Package 'WRTDStidal'

June 23, 2017

Type Package

Title Weighted Regression for Water Quality Evaluation in Tidal Waters

Version 1.1.0

Date 2017-06-23

Author Marcus W. Beck [aut, cre]

Maintainer Marcus W. Beck <mbafs2012@gmail.com>

Description An adaptation for estuaries (tidal waters) of weighted regression on time, discharge, and season to evaluate trends in water quality time series.

BugReports https://github.com/fawda123/wtreg_for_estuaries/issues

License CC0

LazyData true

Imports caret, dplyr, EnvStats, fields, foreach, forecast, gridExtra, lubridate, purrr, quantreg, RColorBrewer, survival, tidyr

Suggests doParallel, grDevices, magrittr

Depends R (>= 3.1.2), ggplot2

RoxygenNote 6.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2017-06-23 20:24:12 UTC

R topics documented:

aiccrq *Akaike's Information Criterion for weighted quantile regression*

Description

Get AIC values for a single weighted quantile regression as used in WRTDS models

Usage

```
aiccrq(mod_in, tau = 0.5)
```

Arguments

mod_in	input crq model
tau	numeric indicating quantile to evaluate

Details

The AIC value is based on the log-likelihood estimate of the model that accounts for the specific quantile, the minimum of the objective function (rho), and the number of model parameters. The residuals are specific to the WRTDS model such that this function cannot be applied to arbitrary crq models.

Value

AIC estimate

Examples

```
# get wts for a model centered on the first observation
ref_in <- tidobj[1, ]
ref_wts <- getwts(tidobj, ref_in)

# get the model
mod <- quantreg::crq(
   survival::Surv(res, not_cens, type = "left") ~
     dec_time + flo + sin(2*pi*dec_time) + cos(2*pi*dec_time),
   weights = ref_wts,
   data = tidobj,
   method = "Portnoy"
   )

aiccrq(mod)
```

all_sims *Simulate a response variable time series using all functions*

Description

Simulate a response variable time series using all simulation functions

Usage

```
all_sims(dat_in, ...)
```

Arguments

dat_in	input data.frame that must include discharge and decimal time columns, see example dataset daydat
...	additional arguments passed to lnres_sim

Details

This is a convenience function that combines lnQ_sim, lnres_err, and lnres_sim. See the help documentation function for more details on each.

Value

Original data frame with additional columns for simulated discharge (lnQ_sim), the random errors of the response variable (errs), the standard error estimates for each residual (scls), a flow-independent response variable time series (lnres_noQ), and a flow-dependent time series (lnres_Q).

See Also

daydat for example data, lnQ_sim for simulating discharge, lnres_err for estimating the error structure of the response variable, and lnres_sim for simulating the response variable

Examples

```
## Not run:
## example data
data(daydat)

## simulate
tmp <- all_sims(daydat)

## End(Not run)
```

annual_agg *Create annual aggregations of WRTDS output*

Description

Create annual aggregations of WRTDS output

Usage

```
annual_agg(dat_in, ...)

## Default S3 method:
annual_agg(dat_in, mo_strt = 10, min_mo = 9,
  logspace = TRUE, ...)
```

Arguments

dat_in	input tidal or tidalmean object
...	arguments passed to or from other methods
mo_strt	numeric indicating month to start aggregation years, defaults to October for USGS water year from October to September
min_mo	numeric value from one to twelve indicating the minimum number of months with observations for averaging by years
logspace	logical indicating if aggregated data are to be shown in log-space or not

Details

WRTDS output is averaged by year for both predictions and flow-normalized predictions. Years are averaged only if one observation is contained in each of the minimum number of months specified by min_mo averaging, otherwise results are not returned for the given year. Note that setting min_mo to values smaller than the default can produce inaccurate trends for years with very few results.

The function is used internally within prdnrmplot and fitplot.

Value

An aggregated data object for plotting, returns only model output and response variable.

Examples

```
## tidal object
annual_agg(tidfit)

## tidalmean object
annual_agg(tidfitmean)
```

chldat *Monthly chlorophyll time series for Hillsborough Bay*

Description

Monthly chlorophyll time series for the Hillsborough Bay segment of Tampa Bay. Data are the median values of monthly observations across all water quality stations in Hillsborough Bay. Date ranges are from January 1974 to December 2012 (452 observations). Variables are date, chlorophyll-a (in log-space, labelled `res` for response), salinity as fraction of freshwater (i.e., 0 - 1, with higher values indicating more freshwater, label `flo` for flow), and the detection limit for all stations for the respective date.

Usage

```
chldat
```

Format

A data frame with 452 rows and 4 variables:

`date` Date

`res` numeric

`flo` numeric

`lim` numeric

chllab *Chlorophyll axis label*

Description

Get the chlorophyll axis label for observed or log-space, including units

Usage

```
chllab(logspace = TRUE)
```

Arguments

`logspace`	logical indicating if chlorophyll is in log space, otherwise observed

Value

An expression indicating (log)-chlorophyll in the appropriate units

Examples

```
## default
chllab()
```

`chngest` *Get trend for a single time period*

Description

Get trend for a single time period

Usage

```
chngest(x, y, single = F)
```

Arguments

`x`	numeric of year values
`y`	norm numeric of annually averaged response variable
`single`	logical if trends are based on only first and last year of aggregated, i.e., no averaging of first/last three

Details

Estimates trends using methods described in `wrtdstrnd`. Used internally and is not to be called by the user. Function runs on individual time period groups defined by arguments in `wrtdstrnd`.

`createsrch` *Create a grid of half-window widths to evaluate*

Description

Create a grid of all unique combinations of half-window widths to evaluate. The result can be passed to `winsrch_grid`.

Usage

```
createsrch(mos = c(seq(0.5, 1, by = 0.25), 2, 10), yrs = c(seq(5, 15, by =
  3), 50), flo = c(seq(0.5, 1, by = 0.1), 5))
```

Arguments

mos	numeric vector of half-window widths for months, a value of one indicates twelve months
yrs	numeric vector of half-window widths for years, a value of one indicates one-year
flo	numeric vector of half-window widths for salinity or flow, a value of one indicates the full range of values (100 percent)

Details

The weighting function uses a tri-cube weighting scheme such that weights diminish with distance from the center of the window. For example, a value of one for the month window does not mean that all months are weighted equally even though the window covers an entire calendar year.

Value

A matrix with number of rows equal to the product of the lengths of each input vector, where each row is a unique combination for the selected half-window widths.

See Also

winsrch_grid

Examples

```
createsrch()
createsrch(1, 1, 1)
```

---

daydat *Daily chlorophyll, salinity, and discharge time series for the Upper Patuxent River Estuary*

Description

Combined daily flow observations from the USGS stream gage station 01594440 near Bowie, Maryland and daily chlorophyll and salinity records from the Jug Bay station of the Chesapeake Bay Maryland National Estuarine Research Reserve. Daily chlorophyll concentrations were estimated from fluorescence values that did not include blue-green algae blooms. These date are provided to illustrate examples with time series simulation.

Usage

```
daydat
```

Format

A data frame with 3506 rows and 9 variables, 1985 to 2014:

`date` Date as daily time step

`flo` numeric for salinity, ppt

`lnres` numeric for chlorophyll fluorescence as ln-transformed ug/l

`Q` numeric for daily discharge m3/s

`lnQ` numeric for daily discharge ln-transformed m3/s

`jday` numeric for julian day

`year` numeric for year

`day` numeric for day from 1-31

`dec_time` numeric for decimal time on the annual period

`dec_time` *Create decimal time from time vector*

Description

Create decimal time on an annual scale from an input time vector

Usage

```
dec_time(date_in)

## S3 method for class 'Date'
dec_time(date_in)
```

Arguments

`date_in`	input time vector, usually a `date` object

Details

Function is used internally within the package.

Value

A named list of four numeric vectors including `day_num` (decimal day on an annual scale), `month` (month of the year as integer), `year`, and `dec_time` (decimal time as sum of `year` and `day_num`)

Examples

```
dt <- Sys.Date()
dts <- seq.Date(dt - 365, dt, by = 'day')

dec_time(dts)
```

dynaplot *Plot model response to salinity or flow as a lineplot for all months*

Description

Plot the relationship between the modelled response and salinity/flow across the time series using line plots for each month. Each line corresponds to a unique year. This can be used to evaluate temporal variation between the two.

Usage

```
dynaplot(dat_in, ...)

## S3 method for class 'tidal'
dynaplot(dat_in, month = c(1:12), tau = NULL,
  years = NULL, col_vec = NULL, alpha = 1, size = 1, logspace = TRUE,
  floscl = TRUE, allflo = FALSE, ncol = NULL, grids = TRUE,
  scales = NULL, pretty = TRUE, use_bw = TRUE, fac_nms = NULL, ...)

## S3 method for class 'tidalmean'
dynaplot(dat_in, month = c(1:12), years = NULL,
  col_vec = NULL, alpha = 1, size = 1, logspace = TRUE, floscl = TRUE,
  allflo = FALSE, ncol = NULL, grids = TRUE, scales = NULL,
  pretty = TRUE, use_bw = TRUE, fac_nms = NULL, ...)
```

Arguments

dat_in	input tidal or tidalmean object
...	arguments passed to other methods
month	numeric input from 1 to 12 indicating the monthly predictions to plot
tau	numeric vector of quantile to plot. The function will plot the 'middle' quantile if none is specified, e.g., if 0.2, 0.3, and 0.4 are present in the fitted model object then 0.3 will be plotted.
years	numeric vector of years to plot, one to many, defaults to all
col_vec	chr string of plot colors to use, passed to gradcols and scale_colour_gradientn for line shading. Any color palette from RColorBrewer can be used as a named input. Palettes from grDevices must be supplied as the returned string of colors for each palette.
alpha	numeric value from zero to one indicating line transparency
size	numeric value for line size
logspace	logical indicating if plots are in log space
floscl	logical indicating if salinity/flow on x-axis is standardized (default) or in original scale

allflo	logical indicating if the salinity or flow values for plotting are limited to the fifth and ninety-fifth percentile of observed values for the month of interest
ncol	numeric argument passed to facet_wrap indicating number of facet columns
grids	logical indicating if grid lines are present
scales	chr string passed to ggplot to change x/y axis scaling on facets, acceptable values are 'free', 'free_x', or 'free_y'
pretty	logical indicating if my subjective idea of plot aesthetics is applied, otherwise the ggplot default themes are used. The aesthetic arguments will not apply if pretty = TRUE.
use_bw	logical indicating if the theme_bw theme is used
fac_nms	optional chr string for facet labels, which must be equal in length to month

Details

These plots can be used to examine how the relationship between the response variable and flow varies throughout the time series. It is essentially identical to the plot produced by gridplot, except lines plots are returned that show the relationship of the response variable with salinity/flow using different lines for each year. The interpolation grid that is stored as an attribute in a fitted tidal object is used to create the plot. Each plot is limited to the same month throughout the time series to limit seasonal variation. Plots are also constrained to the fifth and ninety-fifth percentile of observed salinity/flow values during the month of interest to limit the predictions within the data domain. This behavior can be suppressed by changing the allflo argument.

