress should be printed. Default is `FALSE`. If `TRUE`, the residual (weighted) sum-of-squares and the current parameter estimates are printed after each iteration.

`role` character - the name of the role

`...` additional arguments

Value

an updated recipe

See Also

Other ols: `step_ols()`

Examples

```
data("kennel_2020")
kennel_2020[, datetime := as.numeric(datetime)]
formula <- as.formula(wl~.)
n_knots <- 12
deg_free <- 27
max_lag <- 1 + 720

frec = recipe(formula = formula, data = unclass(kennel_2020)) |>
  step_distributed_lag(baro, knots = hydrorecipes:::log_lags_arma(n_knots, max_lag)) |>
  step_spline_b(datetime, df = deg_free, intercept = FALSE) |>
  step_intercept() |>
  step_drop_columns(baro) |>
  step_drop_columns(datetime) |>
  step_ols(formula) |>
  prep() |>
  bake()
```

`step_normalize` *step_normalize*

Description

step_normalize

Usage

```
step_normalize(
  .rec,
  terms,
  role = "predictor",
  skip = FALSE,
  na_rm = TRUE,
  keep_original_cols = FALSE,
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`terms`	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
`role`	character - the name of the role
`skip`	logical - should the step be skipped
`na_rm`	logical - should NA values be removed from calculations
`keep_original_cols`	logical - keep the original columns or replace them
`...`	additional arguments

Value

an updated recipe

Examples

```
dat <- data.frame(x = rnorm(10), y = rnorm(10))

rec <- recipe(y~x, data = dat) |>
       step_normalize(x)
```

step_ols *step_ols*

Description

Uses the Eigen C++ library fast versions to generate predictions and coefficients from a recipe.

Usage

```
step_ols(.rec, formula, role = "predictor", do_response = TRUE, ...)
```

Arguments

.rec	the R6 recipe object.
formula	formula for the regression
role	character - the name of the role
do_response	logical calculate and return the responses?
...	additional arguments

Value

an updated recipe

See Also

Other ols: step_nls()

Examples

```
data("kennel_2020")
kennel_2020[, datetime := as.numeric(datetime)]
formula <- as.formula(wl~.)
n_knots <- 12
deg_free <- 27
max_lag <- 1 + 720

frec = recipe(formula = formula, data = unclass(kennel_2020)) |>
  step_distributed_lag(baro, knots = hydrorecipes:::log_lags_arma(n_knots, max_lag)) |>
  step_spline_b(datetime, df = deg_free, intercept = FALSE) |>
  step_intercept() |>
  step_drop_columns(baro) |>
  step_drop_columns(datetime) |>
  step_ols(formula) |>
  prep() |>
  bake()
```

step_ols_gap_fill *step_ols_gap_fill*

Description

step_ols_gap_fill

Usage

```
step_ols_gap_fill(.rec, terms, recipe, role = "predictor", ...)
```

Arguments

.rec	the R6 recipe object.
terms	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
recipe	Recipe to use for filling gaps
role	character - the name of the role
...	additional arguments

Value

an updated recipe

Examples

```
dat <- data.frame(x = rnorm(10), y = rnorm(10))
```

step_pca *step_pca*

Description

'StepPca' Does PCA for a set of columns. This currently is an in house function. Use at your own risk!

Usage

```
step_pca(
  .rec,
  terms,
  na_rm = TRUE,
  n_comp = 3,
  center = TRUE,
  scale = TRUE,
  role = "predictor",
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`terms`	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
`na_rm`	logical - should NA values be removed from calculations
`n_comp`	number of components to retain
`center`	center values before PCA
`scale`	scale values before PCA
`role`	character - the name of the role
`...`	additional arguments