Note that the year variable used for color mapping is treated as a continuous variable although it is an integer by definition.

Value

A ggplot object that can be further modified

See Also

fitplot, gridplot, prdnrmplot

Examples

```
# load a fitted tidal object
data(tidfit)

# plot using defaults,
# defaults to the fiftieth quantile for all years
dynaplot(tidfit)
## Not run:
# change the defaults
dynaplot(tidfit, tau = 0.9, month = 2, years = seq(1980, 1990),
 col_vec = rainbow(7), alpha = 0.5, size = 3)

# plot a tidalmean object
```

```
data(tidfitmean)

dynaplot(tidfitmean)

## End(Not run)
```

`fillpo` *Fill the* `predonobs` *attribute*

Description

Fill the `predonobs` attribute

Usage

```
fillpo(dat_in, ...)

## S3 method for class 'tidal'
fillpo(dat_in, ...)

## S3 method for class 'tidalmean'
fillpo(dat_in, ...)
```

Arguments

`dat_in`	input tidal or tidalmean object
`...`	arguments passed to or from other methods

Details

Used in `respred` to fill the `predonobs` attribute. This attribute is used to estimate performance metrics with `wrtdsperf`.

Value

The input tidal or tidalmean object with the filled `predonobs` attribute as predictions for the observed data as a data frame.

fill_grd *Add date columns and fill missing values in the interpolation grids*

Description

Add date, year, month, and day columns to the interpolation grids using dates from `dat_in`. The `interp.surface` function is used after splitting the interpolation matrix by month to fill missing values. Function is used in `wrtds`.

Usage

```
fill_grd(grd_in, dat_in, interp = FALSE)
```

Arguments

`grd_in`	interpolation grid to fill, either a single mean grid or an individual quantile grid created within `wrtds`
`dat_in`	`tidal` or `tidalmean` object
`interp`	logical for interpolation

fitmoplot *Plot the fitted results for a tidal object by month*

Description

Plot a tidal object to view the response variable observations, predictions, and normalized results separately for each month.

Usage

```
fitmoplot(dat_in, ...)

## S3 method for class 'tidal'
fitmoplot(dat_in, month = c(1:12), tau = NULL,
  predicted = TRUE, logspace = TRUE, dt_rng = NULL, ncol = NULL,
  col_vec = NULL, grids = TRUE, pretty = TRUE, lwd = 1, size = 2,
  alpha = 1, ...)

## S3 method for class 'tidalmean'
fitmoplot(dat_in, month = c(1:12), predicted = TRUE,
  logspace = TRUE, dt_rng = NULL, ncol = NULL, col_vec = NULL,
  grids = TRUE, pretty = TRUE, lwd = 1, size = 2, alpha = 1, ...)
```

Arguments

dat_in	input tidal or tidalmean object
...	arguments passed to other methods
month	numeric indicating months to plot
tau	numeric vector of quantiles to plot, defaults to all in object if not supplied
predicted	logical indicating if standard predicted values are plotted, default TRUE, otherwise normalized predictions are plotted
logspace	logical indicating if plots are in log space
dt_rng	Optional chr string indicating the date range of the plot. Must be two values in the format 'YYYY-mm-dd' which is passed to as.Date.
ncol	numeric argument passed to facet_wrap indicating number of facet columns
col_vec	chr string of plot colors to use, passed to gradcols. Any color palette from RColorBrewer can be used as a named input. Palettes from grDevices must be supplied as the returned string of colors for each palette.
grids	logical indicating if grid lines are present
pretty	logical indicating if my subjective idea of plot aesthetics is applied, otherwise the ggplot default themes are used
lwd	numeric value indicating width of lines
size	numeric value indicating size of points
alpha	numeric value indicating transparency of points or lines

Details

The plots are similar to those produced by fitplot except the values are faceted by month. This allows an evaluation of trends over time independent of seasonal variation. Multiple observations within each month for each year are averaged for a smoother plot.

Value

A ggplot object that can be further modified

See Also

fitplot, prdnrmplot, sliceplot

Examples

```
## load a fitted tidal object
data(tidfit)

# plot using defaults
fitmoplot(tidfit)
## Not run:
# get the same plot but use default ggplot settings
```

```
fitmoplot(tidfit, pretty = FALSE)

# plot specific quantiles
fitmoplot(tidfit, tau = c(0.1, 0.9))

# plot the normalized predictions
fitmoplot(tidfit, predicted = FALSE)

# modify the plot as needed using ggplot scales, etc.

library(ggplot2)

fitmoplot(tidfit, pretty = FALSE, linetype = 'dashed') +
 theme_classic() +
 scale_y_continuous(
   'Chlorophyll',
   limits = c(0, 5)
   ) +
 scale_colour_manual(
   'Predictions',
   labels = c('lo', 'md', 'hi'),
   values = c('red', 'green', 'blue'),
   guide = guide_legend(reverse = TRUE)
   )

# plot a tidalmean object
data(tidfitmean)

fitmoplot(tidfitmean)

## End(Not run)
```

`fitplot` *Plot the fitted results for a tidal object*

Description

Plot a tidal object to view response variable observations, predictions, and normalized results.

Usage

```
fitplot(dat_in, ...)

## S3 method for class 'tidal'
fitplot(dat_in, tau = NULL, predicted = TRUE,
  annuals = TRUE, logspace = TRUE, dt_rng = NULL, col_vec = NULL,
  grids = TRUE, min_mo = 9, mo_strt = 10, pretty = TRUE, lwd = 1,
  size = 2, alpha = 1, ...)
```

```
## S3 method for class 'tidalmean'
fitplot(dat_in, predicted = TRUE, annuals = TRUE,
  logspace = TRUE, dt_rng = NULL, col_vec = NULL, grids = TRUE,
  min_mo = 9, mo_strt = 10, pretty = TRUE, lwd = 1, size = 2,
  alpha = 1, ...)
```

Arguments

dat_in	input tidal or tidalmean object
...	arguments passed to other methods
tau	numeric vector of quantiles to plot, defaults to all in object if not supplied
predicted	logical indicating if standard predicted values are plotted, default TRUE, otherwise normalized predictions are plotted
annuals	logical indicating if plots are annual aggregations of results
logspace	logical indicating if plots are in log space
dt_rng	Optional chr string indicating the date range of the plot. Must be two values in the format 'YYYY-mm-dd' which is passed to as.Date.
col_vec	chr string of plot colors to use, passed to gradcols. Any color palette from RColorBrewer can be used as a named input. Palettes from grDevices must be supplied as the returned string of colors for each palette.
grids	logical indicating if grid lines are present
min_mo	numeric value from one to twelve indicating the minimum number of months with observations for averaging by years, applies only if annuals = TRUE. See annual_agg.
mo_strt	numeric indicating month to start aggregation years, defaults to October for USGS water year from October to September, applies only if annuals = TRUE. See annual_agg.
pretty	logical indicating if my subjective idea of plot aesthetics is applied, otherwise the ggplot default themes are used
lwd	numeric value indicating width of lines
size	numeric value indicating size of points
alpha	numeric value indicating transparency of points or lines

Value

A ggplot object that can be further modified

See Also

fitmoplot, prdnrmplot, sliceplot

Examples

```
## load a fitted tidal object
data(tidfit)

# plot using defaults
fitplot(tidfit)

# get the same plot but use default ggplot settings
fitplot(tidfit, pretty = FALSE)

# plot in log space
fitplot(tidfit, logspace = TRUE)

# plot specific quantiles
fitplot(tidfit, tau = c(0.1, 0.9))

# plot the normalized predictions
fitplot(tidfit, predicted = FALSE)

# plot as monthly values
fitplot(tidfit, annuals = FALSE)

# format the x-axis is using annual aggregations
library(ggplot2)

fitplot(tidfit, annual = TRUE) +
 scale_x_date(limits = as.Date(c('2000-01-01', '2012-01-01')))

# modify the plot as needed using ggplot scales, etc.

fitplot(tidfit, pretty = FALSE, linetype = 'dashed') +
 theme_classic() +
 scale_y_continuous(
   'Chlorophyll',
   limits = c(0, 50)
   ) +
 scale_colour_manual(
   'Predictions',
   labels = c('lo', 'md', 'hi'),
   values = c('red', 'green', 'blue'),
   guide = guide_legend(reverse = TRUE)
   )

# plot a tidalmean object
data(tidfitmean)

fitplot(tidfitmean)
```

`getwts` *Get weights for regression*

Description

Get weights for WRTDS for a single observation using a tri-cubic weighting function

Usage

```
getwts(dat_in, ...)

## Default S3 method:
getwts(dat_in, ref_in, wt_vars = c("day_num", "dec_time",
  "flo"), wins = list(0.5, 10, NULL), all = FALSE, slice = TRUE,
  ngrzero = FALSE, wins_only = FALSE, min_obs = 100, ...)
```

Arguments

dat_in	input tidal object
...	arguments passed to or from other methods
ref_in	row of tidal object as reference for weights
wt_vars	chr string of three elements indicating names of columns in tidal object that are used for reference row weights
wins	list of half-window widths for time, year, and flow
all	logical to return individual weights rather than the product of all three, default FALSE
slice	logical indicating if data are subset by observations within the maximum window width for faster calculations
ngrzero	logical indicating if count of observations with weights greater than zero is returned
wins_only	logical if the half-window widths should be returned as a list
min_obs	numeric vector for window widening if the number of observations with non-zero weights is less than the specified value, use min_obs = NULL to suppress this behavior

Details

The default half-window widths for day_num, year, and flow are half a day (12 hours), 10 years, and half the range of salinity/flow in the input data. The half-window widths are expanded by 10% until at least 100 observations have weights greater than zero. This behavior can be suppressed by setting min_obs = NULL.

Value

A vector of weights with length equal to the number of observations (rows) in the tidal object. Vectors for all three weighting variables are returned if all = TRUE.

Examples

```
##
data(tidobj)

# get weights for first row
first <- tidobj[1, ]
wts <- getwts(tidobj, first)

plot(wts, type = 'l')

## Not run:

# get count of observations with grzero weights
sapply(1:nrow(tidobj), function(row) getwts(tidobj, tidobj[row, ],
 ngrzero = TRUE))

## End(Not run)
```

---

goodfit *Quantile regression goodness of fit*

Description

Calculate quantile regression goodness of fit using residuals and non-conditional residuals

Usage

```
goodfit(resid, resid_nl, tau)
```

Arguments

resid	numeric vector of residuals from the conditional quantile model
resid_nl	numeric vector of residuals from the non-conditional (null) quantile model
tau	numeric value from zero to one for the estimated quantile

Details

The goodness of fit measure for quantile regression is estimated as 1 minus the ratio between the sum of absolute deviations in the fully parameterized models and the sum of absolute deviations in the null (non-conditional) quantile model. The values are useful for comparisons between quantile models, but they are not comparable to standard coefficients of determination. The latter is based on the variance of squared deviations, whereas goodness of fit values for quantile regression are based on absolute deviations. Goodness of fit values will always be smaller than R2 values.

Value

A numeric value from 0 to 1 indicating goodness of fit

References

Koenker, R., Machado, J.A.F. 1999. Goodness of fit and related inference processes for quantile regression. Journal of the American Statistical Association. 94(448):1296-1310.

See Also

wrtdsrsd for residuals

Examples

```
library(quantreg)

## random variables
x <- runif(100, 0, 10)
y <- x + rnorm(100)

## quantile model
mod <- rq(y ~ x, tau = 0.5)
res <- resid(mod)

## non-conditional quantile model
mod_nl <- rq(y ~ 1, tau = 0.5)
rsd_nl <- resid(mod_nl)

goodfit(res, rsd_nl, 0.5)

## r2 of mean model for comparison
mod_lm <- lm(y ~ x)

summary(mod_lm)$r.squared
```

---

gradcols *Get colors for plots*

Description

Gets colors used for WRTDS plots

Usage

```
gradcols(col_vec = NULL)
```

Arguments

col_vec	chr string of plot colors to use, typically passed to scale_colour_gradientn for shading. Any color palette from RColorBrewer can be used as a named input. Palettes from grDevices must be supplied as the returned string of colors for each palette.

Details

This is a convenience function for retrieving a color palette that is used by most of the plotting functions. Palettes from RColorBrewer will use the maximum number of colors. The default palette is 'Spectral'.

Value

A character vector of colors in hexadecimal notation.

See Also

dynaplot, gridplot, wtsplot

Examples

```
## defaults
gradcols()

## another RColorBrewer palette
gradcols('Pastel2')

## a silly example
gradcols(rainbow(7))
```

`gridplot` *Plot variable response to salinity/flow as a gridded surface for all months*

Description

Plot the relationship between the response variable and salinity/flow across the time series using a gridded surface for all months. The response is shaded by relative values across all dates for comparison.

Usage

```
gridplot(dat_in, ...)

## S3 method for class 'tidal'
gridplot(dat_in, month = c(1:12), tau = NULL,
  years = NULL, col_vec = NULL, col_lim = NULL, logspace = TRUE,
  floscl = TRUE, allflo = FALSE, flo_fac = 3, yr_fac = 3, ncol = NULL,
  grids = FALSE, pretty = TRUE, ...)

## S3 method for class 'tidalmean'
gridplot(dat_in, month = c(1:12), years = NULL,
  col_vec = NULL, col_lim = NULL, logspace = TRUE, floscl = TRUE,
```

```
allflo = FALSE, flo_fac = 3, yr_fac = 3, ncol = NULL, grids = FALSE,
pretty = TRUE, ...)
```

Arguments

dat_in	input tidal or tidalmean object
...	arguments passed to other methods
month	numeric indicating months to plot or chr string 'all' to indicate all months with no plot facets
tau	numeric vector of quantile to plot. The function will plot the 'middle' quantile if none is specified, e.g., if 0.2, 0.3, and 0.4 are present in the fitted model object then 0.3 will be plotted.
years	numeric vector for range of years to plot
col_vec	chr string of plot colors to use, passed to gradcols and scale_fill_gradientn for grid shading. Any color palette from RColorBrewer can be used as a named input. Palettes from grDevices must be supplied as the returned string of colors for each palette.
col_lim	numeric vector of length two that defines the range of the color ramp in the legend, passed to scale_fill_gradient. This is useful for fixing the color range to evaluate multiple plots.
logspace	logical indicating if plots are in log space
floscl	logical indicating if salinity/flow on x-axis is standardized (default) or in original scale
allflo	logical indicating if the salinity/flow values for plotting are limited to the fifth and ninety-fifth percentile of observed values for the month of interest
flo_fac	numeric value indicating the factor for smoothing the response variable across salinity/flow values. Increasing the value creates more smoothing and setting the value to 1 removes all smoothing.
yr_fac	numeric value indicating the factor for smoothing the response variable across integer years. Increasing the value creates more smoothing and setting the value to 1 removes all smoothing.
ncol	numeric argument passed to facet_wrap indicating number of facet columns
grids	logical indicating if grid lines are present
pretty	logical indicating if my subjective idea of plot aesthetics is applied, otherwise the ggplot default themes are used

Details

These plots can be used to examine how the relationship between the response variable and salinity/flow varies throughout the time series for multiple months. The plot is similar to that returned by dynaplot except changes in the response are shown on a gridded surface of salinity/flow versus time. Multiple months can also be viewed. Color shading is in proportion to the value of the response variable and is relative across the plotted months. The interpolation grid that is stored as an attribute in a fitted tidal object is used to create the plot. By default, the plots are constrained to the fifth and ninety-fifth percentile of observed salinity/flow values during each month to limit

the predictions within the data domain. This behavior can be suppressed by changing the `allflo` argument, although the predicted values of the response variable that are outside of the salinity/flow range for the plotted month are typically unrealistic.

Value

A `ggplot` object that can be further modified

See Also

`dynaplot`, `fitplot`, `gridplot`, `prdnrmplot`

Examples

```
## Not run:
## load a fitted tidal object
data(tidfit)

## defaults to the fiftieth quantile
gridplot(tidfit)

## no facets, all months
gridplot(tidfit, month = 'all')

## change the defaults
gridplot(tidfit, tau = c(0.1), month = c(3, 6, 9, 12),
 col_vec = c('red', 'blue', 'green'), flo_fac = 1)

## plot a tidalmean object
data(tidfitmean)

gridplot(tidfitmean)


## End(Not run)
```

`lnQ_sim` *Simulate a discharge time series*

Description

Simulate a discharge time series by modelling the statistical properties of an existing daily time series

Usage

```
lnQ_sim(dat_in, comps = FALSE, seed = NULL)
```

Arguments

dat_in	input data.frame that must include discharge and decimal time columns, see example dataset daydat
comps	logical indicating if components of the simulated time series are returned, see value.
seed	optional numeric value for random generation seed

Details

Daily flow data are simulated as the additive combination of a stationary seasonal component and serially-correlated errors estimated from the observed data. The stationary seasonal component is based on a seasonal regression of discharge over time. The residuals from this regression are used to estimate the error distribution using an ARIMA model. Parameters of the ARIMA model are chosen using stepwise estimation for nonseasonal univariate time series with the auto.arima function. Random errors from a standard normal distribution for the length of the original time series are generated using the model estimates with the arima.sim function. Finally, the errors are multiplied by the standard deviation of the original residuals and added to the seasonal component to create a simulated, daily log-flow time series.

Value

The original data frame with an additional column of simulated data named lnQ_sim if comps = FALSE. Otherwise, a two-element list is returned where the first element is 1) a list with the linear seasonal model fit to the observed time series and ARIMA model fit to the seasonal residuals, and 2) a data frame with the original data, the fit from the seasonal linear model (seas_fit), residuals from observed flow and seasonal fit (seas_res), standard deviation of seasonal residuals (sd_seas), simulated errors from the ARIMA model (errs), and simulated discharge time series (sim_out). Note that sim_out vector is converted to the same range as the input flow record.

See Also

daydat

Examples

```
## example data
data(daydat)

## simulate
lnQ_sim(daydat)
```

`lnres_err` *Simulate random errors from a time series*

Description

Simulate random errors of a water quality time series by modelling the statistical properties of an observed dataset

Usage

```
lnres_err(dat_in, yr = NULL, comps = FALSE, seed = NULL)
```

Arguments

`dat_in`	input `data.frame` that must include discharge and decimal time columns, see example dataset `daydat`
`yr`	numeric year value to use for the stationary model, defaults to the median year
`comps`	logical indicating if the WRTDS model used to get response error measures is also returned, see value.
`seed`	optional numeric value for random generation seed

Details

Random errors for a stationary seasonal water quality time series on a daily time step are generated by modelling residuals from an observed dataset. First, a stationary seasonal model is created by fitting a `wrtds` model and estimating an error distribution the residuals using the `auto.arima` function. Accumulated standard errors from the regression are also retained for each residual. Random errors using the estimated auto-regressive structures are simulated using `arima.sim` for the entire year and multiplied by the corresponding standard error estimate from the regression. The entire year is then repeated for every year in the observed time series. The final simulated errors are rescaled to the range of the original residuals that were used to estimate the distribution.

Value

The original data frame with additional columns for the random errors (`errs`) and the standard error estimates for each residual (`scls`). If `comps = TRUE`, a two-element list is returned that also includes the WRTDS model used as a basis for errors and scale values.

See Also

`daydat`

Examples

```
## Not run:
## example data
data(daydat)

## get errors
lnres_err(daydat)

## End(Not run)
```

`lnres_sim` *Simulate a water quality time series*

Description

Simulate a water quality time series with an estimated error structure and simulated discharge effect

Usage

```
lnres_sim(dat_in, lnQ_coef = NULL)
```

Arguments

`dat_in`	input `data.frame` that must include estimated error and simulated discharge time series, see `lnres_err` and `lnQ_sim` respectively
`lnQ_coef`	numeric vector of coefficients of the same length of the time series in `dat_in` that is multiplied by the discharge vector, see details

Details

This function creates a simulated water quality time series and requires error estimates for an observed water quality dataset and a simulated discharge time series. The water quality time series is created as the additive combination of a seasonal, stationary time component (as in `lnQ_sim` for discharge), a random error component from `lnres_err`, and a simulated discharge time series from `lnQ_sim`. The discharge time series is considered an explicit component of the water quality time series and is first centered at zero prior to adding. The optional vector of coefficients passed to `lnQ_coef` can mediate the influence of discharge on the water quality time series. For example, a vector of all zeroes implies no effect, whereas a vector of all ones implies a constant effect (default).

Value

The original data frame with additional columns for the seasonal water quality model (`lnres_seas`), a flow-independent water quality time series (`lnres_noQ`), and a flow-dependent time series (`lnres_Q`).

See Also

`daydat` for the format of an input dataset, `lnQ_sim` for simulating discharge, and `lnres_err` for estimating the error distribution of the water quality time series, `all_sims` for completing all steps at once.

Examples

```
## Not run:
## example data
data(daydat)

## get simulated discharge
sims <- lnQ_sim(daydat)

## get error structure of wq time series
sims <- lnres_err(sims)

## get simulated wq time series using results from previous
lnres_sim(sims)

## End(Not run)
```

`modfit` *Fit weighted regression and get predicted/normalized response variable*

Description

Fit weighted regression and get predicted/normalized response variable from a data frame. This is a wrapper for multiple function used to create a weighted regression model and should be used rather than the individual functions.