Value

an updated recipe

Examples

```
set.seed(1)

formula <- as.formula(x~a+b+d+e+f+g)
rows <- 1000

dat <- data.frame(x = rnorm(rows),
                  a = rnorm(rows),
                  b = rnorm(rows),
                  d = rnorm(rows),
                  e = rnorm(rows),
                  f = rnorm(rows),
                  g = rnorm(rows))

rec  = recipe(formula = formula, data = dat) |>
  step_pca(c(x,a,b,d,e,f,g)) |>
  plate()
```

step_scale *step_scale*

Description

Adds a step to scale a data column(s)

Usage

```
step_scale(
  .rec,
  terms,
  role = "predictor",
  skip = FALSE,
  na_rm = TRUE,
  fun = collapse::fsd,
  n_sd = 1L,
  keep_original_cols = FALSE,
  ...
)
```

Arguments

.rec	the R6 recipe object.
terms	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
role	character - the name of the role
skip	logical - should the step be skipped
na_rm	logical - should NA values be removed from calculations
fun	function - the function that is applied to a list or columns of a data.frame like object.
n_sd	numeric - number of standard deviations for the scaling
keep_original_cols	logical - keep the original columns or replace them
...	additional arguments

Value

an updated recipe

Examples

```
dat <- data.frame(x = rnorm(10), y = rnorm(10))

rec <- recipe(y~x, data = dat) |>
       step_scale(x)
```

`step_slug_cbp` *step_slug_cbp*

Description

Cooper, Bredehoeft and Papadopulos, 1967 Slug test solution

Usage

```
step_slug_cbp(
  .rec,
  time,
  radius = 1,
  radius_casing = 0.15,
  radius_well = 0.15,
  specific_storage = 1e-06,
  hydraulic_conductivity = 1e-04,
  head_0 = 1,
  thickness = 1,
  n_terms = 16,
  role = "predictor",
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`time`	the time for evaluation (t)
`radius`	distance from line source or center of well
`radius_casing`	radius of casing
`radius_well`	radius of well screen
`specific_storage`	specific storage of aquifer (L/L)
`hydraulic_conductivity`	the hydraulic conductivity (L/t)
`head_0`	initial displacement
`thickness`	the aquifer thickness (L)
`n_terms`	number of terms for laplace solution inversion
`role`	character - the name of the role
`...`	additional arguments

Value

an updated recipe

References

Cooper, H.H., J.D. Bredehoeft and S.S. Papadopulos, 1967. Response of a finite-diameter well to an instantaneous charge of water, Water Resources Research, vol. 3, no. 1, pp. 263-269.

Examples

```
# check vs. CBP 1967 table 1
times    <- rep(c(1.0, 2.15, 4.64), 4) * 10^(rep(c(-3, -2, -1, 0), each = 3))
dat <- list(x = times)

formula = formula(x~.)

frec1 = recipe(formula = formula, data = dat) |>
  step_slug_cbp(
    time = x,
    radius = 1.0,
    radius_casing = 1.0,
    radius_well = 1.0,
    specific_storage = 1e-1,
    hydraulic_conductivity = 1.0,
    thickness = 1.0,
    head_0 = 1.0,
    n_terms = 12L
  ) |>
  plate("dt")
```

---

step_spline_b *step_spline_b*

Description

generates basis splines.

Usage

```
step_spline_b(
  .rec,
  terms,
  df = 0L,
  internal_knots = NULL,
  boundary_knots = NULL,
  intercept = FALSE,
  periodic = FALSE,
  degree = 3L,
```