Usage

```
modfit(dat_in, ...)

## Default S3 method:
modfit(dat_in, ...)

## S3 method for class 'tidal'
modfit(dat_in, ...)

## S3 method for class 'tidalmean'
modfit(dat_in, ...)

## S3 method for class 'data.frame'
modfit(dat_in, resp_type = "quantile", ...)
```

Arguments

`dat_in`	input `data.frame` for fitting the model, see details
`...`	arguments passed to or from other methods
`resp_type`	chr string indicating the type of model response to use, quantile or mean model

Details

This function is used as a convenience to combine several functions that accomplish specific tasks, primarily the creation of a tidal or tidalmean object, fitting of the weighted regression models with `wrtds`, extraction of fitted values from the interpolation grids using `respred`, and normalization of the fitted values from the interpolation grid using `resnorm`. The format of the input should be a `data.frame` with response variable observations as rows and the first four columns as date, response variable, salinity/flow, and detection limits. The order of the columns may vary provided the order of each of the four critical variables is specified by the `ind` argument that is passed to the `tidal` or `tidalmean` function. The response variable data are also assumed to be in log-space, otherwise use `reslog = FALSE` which is also passed to the `tidal` or `tidalmean` function. The dataset described in `chldat` is an example of the correct format.

For quantile models, the default conditional quantile that is predicted is the median (`tau = 0.5`, passed to the `wrtds` function). Numerous other arguments affect the output and the default parameters may not be appropriate for all scenarios. Arguments used by other functions can be specified explicitly with the initial call. The documentation for the functions under 'see also' should be consulted for available arguments, as well as the examples that illustrate common changes to the default values.

Value

A tidal object with predicted and normalized response variable predictions, attributes updated accordingly.

See Also

See the help files for `tidal`, `tidalmean`, `wrtds`, `getwts`, `respred`, and `resnorm` for arguments that can be passed to this function.

Examples

```
## Not run:
## load data
data(chldat)

## fit the model and get predicted/normalized data for response variable
# default median fit
# grids predicted across salinity range with ten values
res <- modfit(chldat)

# for mean models
res <- modfit(chldat, resp_type = 'mean')

## fit different quantiles and smaller interpolation grid
res <- modfit(chldat, tau = c(0.2, 0.8), flo_div = 5)

## fit with different window widths
# half-window widths of one day, five years, and 0.3 salinity
res <- modfit(chldat, wins = list(1, 5, 0.3))

## suppress console output
```

```
res <- modfit(chldat, trace = FALSE)

## End(Not run)
```

---

nobsplot *Plot number of observations in a WRTDS interpolation grid*

Description

Plot number of observations for each point in a WRTDS interpolation grid. This is a diagnostic plot to identify sample size for each unique location in the domain of the time series that is considered during model fitting.

Usage

```
nobsplot(dat_in, ...)

## Default S3 method:
nobsplot(dat_in, month = "all", years = NULL,
  col_vec = NULL, allflo = TRUE, ncol = NULL, grids = FALSE,
  pretty = TRUE, ...)

## S3 method for class 'tidal'
nobsplot(dat_in, ...)

## S3 method for class 'tidalmean'
nobsplot(dat_in, ...)
```

Arguments

dat_in	input tidal or tidalmean object
...	arguments passed to other methods
month	numeric indicating months to plot or chr string 'all' to indicate all months with no plot facets
years	numeric vector of years to plot, defaults to all
col_vec	chr string of plot colors to use, passed to gradcols and scale_fill_gradientn for grid shading. Any color palette from RColorBrewer can be used as a named input. Palettes from grDevices must be supplied as the returned string of colors for each palette.
allflo	logical indicating if the salinity/flow values for plotting are limited to the fifth and ninety-fifth percentile of observed values for the month of interest
ncol	numeric argument passed to facet_wrap indicating number of facet columns
grids	logical indicating if grid lines are present
pretty	logical indicating if my subjective idea of plot aesthetics is applied, otherwise the ggplot default themes are used

Details

The plots can be used sample size as an indication of model fit for each unique location in the domain space of the time series. The plots show grids of the number of observations with weights greater than zero for each unique date and salinity/flow combination. The `obs` attribute in the `tidal` or `tidalmean` object is created during model fitting and has the same dimensions as the interpolation grid. Each row is a unique date in the original dataset and each column is a salinity/flow value used to fit each regression (i.e., values in the `flo_grd` attribute). In general, low points in the grid may indicate locations in the time series where insufficient data could affect model fit.

Unlike `gridplot`, interpolation of the grids for a smoother appearance is not allowed because the objective is to identify specific locations with low sample size. For the former function, the objective is to characterize general trends over time rather values at specific locations.

Value

A `ggplot` object that can be further modified

See Also

`wtsplot` for an alternative to evaluating weights with different window width combinations

Examples

```
## Not run:
## load a fitted tidal object
data(tidfit)

## default plot
nobsplot(tidfit)

## no facets, all months
nobsplot(tidfit)

## change the defaults
nobsplot(tidfit, tau = c(0.1), month = c(3, 6, 9, 12),
 col_vec = c('red', 'blue', 'green'), flo_fac = 1)

## plot a tidalmean object
data(tidfitmean)

nobsplot(tidfitmean)


## End(Not run)
```

obsplot *Plot observed response variable and salinity/flow data*

Description

Plot observed response variable and salinity/flow time series from a tidal object

Usage

```
obsplot(dat_in, ...)

## Default S3 method:
obsplot(dat_in, lines = TRUE, logspace = TRUE,
  dt_rng = NULL, pretty = TRUE, col = "black", lwd = 1, size = 2,
  alpha = 1, ...)

## S3 method for class 'tidal'
obsplot(dat_in, ...)

## S3 method for class 'tidalmean'
obsplot(dat_in, ...)
```

Arguments

dat_in	input tidal or tidalmean object
...	arguments passed to geom_line
lines	logical indicating if a line plot is used, otherwise points
logspace	logical indicating if plots are in log space
dt_rng	Optional chr string indicating the date range of the plot. Must be two values in the format 'YYYY-mm-dd' which is passed to as.Date.
pretty	logical indicating if my subjective idea of plot aesthetics is applied, otherwise the ggplot default themes are used
col	chr string of plot color to use
lwd	numeric value indicating width of lines
size	numeric value indicating size of points
alpha	numeric value indicating transparency of points or lines

Value

A ggplot object that can be further modified

See Also

fitplot

Examples

```
## load a fitted tidal object
data(tidfit)

## plot using defaults
obsplot(tidfit)

## changing default
obsplot(tidfit, alpha = 0.5, size = 4, col = 'blue', lines = FALSE)

## plot a tidalmean object
data(tidfitmean)

obsplot(tidfitmean)
```

---

prdnrmplot *Plot combined predicted and normalized results from a tidal object*

Description

Plot combined predicted and normalized results from a tidal object to evaluate the influence of salinity or flow changes on the response variable. The plot is similar to that produced by `fitplot` except predicted values are shown as points and observed values are removed.

Usage

```
prdnrmplot(dat_in, ...)

## S3 method for class 'tidal'
prdnrmplot(dat_in, tau = NULL, annuals = TRUE,
  logspace = TRUE, dt_rng = NULL, col_vec = NULL, lwd = 1, size = 2,
  alpha = 1, min_mo = 9, mo_strt = 10, pretty = TRUE, plot = TRUE,
  ...)

## S3 method for class 'tidalmean'
prdnrmplot(dat_in, annuals = TRUE, logspace = TRUE,
  dt_rng = NULL, col_vec = NULL, lwd = 1, size = 2, alpha = 1,
  min_mo = 9, mo_strt = 10, pretty = TRUE, plot = TRUE, ...)
```

Arguments

`dat_in`	input tidal or tidalmean object
`...`	arguments passed to `geom_line`
`tau`	numeric vector of quantiles to plot, defaults to all in object if not supplied
`annuals`	logical indicating if plots are annual aggregations of results
`logspace`	logical indicating if plots are in log space

dt_rng	Optional chr string indicating the date range of the plot. Must be two values in the format 'YYYY-mm-dd' which is passed to as.Date.
col_vec	chr string of plot colors to use, passed to gradcols. Any color palette from RColorBrewer can be used as a named input. Palettes from grDevices must be supplied as the returned string of colors for each palette.
lwd	numeric value indicating width of lines
size	numeric value indicating size of points
alpha	numeric value indicating transparency of points or lines
min_mo	numeric value from one to twelve indicating the minimum number of months with observations for averaging by years, applies only if annuals = TRUE. See annual_agg.
mo_strt	numeric indicating month to start aggregation years, defaults to October for USGS water year from October to September, applies only if annuals = TRUE. See annual_agg.
pretty	logical indicating if my subjective idea of plot aesthetics is applied, otherwise the ggplot default themes are used
plot	logical if plot is returned, otherwise data used in the plot

Value

A ggplot object that can be further modified

See Also

fitplot, sliceplot

Examples

```
## load a fitted tidal object
data(tidfit)

## plot using defaults
prdnrmplot(tidfit)

## get the same plot but use default ggplot settings
prdnrmplot(tidfit, pretty = FALSE)

## plot in log space
prdnrmplot(tidfit, logspace = TRUE)

## plot specific quantiles
prdnrmplot(tidfit, tau = c(0.1, 0.9))

## plot the normalized predictions
prdnrmplot(tidfit, predicted = FALSE)

## plot as monthly values
```

```
prdnrmplot(tidfit, annuals = FALSE)

## format the x-axis is using annual aggregations
library(ggplot2)

prdnrmplot(tidfit, annual = TRUE) +
 scale_x_date(limits = as.Date(c('2000-01-01', '2012-01-01')))

## modify the plot as needed using ggplot scales, etc.
prdnrmplot(tidfit, pretty = FALSE, linetype = 'dashed') +
 theme_classic() +
 scale_y_continuous(
   'Chlorophyll',
   limits = c(0, 50)
   ) +
 scale_colour_manual(
   '',
   labels = c('lo', 'md', 'hi'),
   values = c('red', 'green', 'blue'),
   guide = guide_legend(reverse = TRUE)
   )

 ## plot a tidalmean object
 data(tidfitmean)

 prdnrmplot(tidfitmean)
```

`resnorm` *Get salinity/flow normalized WRTDS predictions from interpolation grids*

Description

Get normalized model predictions from WRTDS to remove the effect of salinity/flow on the response variable. Predicted values in the interpolation grids are averaged across dates.

Usage

```
resnorm(dat_in, ...)

## S3 method for class 'tidal'
resnorm(dat_in, trace = TRUE, ...)

## S3 method for class 'tidalmean'
resnorm(dat_in, trace = TRUE, ...)
```

Arguments

dat_in	input tidal or tidalmean object
...	arguments passed to or from other methods
trace	logical indicating if progress is shown in the console

Details

This function is used after wrtds to normalize predicted values of the response variable from the interpolation grid for each model. The normalized values are based on the average of all predicted estimates across the range of salinity/flow values that have occurred on the same date throughout each year. For example, normalized values for July 2000 are the mean predicted response at that date using the observed salinity/flow values that occur in July of all years. The normalized values allow an interpretation of trends in the response variable that are independent of changes in salinity or freshwater inputs.

Value

Appends columns to the data.frame for normalized values. For, tidal objects, columns are named starting with the prefix 'norm', e.g., 'norm0.5' are the normalized values for the fit through the median. For tidalmean objects, columns are appended for the log-transformed and back-transformed normalized values, named 'norm' and 'bt_norm'.