```
  role = "predictor",
  ...
)
```

Arguments

.rec	the R6 recipe object.
terms	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
df	Degree of freedom that equals to the column number of the returned matrix. One can specify `df` rather than `knots`, then the function chooses `df - degree - as.integer(intercept)` internal knots at suitable quantiles of `x` ignoring missing values and those `x` outside of the boundary. For periodic splines, `df - as.integer(intercept)` internal knots will be chosen at suitable quantiles of `x` relative to the beginning of the cyclic intervals they belong to (see Examples) and the number of internal knots must be greater or equal to the specified `degree - 1`. If internal knots are specified via `knots`, the specified `df` will be ignored.
internal_knots	equivalent to knots from 'splines2::bSplines'
boundary_knots	equivalent to Boundary.knots from 'splines2::bSplines'
intercept	If `TRUE`, the complete basis matrix will be returned. Otherwise, the first basis will be excluded from the output.
periodic	A logical value. If `TRUE`, the periodic splines will be returned. The default value is `FALSE`.
degree	A nonnegative integer specifying the degree of the piecewise polynomial. The default value is 3 for cubic splines. Zero degree is allowed for piecewise constant basis functions.
role	character - the name of the role
...	Optional arguments that are not used.

Value

an updated recipe

Examples

```
formula <- as.formula(x~y+z)
rows <- 1e5

dat <- data.frame(x = rnorm(rows),
                  y = 1:rows,
                  z = cumsum(rnorm(rows)))
ik <- collapse::fquantile(dat$x, probs = seq(0, 1, 0.1))
bk <- ik[c(1, length(ik))]
```

```
ik <- ik[-c(1, length(ik))]

frec = recipe(formula = formula, data = dat) |>
  step_spline_b(x, df = 11L, intercept = FALSE)  |>
 plate("tbl")
```

`step_spline_n` *step_spline_n*

Description

generates basis splines.

Usage

```
step_spline_n(
  .rec,
  terms,
  df = 0L,
  internal_knots = NULL,
  boundary_knots = NULL,
  intercept = FALSE,
  periodic = FALSE,
  degree = 3L,
  role = "predictor",
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`terms`	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
`df`	Degree of freedom that equals to the column number of returned matrix. One can specify `df` rather than `knots`, then the function chooses `df - 1 - as.integer(intercept)` internal knots at suitable quantiles of `x` ignoring missing values and those `x` outside of the boundary. Thus, `df` must be greater than or equal to `2`. If internal knots are specified via `knots`, the specified `df` will be ignored.
`internal_knots`	equivalent to knots from 'splines2::bSplines'
`boundary_knots`	equivalent to Boundary.knots from 'splines2::bSplines'

`intercept`	If `TRUE`, the complete basis matrix will be returned. Otherwise, the first basis will be excluded from the output.
`periodic`	A logical value. If `TRUE`, the periodic splines will be returned. The default value is `FALSE`.
`degree`	A nonnegative integer specifying the degree of the piecewise polynomial. The default value is 3 for cubic splines. Zero degree is allowed for piecewise constant basis functions.
`role`	character - the name of the role
`...`	Optional arguments that are not used.

Value

an updated recipe

Examples

```
formula <- as.formula(x~y+z)
rows <- 1e5

dat <- data.frame(x = rnorm(rows),
                  y = 1:rows,
                  z = cumsum(rnorm(rows)))
ik <- collapse::fquantile(dat$x, probs = seq(0, 1, 0.1))
bk <- ik[c(1, length(ik))]
ik <- ik[-c(1, length(ik))]

frec = recipe(formula = formula, data = dat) |>
  step_spline_n(x, df = 11L, intercept = FALSE)  |>
 plate("tbl")
```

`step_subset_na_omit` *step_subset_na_omit*

Description

selects rows from output.

Usage

```
step_subset_na_omit(.rec, terms, role = "modify", ...)
```

Arguments

`.rec`	the R6 recipe object.
`terms`	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.

role	character - the name of the role
...	additional arguments

Value

an updated recipe

`step_subset_rows` *step_subset_rows*

Description

selects rows from output.

Usage

```
step_subset_rows(.rec, row_numbers, role = "modify", ...)
```

Arguments

.rec	the R6 recipe object.
row_numbers	integer vector of row numbers to keep.
role	character - the name of the role
...	additional arguments

Value

an updated recipe

Examples

```
dat <- data.frame(x = as.numeric(1:200), y = rnorm(200))
formula <- as.formula(y~x)

frec1 = recipe(formula = formula, data = dat) |>
  step_subset_rows(row_numbers = c(1, 5, 10)) |>
  plate("dt")
```

`step_subset_sample` *step_subset_sample*

Description

selects rows from output.