Examples

```
## Not run:
##

# load a tidal object
data(tidobj)

# get flow-normalized values for each quantile
res <- resnorm(tidobj)

# load a tidalmean object
data(tidobjmean)

# get flow-normalized values
res <- resnorm(tidobjmean)

## End(Not run)
```

---

respred *Get WRTDS predictions from interpolation grids*

Description

Get model predictions from WRTDS using linear interpolation of values in grids

Usage

```
respred(dat_in, ...)

## S3 method for class 'tidal'
respred(dat_in, dat_pred = NULL, trace = TRUE,
  omit = TRUE, ...)

## S3 method for class 'tidalmean'
respred(dat_in, dat_pred = NULL, trace = TRUE,
  omit = TRUE, ...)
```

Arguments

dat_in	input tidal or tidalmean object
...	arguments passed to or from other methods
dat_pred	optional data to predict using the interpolation grids in dat_in, defaults to observed data in dat_in if not supplied, see details
trace	logical indicating if progress is shown in the console
omit	logical indicating if observations in dat_pred that are outside of the range of data used to fit the model are removed, see details

Details

This function is used after wrtds to estimate predicted values of the response variable from the interpolation grids. The estimated values are based on a bilinear interpolation of the four predicted response values at two salinity/flow and two date values nearest to the observed salinity/flow and date values to predict.

Data for dat_pred must be a data frame of two columns for date and flow variables (date and numeric objects). The columns must be named 'date' and 'flo'. Values that are outside of the range of data used to fit the model are removed with a warning. It is assumed that the flow variable is not scaled (i.e., raw data) as in a tidal or tidalmean object. The dimensions of the output data are modified to match dat_pred if observations are removed. The omit argument should not equal FALSE and is included only for use with wrtdscv to evaluate folds of the original dataset.

Value

Appends columns to the input data.frame for the predicted values. For tidal objects, columns are named starting with the prefix 'fit', e.g., 'fit0.5' are the predicted values for the fit through the median. For tidalmean objects, predicted values are appended for the mean model in log-space and the observed values from the back-transformed grids. Columns are named as 'fits' and 'bt_fits'.

Examples

```
##

# load a tidal object
data(tidobj)
```

```
# get fitted values for each quantile
res <- respred(tidobj)

# load a tidalmean object
data(tidobjmean)

# get predicted values
res <- respred(tidobjmean)
```

`resscls` *Get the scale parameters for predicted values*

Description

Get the scale parameters for predicted values of the response variable, only applies to `tidalmean` objects.

Usage

```
resscls(dat_in, ...)

## S3 method for class 'tidalmean'
resscls(dat_in, dat_pred = NULL, ...)
```

Arguments

`dat_in`	input tidalmean object
`...`	arguments passed to or from other methods
`dat_pred`	optional data to predict using the interpolation grids in dat_in, defaults to observed data in `dat_in` if not supplied

Details

This function is used after `wrtds` to get scale parameters for predicted values of the response variable from the interpolation grids. The values are based on a bilinear interpolation of the four predicted response values at two salinity/flow and two date values nearest to the observed salinity/flow and date values to predict.

Value

Appends columns to the data.frame for the associated scale value for the predicted values. A column is appended to the `dat_in` object, named 'scls'.

Examples

```
##

# load a tidalmean object
data(tidobjmean)

# get predicted values
res <- resscls(tidobjmean)
```

`samp_sim` *Sample a daily time series at a set frequency*

Description

Sample a daily water quality time series at a set monthly frequency

Usage

```
samp_sim(dat_in, unit = "month", irregular = TRUE, missper = 0,
  blck = 1, blckper = FALSE)
```

Arguments

`dat_in`	input `data.frame` that is returned from `lnres_sim` or `all_sims`
`unit`	chr string indicating sampling unit, must be year, quarter, month, week, or yday for equivalent lubridate function
`irregular`	logical indicating if monthly sampling is done randomly within each `unit`, otherwise the first value is returned
`missper`	numeric from 0-1 indicating percentage of observations used for test dataset
`blck`	numeric indicating block size for resampling test dataset, see details
`blckper`	logical indicating if the value passed to `blck` is a proportion of `missper`, i.e., blocks are to be sized as a percentage of the total size of the missing data

Details

This function is intended for sampling a simulated daily time series of water quality that is returned by `lnres_sim` or `all_sims`.

The `missper` argument is used to create a test dataset as a proportion of all observations in the sub-sampled output dataset. The test dataset is created with random block sampling appropriate for time series. Block sampling of the output dataset occurs until the number of unique observations is equal to the percentage defined by `missper`. Overlap of blocks are not doubly considered towards the observation counts to satisfy `missper`, i.e., sets of continuous observations longer than `blck` can be returned because of sampling overlap. Setting `blck = 1` and `blockper = FALSE` is completely random sampling for missing data. Values for `blck` must be 1 or greater if `blockper = FALSE` and 1 or less if `blckper = T`. If `blck = 1` and `blckper = T`, the missing data will be one continuous block.

Value

Original data frame with rows subset based on number of desired monthly samples. If `missper > 0`, a list is returned where the first element is the index values for the test dataset and the second is the complete subsampled dataset.

See Also

`lnres_sim`, `all_sims`

Examples

```
## Not run:
## example data
data(daydat)

## simulate
tosamp <- all_sims(daydat)

## sample
samp_sim(tosamp)

## sample and create test dataset
# test dataset is 30% size of monthly subsample using block sampling with size = 4
samp_sim(tosamp, missper = 0.3, blck = 4)

## End(Not run)
```

---

seasplot *Plot seasonal trends across all years*

Description

Plot seasonal trends by combining annual data

Usage

```
seasplot(dat_in, ...)

## S3 method for class 'tidal'
seasplot(dat_in, tau = NULL, predicted = TRUE, span = 0.4,
  lwd = 1, size = 2, alpha = 1, col_vec = NULL, grids = TRUE,
  logspace = TRUE, ...)

## S3 method for class 'tidalmean'
seasplot(dat_in, predicted = TRUE, span = 0.4,
  lwd = 1, size = 2, alpha = 1, col_vec = NULL, grids = TRUE,
  logspace = TRUE, ...)
```

Arguments

dat_in	Input data object
...	arguments passed to other methods
tau	numeric of quantile to plot
predicted	logical indicating if seasonal smooth is based on model predictions, default TRUE, otherwise the smooth is based on flow-normalized predictions
span	numeric indicating the smoothing parameter for the loess fit, passed to stat_smooth
lwd	numeric value indicating width of lines
size	numeric value indicating size of points
alpha	numeric value indicating transparency of points or lines
col_vec	chr string of plot colors to use, passed to gradcols. Any color palette from RColorBrewer can be used as a named input. Palettes from grDevices must be supplied as the returned string of colors for each palette.
grids	logical indicating if grid lines are present
logspace	logical indicating if plots are in log space

Details

Seasonal variation across all years can be viewed by showing the observed annual data on a common y-axis. The year value is removed from the results such that the y-axis shows only the day of the year. A simple loess (locally estimated) polynomial smooth is added to show the seasonal trend in the results, where the smoother is fit through the model results for the observed data. The fit can be smoothed through the model predictions or the flow-normalized predictions, neither of which are shown on the plot.

See Also

dynaplot, fitmoplot, gridplot, and sliceplot produce similar graphics except variation in the same month across years is emphasized.

Examples

```
# load a fitted tidal object
data(tidfit)

# plot using defaults
# defaults to all quantiles for tidal object
seasplot(tidfit)

# tidalmean object
seasplot(tidfitmean)
```

`seasyrplot` *Plot seasonal model response by years*

Description

Plot seasonal model response by years on a common axis

Usage

```
seasyrplot(dat_in, ...)

## S3 method for class 'tidal'
seasyrplot(dat_in, years = NULL, tau = NULL,
  predicted = TRUE, logspace = TRUE, col_vec = NULL, grids = TRUE,
  pretty = TRUE, lwd = 0.5, alpha = 1, ...)

## S3 method for class 'tidalmean'
seasyrplot(dat_in, years = NULL, tau = NULL,
  predicted = TRUE, logspace = TRUE, col_vec = NULL, grids = TRUE,
  pretty = TRUE, lwd = 0.5, alpha = 1, ...)
```

Arguments

`dat_in`	input tidal or tidalmean object
`...`	arguments passed to other methods
`years`	numeric vector of years to plot
`tau`	numeric vector of quantiles to plot, defaults to all in object if not supplied
`predicted`	logical indicating if standard predicted values are plotted, default `TRUE`, otherwise normalized predictions are plotted
`logspace`	logical indicating if plots are in log space
`col_vec`	chr string of plot colors to use, passed to `gradcols`. Any color palette from RColorBrewer can be used as a named input. Palettes from grDevices must be supplied as the returned string of colors for each palette.
`grids`	logical indicating if grid lines are present
`pretty`	logical indicating if my subjective idea of plot aesthetics is applied, otherwise the `ggplot` default themes are used
`lwd`	numeric value indicating width of lines
`alpha`	numeric value indicating transparency of points or lines

Details

The plot is similar to that produced by seasplot except the model estimates are plotted for each year as connected lines, as compared to loess lines fit to the model results. seasyrplot is also similar to sliceplot except the x-axis and legend grouping variable are flipped. This is useful for evaluating between-year differences in seasonal trends.

Multiple predictions per month are averaged for a smoother plot.

Note that the year variable used for color mapping is treated as a continuous variable although it is an integer by definition.

Value

A ggplot object that can be further modified

See Also

seasplot, sliceplot

Examples

```
## load a fitted tidal object
data(tidfit)

# plot using defaults
seasyrplot(tidfit)

# get the same plot but use default ggplot settings
seasyrplot(tidfit, pretty = FALSE)

# plot specific quantiles
seasyrplot(tidfit, tau = c(0.9))

# plot the normalized predictions
seasyrplot(tidfit, predicted = FALSE)

# modify the plot as needed using ggplot scales, etc.

library(ggplot2)

seasyrplot(tidfit, pretty = FALSE, linetype = 'dashed') +
 theme_classic() +
 scale_y_continuous(
   'Chlorophyll',
   limits = c(0, 5)
   )

# plot a tidalmean object
data(tidfitmean)

seasyrplot(tidfitmean)
```

`sliceplot` *Plot time slices within a tidal object*

Description

Plot time slices within a tidal object to view response variable observations, predictions, and normalized results at regular annual intervals.

Usage

```
sliceplot(dat_in, ...)

## S3 method for class 'tidal'
sliceplot(dat_in, slices = c(1, 7), tau = NULL,
  dt_rng = NULL, col_vec = NULL, predicted = TRUE, logspace = TRUE,
  grids = TRUE, pretty = TRUE, lwd = 1, size = 2, alpha = 1, ...)

## S3 method for class 'tidalmean'
sliceplot(dat_in, slices = c(1, 7), predicted = TRUE,
  dt_rng = NULL, col_vec = NULL, logspace = TRUE, grids = TRUE,
  pretty = TRUE, lwd = 1, size = 2, alpha = 1, ...)
```

Arguments

`dat_in`	input tidal or tidalmean object
`...`	arguments passed to other methods
`slices`	numeric vector of calender months to plot, i.e., 1 - 12
`tau`	numeric vector of quantile to plot. The function will plot the 'middle' quantile if none is specified, e.g., if 0.2, 0.3, and 0.4 are present in the fitted model object then 0.3 will be plotted.
`dt_rng`	Optional chr string indicating the date range of the plot. Must be two values in the format 'YYYY-mm-dd' which is passed to `as.Date`.
`col_vec`	chr string of plot colors to use, passed to `gradcols`. Any color palette from RColorBrewer can be used as a named input. Palettes from grDevices must be supplied as the returned string of colors for each palette.
`predicted`	logical indicating if standard predicted values are plotted, default `TRUE`, otherwise normalized predictions are plotted
`logspace`	logical indicating if plots are in log space
`grids`	logical indicating if grid lines are present
`pretty`	logical indicating if my subjective idea of plot aesthetics is applied, otherwise the `ggplot` default themes are used
`lwd`	numeric value indicating width of lines
`size`	numeric value indicating size of points
`alpha`	numeric value indicating transparency of points or lines

Details

This is a modification of fitplot that can be used to plot selected time slices from the results of a fitted tidal object. For example, all results for a particular month across all years can be viewed. This is useful for evaluating between-year differences in results for constant season. Only one quantile fit can be shown per plot because the grouping variable is mapped to the slices.

Value

A ggplot object that can be further modified

See Also

fitplot, prdnrmplot

Examples

```
## load a fitted tidal object
data(tidfit)

# plot using defaults
sliceplot(tidfit)

# get different months - march and september
sliceplot(tidfit, slices = c(3, 9))

# normalized predictions, 10th percentile
sliceplot(tidfit, tau = 0.1, predicted = FALSE)

# normalized values all months, change line aesthetics, log-space, 90th
# add title
library(ggplot2)
sliceplot(tidfit,
 slices = 1:12,
 size = 1.5,
 tau = 0.9,
 alpha = 0.6,
 predicted = FALSE,
 logspace = TRUE
) +
ggtitle('Normalized predictions for all months, 90th percentile')

 ## plot a tidalmean object
 data(tidfitmean)

 sliceplot(tidfitmean)
```

`tidal` *Create a tidal class object*

Description

Prepare water quality data for weighted regression by creating a tidal class object

Usage

```
tidal(dat_in, ind = c(1, 2, 3, 4), reslab = NULL, flolab = NULL,
  reslog = TRUE, rm_miss = FALSE, ...)
```

Arguments

`dat_in`	Input data frame for a water quality time series with four columns for date (Y-m-d format), response variable, salinity/flow, and detection limit for left-censored data
`ind`	four element numeric vector indicating column positions of date, response variable, salinity/flow, and detection limit of input data frame
`reslab`	character string or expression for labelling the response variable in plots, defaults to log-chlorophyll in ug/L
`flolab`	character string or expression for labelling the flow variable in plots, defaults to Salinity
`reslog`	logical indicating if the response variable is already in log-space, default `TRUE`
`rm_miss`	logical indicating if missing observations in the input data are removed
`...`	arguments passed from other methods

Details

This function is a simple wrapper to `structure` that is used to create a tidal object for use with weighted regression in tidal waters. Input data should be a four-column `data.frame` with date, response variable, salinity/flow data, and detection limit for each observation of the response. The response variable is assumed to be log-transformed, otherwise use `reslog = FALSE`. Salinity data can be provided as fraction of freshwater or as parts per thousand. The limit column can be entered as a sufficiently small number if all values are above the detection limit or no limit exists. The current implementation of weighted regression for tidal waters only handles left-censored data. Missing observations are also removed.

Value

A tidal object as a data frame and attributes. The data frame has columns ordered as date, response variable, salinity/flow (rescaled to 0, 1 range), detection limit, logical for detection limit, day number, month, year, and decimal time. The attributes are as follows:

`names` Column names of the data frame

`row.names` Row names of the data frame

`class` Class of the object

`half_wins` List of numeric values used for half-window widths for model fitting, in the same order as the wt_vars argument passed to `getwts`. Initially will be NULL if `wrtds` has not been used.

`fits` List of matrices with fits for the WRTDS interpolation grid, defaults to one list for the median quantile. Initially will be `NULL` if `wrtds` has not been used.

`predonobs` A `data.frame` of predictions using the observed data that were used to fit the model. This is required for `wrtdsperf` if a novel dataset is used for predictions after fitting the model. Initially will be NULL if `respred` has not been used.

`flo_grd` Numeric vector of salinity/flow values that was used for the interpolation grids

`floobs_rng` Two element vector indicating the salinity/flow range of the observed data

`nobs` List with one matrix showing the number of weights greater than zero for each date and salinity/flow combination used to create the fit matrices in `fits`. Number of observations are the same for each quantile model. Initially will be `NULL` if `wrtds` has not been used.

`reslab` expression or character string for response variable label in plots

`flolab` expression or character string for flow variable label in plots

Examples

```
## raw data

data(chldat)

## format
chldat <- tidal(chldat)
```

`tidalmean` *Create a tidalmean class object*

Description

Prepare water quality data for weighted regression for the mean response by creating a tidalmean class object

Usage

```
tidalmean(dat_in, ind = c(1, 2, 3, 4), reslab = NULL, flolab = NULL,
  reslog = TRUE, rm_miss = FALSE, ...)
```

Arguments

`dat_in`	Input data frame for a water quality time series with four columns for date (Y-m-d format), response variable, salinity/flow, and detection limit for left-censored data
`ind`	four element numeric vector indicating column positions of date, response variable, salinity/flow, and detection limit of input data frame

reslab	character string or expression for labelling the response variable in plots, defaults to log-chlorophyll in ug/L
flolab	character string or expression for labelling the flow variable in plots, defaults to Salinity
reslog	logical indicating if input response variable is already in log-space, default TRUE
rm_miss	logical indicating if missing observations in the input data are removed
...	arguments passed from other methods

Details

This function is a simple wrapper to structure that is used to create a tidalmean object for use with weighted regression in tidal waters, specifically to model the mean response as compared to a conditional quantile. Input data should be a four-column data.frame with date, response variable, salinity/flow data, and detection limit for each observation of the response. The response data are assumed to be log-transformed, otherwise use reslog = FALSE. Salinity data can be provided as fraction of freshwater or as parts per thousand. The limit column can be entered as a sufficiently small number if all values are above the detection limit or no limit exists. The current implementation of weighted regression for tidal waters only handles left-censored data. Missing observations are also removed.

The tidalmean object structure is almost identical to the tidal object, with the exception of an additional attribute for the back-transformed interpolation grid. This is included to account for retransformation bias of log-transformed variables associated with mean models.

Value

A tidalmean object as a data frame and attributes. The data frame has columns ordered as date, response variable, salinity/flow (rescaled to 0, 1 range), detection limit, logical for detection limit, day number, month, year, and decimal time. The attributes are as follows:

names Column names of the data frame

row.names Row names of the data frame

class Class of the object

half_wins List of numeric values used for half-window widths for model fitting, in the same order as the wt_vars argument passed to getwts. Initially will be NULL if wrtds has not been used.

fits List with a single element with fits for the WRTDS mean interpolation grid. Initially will be NULL if wrtds has not been used.

predonobs A data.frame of predictions using the observed data that were used to fit the model. This is required for wrtdsperf if a novel dataset is used for predictions after fitting the model. Initially will be NULL if respred has not been used.

bt_fits List with a single element with back-transformed fits for the WRTDS mean interpolation grid. Initially will be NULL if wrtds has not been used.

flo_grd Numeric vector of salinity/flow values that was used for the interpolation grids

floobs_rng Two element vector indicating the salinity/flow range of the observed data

nobs List with one matrix showing the number of weights greater than zero for each date and salinity/flow combination used to create the fit matrices in fits. Initially will be NULL if wrtds has not been used.

`reslab` expression or character string for response variable label in plots

`flolab` expression or character string for flow variable label in plots

Examples

```
## raw data

data(chldat)

## format
chldat <- tidalmean(chldat)
```

`tidfit` *Monthly chlorophyll time series for Hillsborough Bay as a tidal object*

Description

An identical object as `tidobj` with the addition of chlorophyll predictions and normalized estimates after running `respred` and `resnorm`.

Usage

```
tidfit
```

Format

A `tidal` and `data.frame` object with 156 rows and 15 variables:

`date` Date

`res` numeric

`flo` numeric

`lim` numeric

`not_cens` logical

`day_num` numeric

`month` numeric

`year` numeric

`dec_time` numeric

`fit0.1` numeric

`fit0.5` numeric

`fit0.9` numeric

`norm0.1` numeric

`norm0.5` numeric

`norm0.9` numeric

See Also

tidal for full list of attributes in tidal objects, wrtds for creating the fits interpolation grids, and respred and resnorm for interpolating predicted and normalized values from the grids.

---

tidfitmean — *Monthly chlorophyll time series for Hillsborough Bay as a tidal object for the conditional mean model*

Description

An identical object as tidobjmean with the addition of predicted and normalized chlorophyll (log-space and back-transformed) after running respred and resnorm.

Usage

```
tidfitmean
```

Format

A tidalmean and data.frame object with 156 rows and 11 variables:

date Date

res numeric

flo numeric

lim numeric

not_cens logical

day_num numeric

month numeric

year numeric

dec_time numeric

fits numeric

bt_fits numeric

norm numeric

bt_norms numeric

See Also

tidalmean for full list of attributes in tidalmean objects, wrtds for creating the fits, bt_fits, and scls interpolation grids in the attributes, and respred and resnorm for interpolating predicted and normalized values from the grids.

tidobj *Monthly chlorophyll time series for Hillsborough Bay as a tidal object*

Description

Monthly chlorophyll time series for the Hillsborough Bay segment of Tampa Bay as a tidal object. Raw data are those in chldat with the addition of further processing using the tidal and wrtds functions. Model predictions are obtained for the tenth, median, and ninetieth conditional quantile of chlorophyll. The object also inherits methods from the data.frame class. The processed data includes columns for date, chlorophyll-a (res, in log-space), salinity as fraction of freshwater (flo, i.e., 0 - 1, with higher values indicating more freshwater), the detection limit for all stations for the respective date, a logical column indicating if the observed chlorophyll is at or below the detection limit, the date as decimal time minus the year, the month from 1 to 12, the year, and total decimal time. Attributes include column names, row names, class of the object, matrices of interpolation grids from the weighted regression for each conditional quantile, and vector of salinity values that were used to create the interpolation grids.

Usage

```
tidobj
```

Format

A tidal and data.frame object with 156 rows and 9 variables:

date Date

res numeric

flo numeric

lim numeric

not_cens logical

day_num numeric

month numeric

year numeric

dec_time numeric

See Also

tidal for full list of attributes in tidal objects and wrtds for creating the fits interpolation grids.