Usage

```
step_subset_sample(.rec, terms, size, role = "modify", ...)
```

Arguments

`.rec`	the R6 recipe object.
`terms`	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
`size`	number of samples.
`role`	character - the name of the role
`...`	additional arguments

Value

an updated recipe

`step_transport_fractures_heat`
step_transport_fractures_heat

Description

Sudicky and Frind 1982 solution adapted for heat. Two parallel fractures.

Usage

```
step_transport_fractures_heat(
  .rec,
  time,
  distance_fracture,
  distance_matrix,
  temperature_influent = 15,
  time_influent = 0,
  temperature_initial = 10,
  fracture_aperture = 2e-04,
```

```
  fracture_spacing = 1,
  velocity = 0.1/86400,
  thermal_conductivity_water = 0.615,
  thermal_conductivity_solids = 3.4,
  specific_heat_water = 4192,
  specific_heat_solids = 908,
  density_water = 1,
  density_solids = 2.5,
  porosity = 0.1,
  n_terms = 30L,
  role = "predictor",
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`time`	vector elapsed time (t)
`distance_fracture`	vector distance along fracture (z)
`distance_matrix`	vector distance into matrix (x)
`temperature_influent`	vector temperature history (t_in)
`time_influent`	vector time of influent values (t_in)
`temperature_initial`	double temperature (t_0)
`fracture_aperture`	double fracture aperture (2b)
`fracture_spacing`	double fracture aperture (2B)
`velocity`	double water velocity in fracture (v)
`thermal_conductivity_water`	double water thermal conductivity (_f)
`thermal_conductivity_solids`	double solids thermal conductivity (_s)
`specific_heat_water`	double specific heat of water
`specific_heat_solids`	double specific heat of solid particles
`density_water`	double density of the water (_w)
`density_solids`	double density of the solid particles (_s)
`porosity`	double matrix porosity ( )
`n_terms`	integer the number of laplace terms
`role`	character - the name of the role
`...`	additional arguments

Value

an updated recipe

See Also

Other transport: `step_transport_fractures_solute()`, `step_transport_ogata_banks()`

Examples

```
formula <- as.formula(~time+z+x)

dat <- as.data.frame(expand.grid(10^(3:8),
                         seq(0.0, 100, 1),
                         c(0.0, 0.05)))

names(dat) <- c("time", "z", "x")


frec1 = recipe(formula = formula, data = dat) |>
  step_transport_fractures_heat(time = time,
                                distance_fracture = z,
                                distance_matrix = x) |>
  plate()
```

---

step_transport_fractures_solute

step_transport_fractures_solute

Description

Sudicky and Frind 1982 solution. Two parallel fractures

Usage

```
step_transport_fractures_solute(
  .rec,
  time,
  distance_fracture,
  distance_matrix,
  concentration_influent = 1,
  time_influent = 0,
  concentration_initial = 0,
  fracture_aperture = 2e-04,
  fracture_spacing = 1,
  velocity = 0.1/86400,
  dispersivity_longitudinal = 0.1,
  diffusion = 1e-09,
```

```
  sorption_fracture = 0,
  sorption_matrix = 0,
  decay = 1e+15,
  density_bulk = 2.5,
  porosity = 0.1,
  tortuosity = 0.1,
  n_terms = 30L,
  role = "predictor",
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`time`	vector elapsed time (t)
`distance_fracture`	vector distance along fracture (z)
`distance_matrix`	vector distance into matrix (x)
`concentration_influent`	vector concentration history (c_in)
`time_influent`	vector concentration history (t_in)
`concentration_initial`	double concentration (c_0)
`fracture_aperture`	double fracture aperture (2b)
`fracture_spacing`	double fracture aperture (2B)
`velocity`	double water velocity in fracture (v)
`dispersivity_longitudinal`	double longitudinal dispersivity (_l)
`diffusion`	double free-water diffusion coefficient (D*)
`sorption_fracture`	double fracture distribution coefficient (K_f)
`sorption_matrix`	double matrix distribution coefficient (K_m)
`decay`	double radioactive half-life for solute ( )
`density_bulk`	double dry bulk density (_b)
`porosity`	double porosity ( )
`tortuosity`	double tortuosity ( )
`n_terms`	integer number of terms for laplace inversion
`role`	character - the name of the role
`...`	additional arguments

Value

an updated recipe

References

Sudicky, E.A., Frind, E.O., Contaminant transport in fractured porous media: Analytical solutions for a system of parallel fractures, December 1982 https://doi.org/10.1029/WR018i006p01634

See Also

Other transport: `step_transport_fractures_heat()`, `step_transport_ogata_banks()`

Examples

```
formula <- as.formula(~time+z+x)

dat <- as.data.frame(expand.grid(10^(3:8),
                         seq(0.0, 10, 1),
                         c(0.0)))

names(dat) <- c("time", "z", "x")

frec1 = recipe(formula = formula, data = dat) |>
  step_transport_fractures_solute(time = time,
                                  distance_fracture = z,
                                  distance_matrix = x) |>
  plate()
```

---

step_transport_ogata_banks

step_transport_ogata_banks

Description

Ogata, A., Banks, R.B., 1961. A solution of the differential equation of longitudinal dispersion in porous media. U. S. Geol. Surv. Prof. Pap. 411-A. 1-D, infinite source, uniform flow, constant parameters, decay, retardation

To have values match the excel sheet https://www.civil.uwaterloo.ca/jrcraig/pdf/OgataBanks.xlsm the decay coefficient needs to be scaled by the retardation coefficient.

Care must be taken so that input values do not lead to NaN. -Need to fix this.

1-D infinite source uniform flow constant parameters no decay no retardation

Usage

```
step_transport_ogata_banks(
  .rec,
  time,
  distance,
  concentration_initial = 1,
  velocity = 0.1,
  diffusion = 0.1,
  retardation = 1,
  decay = 0,
  role = "predictor",
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`time`	vector time
`distance`	vector x position
`concentration_initial`	double concentration
`velocity`	double velocity
`diffusion`	double diffusion coefficient
`retardation`	double retardation coefficient
`decay`	double decay coefficient
`role`	character - the name of the role
`...`	additional arguments

Value

Ogata-Banks solution for time and distance pairs

References

Ogata, A., Banks, R.B., 1961. A solution of the differential equation of longitudinal dispersion in porous media. U. S. Geol. Surv. Prof. Pap. 411-A. 1-D, infinite source, uniform flow, constant parameters, decay, retardation

See Also

Other transport: `step_transport_fractures_heat()`, `step_transport_fractures_solute()`

Examples

```
formula <- as.formula(y~x)
rows <- 100

dat <- data.frame(expand.grid(as.numeric(1:rows), as.numeric(1:10)))
names(dat) <- c('x', 'y')
frec1 = recipe(formula = formula, data = dat) |>
  step_transport_ogata_banks(time = x, distance = y) |>
  plate("dt")
```

---

step_vadose_weeks *step_vadose_weeks*

Description

Weeks 1979 solution

Usage

```
step_vadose_weeks(
  .rec,
  time,
  air_diffusivity = 0.2,
  thickness = 40,
  precision = 1e-12,
  inverse = FALSE,
  role = "predictor",
  ...
)
```

Arguments

`.rec`	the R6 recipe object.
`time`	vector time elapsed time from start of pressure change
`air_diffusivity`	double vadose zone air diffusivity
`thickness`	double vadose zone thickness
`precision`	double stop the sum when precision is reached
`inverse`	double whether the response is invers
`role`	character - the name of the role
`...`	additional arguments

Value

an updated recipe

References

Weeks, E.P., 1979. Barometric fluctuations in wells tapping deep unconfined aquifers. Water Resources Research, 15(5), pp.1167-1176.

Examples