`tidobjmean` *Monthly chlorophyll time series for Hillsborough Bay as a tidal object, conditional mean model*

Description

Monthly chlorophyll time series for the Hillsborough Bay segment of Tampa Bay as a tidal object. Raw data are those in `chldat` with the addition of further processing using the `tidalmean` and `wrtds` functions. Model predictions are obtained using weighted regression for the conditional mean of chlorophyll. The object also inherits methods from the `data.frame` class. The processed data includes columns for date, chlorophyll-a (`res`, in log-space), salinity as fraction of freshwater (`flo`, i.e., 0 - 1, with higher values indicating more freshwater), the detection limit for all stations for the respective date, a logical column indicating if the observed chlorophyll is at or below the detection limit, the date as decimal time minus the year, the month from 1 to 12, the year, and total decimal time. Attributes include column names, row names, class of the object, an interpolation grid from the weighted regression for the mean response, a back-transformed interpolation grid from the weighted regression for the mean response, and vector of salinity values that were used to create the interpolation grids.

Usage

```
tidobjmean
```

Format

A `tidalmean` and `data.frame` object with 156 rows and 9 variables:

`date` Date

`res` numeric

`flo` numeric

`lim` numeric

`not_cens` logical

`day_num` numeric

`month` numeric

`year` numeric

`dec_time` numeric

See Also

`tidalmean` for full list of attributes in tidalmean objects and `wrtds` for creating the `fits`, `bt_fits`, and `scls` interpolation grids.

`winsrch_constrOptim` *Find the optimal half-window width combination*

Description

Find the optimal half-window width combination to use for weighted regression. This differs from `winsrch_optim` by using `constrOptim`

Usage

```
winsrch_constrOptim(dat_in, ...)

## Default S3 method:
winsrch_constrOptim(dat_in, wins_in = NULL,
  control = list(), lower = c(0.1, 1, 0.1), upper = c(2, 15, 2), ...)
```

Arguments

`dat_in`	input data object to use with weighted regression
`...`	arguments passed to `wrtdscv`, `wrtds`, or `getwts`
`wins_in`	starting list of window weights for initializing the search algorithm
`control`	A list of control parameters passed to `optim` (see details in `optim` help file).
`lower`	vector of minimum half-window widths to evaluate
`upper`	vector of maximum half-window widths to evaluate

Details

This function uses `optim` to minimize the error returned by `wrtdscv` for a given window combination. The search algorithm uses the limited-memory modification of the BFGS quasi-Newton method to impose upper and lower limits on the optimization search. These limits can be changed using the `lower` and `upper` arguments.

Value

Some stuff

See Also

`wrtdscv`, `winsrch_grid`

Examples

```
## Not run:
# setup parallel backend
library(doParallel)
ncores <- detectCores() - 1
registerDoParallel(cores = ncores)

# run search function - takes a while
res <- winsrch_optim(tidobjmean)

## End(Not run)
```

`winsrch_grid` *Evaluate half-window width combinations*

Description

Evaluate a grid of half-window width combinations to use for weighted regression

Usage

```
winsrch_grid(dat_in, ...)

## Default S3 method:
winsrch_grid(dat_in, grid_in = NULL, ...)
```

Arguments

`dat_in`	input data object to use with weighted regression
`...`	arguments passed to or from other methods
`grid_in`	optional input matrix of half-window widths created with `createsrch`, a default search grid is used if no input

Details

Processing time can be reduced by setting up a parallel backend, as in the examples. Note that this is not effective for small k-values (e.g., < 4) because each fold is sent to a processor, whereas the window width combinations in `grid_in` are evaluated in sequence.

This function should only be used to view the error surface associated with finite combinations of window-width combinations. A faster function to identify the optimal window widths is provided by `winsrch_optim`.

Value

A data frame of the search grid with associated errors for each cross-validation result. Errors for each grid row are averages of all errors for each fold used in cross-validation.

See Also

createsrch, wrtdscv, winsrch_optim

Examples

```
## Not run:
##
# setup parallel backend
library(doParallel)
ncores <- detectCores() - 2
registerDoParallel(cores = ncores)

# run search function using default search grid - takes a while
res <- winsrch_grid(tidobjmean)

# view the error surface
library(ggplot2)
ggplot(res, aes(x = factor(mos), y = factor(yrs), fill = err)) +
   geom_tile() +
   facet_wrap(~ flo) +
   scale_x_discrete(expand = c(0, 0)) +
   scale_y_discrete(expand = c(0,0)) +
   scale_fill_gradientn(colours = gradcols())

# optimal combo
res[which.min(res$err), ]

##
# create a custom search grid, e.g. years only
grid_in <- createsrch(mos = 1, yrs = seq(1, 10), flo = 1)

res <- winsrch_grid(tidobjmean, grid_in)


## End(Not run)
```

---

winsrch_optim *Find the optimal half-window width combination*

Description

Find the optimal half-window width combination to use for weighted regression.

Usage

```
winsrch_optim(dat_in, ...)

## Default S3 method:
winsrch_optim(dat_in, wins_in = NULL, control = list(factr
```

```
= 1e+07, parscale = c(1, 10, 1)), lower = c(0.1, 1, 0.1), upper = c(2, 15,
2), ...)
```

Arguments

dat_in	input data object to use with weighted regression
...	arguments passed to wrtdscv, wrtds, or getwts
wins_in	starting list of window weights for initializing the search algorithm
control	A list of control parameters passed to optim (see details in optim help file). The value passed to factr controls the convergence behavior of the "L-BFGS-B" method. Values larger than the default will generally speed up the optimization with a potential loss of precision. parscale describes the scaling values of the parameters.
lower	vector of minimum half-window widths to evaluate
upper	vector of maximum half-window widths to evaluate

Details

This function uses optim to minimize the error returned by wrtdscv for a given window combination. The search algorithm uses the limited-memory modification of the BFGS quasi-Newton method to impose upper and lower limits on the optimization search. These limits can be changed using the lower and upper arguments.

Value

Some stuff

See Also

wrtdscv, winsrch_grid

Examples

```
## Not run:
# setup parallel backend
library(doParallel)
ncores <- detectCores() - 1
registerDoParallel(cores = ncores)

# run search function - takes a while
res <- winsrch_optim(tidobjmean)

## End(Not run)
```

`wrtds` *Get WRTDS prediction grid*

Description

Get WRTDS prediction grid for observations of the response variable in a tidal or tidalmean object

Usage

```
wrtds(dat_in, ...)

## S3 method for class 'tidal'
wrtds(dat_in, flo_div = 10, tau = 0.5, trace = TRUE,
  fill_empty = FALSE, ...)

## S3 method for class 'tidalmean'
wrtds(dat_in, flo_div = 10, fill_empty = FALSE,
  trace = TRUE, ...)
```

Arguments

`dat_in`	input tidal or tidalmean object
`...`	arguments passed to or from other methods
`flo_div`	numeric indicating number of divisions across the range of salinity/flow to create the interpolation grid
`tau`	numeric vector indicating conditional quantiles to fit in the weighted regression, can be many
`trace`	logical indicating if progress is shown in the console
`fill_empty`	logical to fill missing values in interpolation grid using bilinear interpolation by season, see details

Value

Appends interpolation grid attributes to the input object. For a tidal object, this could include multiple grids for each quantile. For tidalmean objects, only one grid is appended to the 'fits' attribute, in addition to a back-transformed grid as the 'bt_fits' attribute and a grid of the scale parameter of each prediction as the 'scls' attribute. Grid rows correspond to the dates in the input data.

The `fill_empty` arguments uses bilinear interpolation of time by flow to fill missing data in the interpolation grids. The grids are subset by month before interpolating to retain the seasonal variation captured by the models. In general, this argument should not be used if more than ten percent of the interpolation grids are missing data. It may be helpful to improve visual appearance of some of the plotting results.

Examples

```
## Not run:
## load data
data(chldat)

## as tidal object
dat_in <- tidal(chldat)
res <- wrtds(dat_in)

## as tidalmean object
dat_in <- tidalmean(chldat)
res <- wrtds(dat_in)

## multiple quantiles
res <- wrtds(dat_in, tau = c(0.1, 0.5, 0.9))

## End(Not run)
```

`wrtdscv` *Use k-fold cross-validation to evaluate WRTDS model fit*

Description

Use k-fold cross-validation to evaluate WRTDS model fit based on supplied half-window widths.

Usage

```
wrtdscv(dat_in, ...)

## Default S3 method:
wrtdscv(dat_in, wins, k = 10, seed_val = 123,
  trace = TRUE, ...)
```

Arguments

`dat_in`	input tidal or tidalmean object
`...`	arguments passed to `wrtds` or `getwts`, e.g., `tau = 0.2` if a `tidal` object is used for `dat_in`
`wins`	list of input half-window widths of the order months, years, and salinity/flow, passed to `getwts`
`k`	number of folds to evaluate
`seed_val`	seed to keep the same dataset divisions between window width comparisons
`trace`	logical indicating if progress is printed in the console

Details

Default number of folds is ten. Each fold can be evaluated with multiple cores if a parallel back end is created prior to running the function (see the examples). This will greatly increase processing speed unless k is set to a small number.

Value

Overall error is the average of all errors for each fold.

See Also

getwts, wtsplot, winsrch_grid, winsrch_optim

Examples

```
## Not run:

library(doParallel)
ncores <- detectCores() - 1
registerDoParallel(cores = ncores)

# half-window widths to evaluate
# months, years, and salinity/flow
wins <- list(0.5, 10, 0.5)

# get ocv score for k = 10
wrtdscv(tidobjmean, wins = wins)

# get ocv score k = 2, tau = 0.2
wrtdscv(tidobj, wins = wins, tau = 0.2)

## End(Not run)
```

---

wrtdsperf *Get WRTDS performance metrics*

Description

Get WRTDS performance metrics including goodness of fit, root mean square error, and normalized mean square error.

Usage

```
wrtdsperf(dat_in, ...)

## S3 method for class 'tidal'
wrtdsperf(dat_in, logspace = TRUE, ...)

## S3 method for class 'tidalmean'
wrtdsperf(dat_in, logspace = TRUE, ...)
```

Arguments

dat_in	input tidal object which must already have fitted model data
...	arguments passed to or from other methods
logspace	logical if performance metrics use back-transformed residuals

Details

Goodness of fit is calculated using the goodfit function for quantile regression described in Koenker and Mochado 1999. Root mean square error is based on square root of the mean of the squared residuals. Normalized mean square error described in Gershenfeld and Weigend 1993 is the sum of the squared errors divided by the sum of the non-conditional errors (i.e., sum of the squared values of the observed minus the mean of the observed). This measure allows comparability of error values for data with different ranges, although the interpretation for quantile models is not clear. The value is provided as a means of comparison for WRTDS models created from the same data set but with different window widths during model fitting.

Performance metrics are only valid for observations and model residuals in log-space. Metrics also only apply to the data used to fit the model, i.e., performance will not be evaluated for novel data if the dat_pred argument was used with respred.

Value

A data.frame with the metrics for each quantile model

References

Gershenfeld, N.A., Weigend, A.S. 1993. The future of time series: learning and understanding. In: Weigend, A.S., Gershenfeld, N.A. (eds). Time Series Prediction: Forecasting the Future and Understanding the Past., second ed. Addison-Wesley, Santa Fe, New Mexico. pp. 1-70.

Koenker, R., Machado, J.A.F. 1999. Goodness of fit and related inference processes for quantile regression. Journal of the American Statistical Association. 94(448):1296-1310.