```
formula <- as.formula(y~x)

n <- 100
dat <- data.frame(x = as.numeric(1:n),
                  y = as.numeric(1:n))

frec1 = recipe(formula = formula, data = dat) |>
  step_vadose_weeks(time = x,
                    air_diffusivity = 0.8,
                    thickness = 5,
                    precision = 1e-12) |>
  plate()
```

`step_varying` *step_varying*

Description

remove columns that only contain a single value.

Usage

```
step_varying(.rec, terms, role = "predictor", ...)
```

Arguments

`.rec`	the R6 recipe object.
`terms`	the unquoted names of the variables to use or a selector function. terms replaces the '...' of the recipes package but requires variables to be included within 'c()'. For example to include variables x and y you would write 'c(x,y)' in the hydrorecipes package.
`role`	character - the name of the role
`...`	additional arguments

Value

an updated recipe

Examples

```
formula <- as.formula(y~x+z)
rows <- 1000
dat <- data.frame(x = rep(1, rows),
                  y = 1:rows,
                  z = rnorm(rows))

frec = recipe(formula = formula, data = dat) |>
  step_varying(c(x, y, z)) |>
  plate()
```

`tidal_cooper_1965` *tidal_cooper_1965*

Description

Cooper Jr, H.H., Bredehoeft, J.D., Papadopulos, I.S. and Bennett, R.R., 1965. The response of well-aquifer systems to seismic waves. Journal of Geophysical Research, 70(16), pp.3915-3926.

Usage

```
tidal_cooper_1965(
  frequency,
  storativity,
  transmissivity,
  thickness_aquifer,
  height_water,
  radius_well,
  radius_casing = radius_well,
  gravity = 9.80665
)
```

Arguments

`frequency`	the frequency of the response
`storativity`	layer storativity (L/L)
`transmissivity`	aquifer transmissivity (L*L/t)
`thickness_aquifer`	aquifer thickness
`height_water`	height of water in well
`radius_well`	well radius
`radius_casing`	casing radius
`gravity`	gravitational acceleration at well

Value

tidal response

Examples

```
data('hsieh_1987_fig_2_3')
storativity <- 1e-07
transmissivity <- 1e-03
radius_well <- 0.05
frequency <- 10^seq(-5, 2, by = 0.1)
tau   <- 1 / frequency
cooper <- tidal_cooper_1965(frequency,
                            storativity,
                            transmissivity,
                            thickness_aquifer = 1,
                            height_water = 1,
                            radius_well)
plot(Mod(response)~dimensionless_frequency, cooper,
 type='l',
 log = 'x',
 xlim = c(1, 1000))
points(response~dimensionless_frequency,
  hsieh_1987_fig_2_3[variable=='gain' & S == storativity])

plot(unwrap(Arg(response)) * 180/pi~dimensionless_frequency, cooper,
 type='l',
 log = 'x',
 xlim = c(1, 1000),
 ylim = c(0, -90))
points(response~dimensionless_frequency,
  hsieh_1987_fig_2_3[variable=='phase' & S == storativity])
```

---

tidal_hsieh_1987 *tidal_hsieh_1987 Solution for estimating transmissivity and storativity from earth tides.*

Description

Hsieh, P.A., Bredehoeft, J.D. and Farr, J.M., 1987. Determination of aquifer transmissivity from Earth tide analysis. Water resources research, 23(10), pp.1824-1832.

Usage

```
tidal_hsieh_1987(
  frequency,
  storativity,
  transmissivity,
  radius_well,
```