See Also

wrtdsrsd for residuals, goodfit

Examples

```
## load a fitted model object
data(tidfit)

## get performance metrics
wrtdsperf(tidfit)
```

`wrtdsrsd` *Get WRTDS residuals*

Description

Get WRTDS residuals for each quantile model. These are used to estimate goodness of fit of the model predictions.

Usage

```
wrtdsrsd(dat_in, ...)

## S3 method for class 'tidal'
wrtdsrsd(dat_in, trace = TRUE, ...)

## S3 method for class 'tidalmean'
wrtdsrsd(dat_in, trace = TRUE, ...)
```

Arguments

`dat_in`	input tidal object which must already have fitted model data
`...`	arguments passed to or from other methods
`trace`	logical indicating if progress is shown in the console

Value

Columns are added to the data of the tidal object for residuals and non-conditional residuals. Both are required to assess the goodness of fit measure described for quantile regression in Koenker and Machado (1999).

A tidal object with columns added to the `predonobs` attribute for the residuals ('rsd') and non-conditional residuals ('rsdnl') of each quantile model or a tidalmean object with columns added to the `predonobs` attribute for the residuals ('rsd') and back-transformed residuals ('bt_rsd').

References

Koenker, R., Machado, J.A.F. 1999. Goodness of fit and related inference processes for quantile regression. Journal of the American Statistical Association. 94(448):1296-1310.

See Also

`wrtds`, `wrtdsperf`, `goodfit`

Examples

```
## load a fitted model object
data(tidfit)

## run the function
res <- wrtdsrsd(tidfit)
head(res)
```

`wrtdstrnd` *Get WRTDS trends*

Description

Get WRTDS trends for annual and monthly groupings

Usage

```
wrtdstrnd(dat_in, ...)

## Default S3 method:
wrtdstrnd(dat_in, mobrks, yrbrks, molabs, yrlabs,
  aves = FALSE, mo_strt = 10, min_mo = 9, ...)

## S3 method for class 'tidal'
wrtdstrnd(dat_in, mobrks, yrbrks, molabs, yrlabs, tau = NULL,
  aves = FALSE, mo_strt = 10, min_mo = 9, ...)

## S3 method for class 'tidalmean'
wrtdstrnd(dat_in, mobrks, yrbrks, molabs, yrlabs,
  aves = FALSE, mo_strt = 10, min_mo = 9, ...)
```

Arguments

`dat_in`	input tidal or tidalmean object which must already have fitted model data
`...`	methods passed to or from other methods
`mobrks`	list of month groupings where each month is an integer from 1 to 12, see examples
`yrbrks`	numeric vector of breaks for years, see examples
`molabs`	character vector of names for month breaks, see examples
`yrlabs`	character vector of names for year breaks, see examples
`aves`	logical if averages within each period are also returned
`mo_strt`	numeric indicating month to start aggregation years for annual trends, defaults to October for USGS water year from October to September, passed to `annual_agg`
`min_mo`	numeric value from one to twelve indicating the minimum number of months with observations for averaging by years, passed to `annual_agg`
`tau`	numeric vector of quantile for estimating trends

Details

Trends are reported as percent changes of annual averages from the beginning to the end of each period. To reduce the effects of odd years at the beginning and end of each period, percent changes are based on an average of the first three and last three annual averages. For example, percent changes for January throughout an an entire time series from 1980 to 2000 would be the change of the average from January in 1980-1982 to the average from January in 1998-2000. Annual trends, e.g., percent changes from 1980-1986, 1987-1993, etc. do not average by the first and last three years in each grouping because the values are already based on annual averages as returned by `annual_agg`.

Note that the default minimum number of months argument (`min_mo`) may not be appropriate for all cases. Annual estimates should first be evaluated with `prdnrmplot` to verify that odd years with missing months are not driving results for the annual percent changes.

Averages in each period can be returned if `aves = TRUE`. These averages are based on annual averages within each period for congruency with the trend estimates.

All trends are based on back-transformed, flow-normalized results.

The user must supply the annual and monthly aggregation periods to the appropriate arguments. These are passed to `cut` and are left-open, right-closed along the interval.

Value

A `data.frame` with summary trends for each grouping

Examples

```
## load a fitted model object
data(tidfit)
data(tidfitmean)

## get trends

# setup month, year categories
mobrks <- list(c(1, 2, 3), c(4, 5, 6), c(7, 8, 9), c(10, 11, 12))
yrbrks <- c(1973, 1985, 1994, 2003, 2012)
molabs <- c('JFM', 'AMJ', 'JAS', 'OND')
yrlabs <- c('1974-1985', '1986-1994', '1995-2003', '2004-2012')

wrtdstrnd(tidfit, mobrks, yrbrks, molabs, yrlabs)
wrtdstrnd(tidfitmean, mobrks, yrbrks, molabs, yrlabs)

# get averages in each period
wrtdstrnd(tidfit, mobrks, yrbrks, molabs, yrlabs, aves = TRUE)
```

`wrtdstrnd_sk` *Get WRTDS trends using seasonal Kendall tests*

Description

Get WRTDS trends using seasonal Kendall tests

Usage

```
wrtdstrnd_sk(dat_in, ...)

## Default S3 method:
wrtdstrnd_sk(dat_in, mobrks, yrbrks, molabs, yrlabs, ...)

## S3 method for class 'tidal'
wrtdstrnd_sk(dat_in, mobrks, yrbrks, molabs, yrlabs,
  tau = NULL, trndvar = "norm", ...)

## S3 method for class 'tidalmean'
wrtdstrnd_sk(dat_in, mobrks, yrbrks, molabs, yrlabs,
  trndvar = "bt_norm", ...)
```

Arguments

dat_in	input tidal or tidalmean object which must already have fitted model data
...	methods passed to or from other methods
mobrks	list of month groupings where each month is an integer from 1 to 12, see examples
yrbrks	numeric vector of breaks for years, see examples
molabs	character vector of names for month breaks, see examples
yrlabs	character vector of names for year breaks, see examples
tau	numeric vector of quantile for estimating trends
trndvar	chr string of variable for trend evaluation, usually back-transformed, flow-normalized results, see details

Details

Trends are based on kendallSeasonalTrendTest for user-specified time periods. In general, the seasonal Kendall test evaluates monotonic trends using a non-parametric approach that accounts for seasonal variation in the time series.

All trends are based on back-transformed, flow-normalized results by default. The variable for evaluating trends can be changed with 'trndvar' as 'res', 'norm', or 'fit' for tidal objects and as 'res', 'bt_norm', or 'bt_fits' for tidalmean objects. In all cases, back-transformed variables are evaluated.

The user must supply the annual and monthly aggregation periods to the appropriate arguments. These are passed to cut and are left-open, right-closed along the interval.

Value

A data.frame with summary trends for each grouping, including med as the median value for the period of observation, tau as the magnitude and direction of the trend, slope as the Thiel-Sen slope for change per year, chitest as the significance test evaluating heterogeneity between seasons, ztest indicating significance of the overall trend, and perchg as 100 multiplied by the ratio of the annual slope to the median estimate of the time period (percent change per year).

As noted in kendallSeasonalTrendTest, the overall test is not appropriate if chitest indicates a small p-value.

References

Hirsch, R.M., Slack, J.R., Smith, R.A. 1982. Techniques of trend analysis for monthly water quality data. Water Resources Research, 18:107-121.

Millard, S. P. 2013. EnvStats: An R Package for Environmental Statistics. Springer, New York.

Examples

```
## load a fitted model object
data(tidfit)
data(tidfitmean)

## get trends

# setup month, year categories
mobrks <- list(c(1, 2, 3), c(4, 5, 6), c(7, 8, 9), c(10, 11, 12))
yrbrks <- c(1973, 1985, 1994, 2003, 2012)
molabs <- c('JFM', 'AMJ', 'JAS', 'OND')
yrlabs <- c('1974-1985', '1986-1994', '1995-2003', '2004-2012')

wrtdstrnd_sk(tidfit, mobrks, yrbrks, molabs, yrlabs)
wrtdstrnd_sk(tidfitmean, mobrks, yrbrks, molabs, yrlabs)
```

---

wtsplot *Plot the weights for an observation*

Description

Create several plots showing the weights used to fit a model for a single observation.

Usage

```
wtsplot(dat_in, ...)

## Default S3 method:
wtsplot(dat_in, ref = NULL, wins = list(0.5, 10, NULL),
  min_obs = TRUE, slice = FALSE, dt_rng = NULL, pt_rng = c(1, 12),
```

```
  col_vec = NULL, col_lns = NULL, alpha = 1, as_list = FALSE, ...)

## S3 method for class 'tidal'
wtsplot(dat_in, ...)

## S3 method for class 'tidalmean'
wtsplot(dat_in, ...)
```

Arguments

dat_in	input tidal object
...	arguments passed to other methods
ref	chr string indicating the date at the center of the weighting window. Must be in the format 'YYYY-mm-dd' which is passed to as.Date. The closest observation is used if the actual is not present in the data. Defaults to the mean date if not supplied.
wins	list with three elements passed to getwts indicating the half-window widths for day, year, and salinity/flow
min_obs	logical to use window widening if less than 100 non-zero weights are found, passed to getwts
slice	logical indicating if only weights bounded by the year window (i.e., the limiting window for the combined weights) are shown, passed to getwts
dt_rng	Optional chr string indicating the date range for all plots except seasonal (day) weights. Must be two values in the format 'YYYY-mm-dd' which is passed to as.Date.
pt_rng	numeric vector of two elements indicating point scaling for all weights in the plot of salinity/flow vs time.
col_vec	chr string of plot colors to use, passed to gradcols and scale_colour_gradientn for weight shading. The last value in the vector is used as the line color if col_lns = NULL. Any color palette from RColorBrewer can be used as a named input. Palettes from grDevices must be supplied as the returned string of colors for each palette.
col_lns	chr string of line color in plots
alpha	numeric value from zero to one indicating transparency of points and lines
as_list	logical indicating if plots should be returned in a list

Details

Create diagnostic plots to view the effects of different weighting windows on model predictions. The plots illustrate the weights that are used when fitting a weighted regression in reference to a single observation. The process is repeated for all observations when the entire model is fit. Five plots are produced by the function, each showing the weights in relation to time and the selected observation (i.e., center of the weighting window). The top plot shows salinity/flow over time with the points colored and sized by the combined weight vector. The remaining four plots show the weights over time for each separate weighting component (months/days, year, and salinity/flow) and the final combined vector.

Value

A combined ggplot object created using grid.arrange. A list with elements for each individual plot will be returned if as_list = TRUE.

See Also

getwts

Examples

```
## load a fitted tidal object
data(tidfit)

## plot using defaults,
wtsplot(tidfit)

## change the defaults
wtsplot(tidfit, ref = '2000-01-01', wins = list(0.5, 15, Inf),
 dt_rng = c('1990-01-01', '2010-01-01'),
 pt_rng = c(3, 8), col_vec = c('lightgreen', 'lightblue', 'purple'),
 alpha = 0.7)
```

Index