```
  radius_casing = radius_well
)
```

Arguments

`frequency`	the frequency of the response
`storativity`	layer storativity (L/L)
`transmissivity`	aquifer transmissivity (L*L/t)
`radius_well`	well radius
`radius_casing`	casing radius

Value

tidal response

Examples

```
data('hsieh_1987_fig_2_3')
storativity <- 1e-07
transmissivity <- 1e-03
radius_well <- 0.05
frequency <- 10^seq(-5, 2, by = 0.05)
tau   <- 1 / frequency
hsieh <- tidal_hsieh_1987(frequency, storativity, transmissivity, radius_well)
plot(Mod(response)~dimensionless_frequency, hsieh,
 type='l',
 log = 'x',
 xlim = c(1, 1000))
points(response~dimensionless_frequency, hsieh_1987_fig_2_3[variable=='gain' & S == storativity])

plot(unwrap(Arg(response)) * 180/pi~dimensionless_frequency, hsieh,
 type='l',
 log = 'x',
 xlim = c(1, 1000),
 ylim = c(0, -90))
points(response~dimensionless_frequency, hsieh_1987_fig_2_3[variable=='phase' & S == storativity])
```

---

unwrap — *unwrap Removes the large phase shifts that can be associated with using Arg or atan2.*

Description

unwrap Removes the large phase shifts that can be associated with using Arg or atan2.

Usage

```
unwrap(phase)
```

Arguments

phase	the phase in radians

Value

unwrapped vector

---

vadose_response *vadose_response*

Description

weeks_1979 1-D air diffusivity

Usage

```
vadose_response(time, air_diffusivity, thickness, precision, inverse)
```

Arguments

time	numeric vector of elapsed times
air_diffusivity	A numeric value of the unsaturated zone air diffusivity
thickness	A numeric value of the unsaturated zone thickness
precision	A numeric value of for the solution precision
inverse	A logical value indicating if an inverse water level relationship is desired

Value

weeks 1979 model

Examples

```
vr <- vadose_response(time = 0:43200,
                      air_diffusivity = 0.20,
                      thickness = 40,
                      precision = 1e-10,
                      inverse = FALSE)
```

`window_blackman_harris` *window_blackman_harris*

Description

Blackman-Harris window for FFT

Usage

```
window_blackman_harris(n)
```

Arguments

`n`	length of the window vector (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 the window vector (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 the window vector (integer)
`a0`	`double` coefficient
`a1`	`double` coefficient
`a2`	`double` coefficient
`a3`	`double` coefficient

Value

window

Examples

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

`window_hann` *window_hann*

Description

Hann window for FFT.

Usage

```
window_hann(n)
```

Arguments

`n`	length of the window vector (integer)

Value

window of length n.

`window_hann_cplx` *window_hann_cplx*

Description

Hann window for complex FFT.

Usage

```
window_hann_cplx(n)
```

Arguments

`n`	length of the window vector (integer)

Value

window of length n.

`window_nuttall` *window_nuttall*

Description

Nuttall window for FFT

Usage

```
window_nuttall(n)
```

Arguments

`n`	length of the window vector (integer)

Value

window

Examples

```
window_nuttall(100)
```

`window_rectangle` *window_rectangle*

Description

Rectangular window.

Usage

```
window_rectangle(n)
```

Arguments

`n`	length of the window vector (integer)

Value

window of length n.

`window_scale` *window_scale*

Description

Scale factor for a window function.

Usage

```
window_scale(window, n_new, n_fft)
```

Arguments

`window`	the window function (numeric vector)
`n_new`	length of the padded series (integer)
`n_fft`	length of the input series (integer)

Value

window of length n.

`window_tukey` *window_tukey*

Description

Tukey window for FFT.

Usage

```
window_tukey(n, r)
```

Arguments

`n`	length of the window vector (integer)
`r`	percent on each side to taper (double)

Value

window of length n.

Index