Package 'baytrends'

March 14, 2019

Type Package

Title Long Term Water Quality Trend Analysis

Description Enable users to evaluate long-term trends using a Generalized Additive Modeling (GAM) approach. The model development includes selecting a GAM structure to describe nonlinear seasonally-varying changes over time, incorporation of hydrologic variability via either a river flow or salinity, the use of an intervention to deal with method or laboratory changes suspected to impact data values, and representation of left- and interval-censored data. The approach has been applied to water quality data in the Chesapeake Bay, a major estuary on the east coast of the United States to provide insights to a range of management- and research-focused questions.

Author Rebecca Murphy, Elgin Perry, Jennifer Keisman, Jon Harcum, Erik W Leppo

Version 1.1.0

Date 2019-03-14

Maintainer Erik Leppo <Erik.Leppo@tetratech.com>

Depends R (>= 3.2.0), lubridate, mgcv

License GPL-3

LazyData TRUE

RoxygenNote 6.1.1

Encoding UTF-8

Suggests devtools, fitdistrplus, imputeTS, knitr, nlme, pander, readxl, rmarkdown, sessioninfo, testthat

Imports XML, dataRetrieval, digest, gdata, memoise, methods, plyr, survival, zCompositions

URL https://github.com/tetratech/baytrends

VignetteBuilder knitr

Collate 'a1_smwrQW_qw-class.R' 'a2_smwrQW_lcens-class.R' 'a3_smwrQW_mcens-class.R' 'analysisOrganizeData.R' 'appendDateFeatures.R' 'as.lcens.R' 'as.mcens.R' 'as.qw.R' 'baytrends.R' 'checkRange.R' 'chkParameter.R' 'closeOut.R'

'createResiduals.R' 'data.R' 'detrended.flow.R'
'detrended.salinity.R' 'expectMaxFunctions.r' 'findFile.R'
'flwAveragePred.R' 'fmtPval.R' 'gamDiff.R' 'gamPlotCalc.r'
'gamPlotDisp.R' 'gamTables.R' 'gamTest.r' 'getUSGSflow.R'
'headers2.R' 'imputeCensored.R' 'initializeResults.r'
'layerAggregation.R' 'loadData.R' 'loadExcel.R' 'loadModels.R'
'loadModelsResid.R' 'mergeFlow.R' 'mergeSalinity.R'
'qw.export.R' 'qw.import.R' 'reAttDF.R' 'saveDF.R'
'seasAdjflow2.R' 'selectData.R' 'smwrBase_baseDay.R'
'smwrBase_baseDay2decimal.R' 'smwrBase_dectime.R'
'smwrBase_dectime2Date.R' 'smwrBase_eventProcessing.R'
'smwrBase_fillMissing.R' 'smwrBase_group2row.R'
'smwrBase_importRDB.R' 'smwrBase_isLike.R' 'smwrBase_na2miss.R'
'smwrBase_z%cn%.R' 'smwrGraphs_addCaption.R'
'smwrGraphs_colorPlot.R' 'smwrGraphs_datePretty.R'
'smwrGraphs_lineWt.R' 'smwrGraphs_linearPretty.R'
'smwrGraphs_month.USGS.R' 'smwrGraphs_namePretty.R'
'smwrGraphs_numericData.R' 'smwrGraphs_refLine.R'
'smwrGraphs_renderPretty.R' 'smwrGraphs_setColor.R'
'smwrGraphs_setExplan.R' 'smwrGraphs_setGD.R'
'smwrGraphs_setMargin.R' 'smwrGraphs_setMultiPlot.R'
'smwrGraphs_setPage.R' 'smwrGraphs_setPlot.R'
'smwrGraphs_strip.blanks.R' 'smwrGraphs_timePlot.R'
'smwrGraphs_timePretty.R' 'smwrGraphs_transData.R'
'smwrGraphs_xyPlot.R' 'smwrQW_censoring.R' 'smwrQW_is.na.R'
'smwrQW_mcenKM.R' 'smwrQW_mcenMLE.R' 'smwrQW_mcenROS.R'
'smwrQW_mdlMLE.R' 'smwrQW_mdlROS.R' 'smwrQW_pcodeNWISqw.R'
'smwrQW_x_Arith-censored.R' 'smwrQW_x_Math-censored.R'
'smwrQW_x_View.r' 'smwrQW_x_add.R' 'smwrQW_x_as.character.R'
'smwrQW_x_as.data.frame.R' 'smwrQW_x_as.double.R'
'smwrQW_x_c.R' 'smwrQW_x_censorlevels.R' 'smwrQW_x_censpp.R'
'smwrQW_x_convert2qw.R' 'smwrQW_x_convertFqw.R'
'smwrQW_x_dlimit.R' 'smwrQW_x_format.R'
'smwrQW_x_importNWISqw.R' 'smwrQW_x_importQW.R'
'smwrQW_x_imputeLessThans.R' 'smwrQW_x_length.R'
'smwrQW_x_makeColNames.R' 'smwrQW_x_mdlKM.R'
'smwrQW_x_mdlKMstats.R' 'smwrQW_x_mean.qw.R'
'smwrQW_x_meanStats.R' 'smwrQW_x_medianStats.R'
'smwrQW_x_print.censMCT.R' 'smwrQW_x_print.censQuantile.R'
'smwrQW_x_print.censReg.R' 'smwrQW_x_print.censStats.R'
'smwrQW_x_print.summary.censReg.R' 'smwrQW_x_printData.R'
'smwrQW_x_quantile.R' 'smwrQW_x_qw2mcens.R'
'smwrQW_x_qwCoalesce.R' 'smwrQW_x_ratio.R' 'smwrQW_x_rbindQW.R'
'smwrQW_x_readNWQLdl.R' 'smwrQW_x_residuals.censReg.R'
'smwrQW_x_sdlFill.R' 'smwrQW_x_show-censored.R'
'smwrQW_x_sort.censored.R' 'smwrQW_x_splitQual.R'
'smwrQW_x_str.R' 'smwrQW_x_subset.R' 'smwrQW_x_summary.R'
'smwrQW_x_summary.censReg.R' 'smwrQW_x_vcov.censReg.R'

'smwrQW_x_z%eq%.R' 'smwrStats_rmse.R' 'smwrStats_vif.R'
'supportFunctions.R' 'zzz.R'

NeedsCompilation no

Repository CRAN

Date/Publication 2019-03-14 18:23:27 UTC

R topics documented:

analysisOrganizeData . . . 3
baytrends . . . 6
closeOut . . . 7
createResiduals . . . 7
dataCensored . . . 8
detrended.flow . . . 9
detrended.salinity . . . 11
gamDiff . . . 13
gamPlotDisp . . . 15
gamTest . . . 16
impute . . . 17
layerLukup . . . 18
loadData . . . 19
loadExcel . . . 20
loadModels . . . 21
loadModelsResid . . . 22
parameterList . . . 22
qw.export . . . 23
qw.import . . . 24
rbindQW . . . 25
sal . . . 26
saveDF . . . 26
selectData . . . 27
stationMasterList . . . 30
usgsGages . . . 31

Index **32**

`analysisOrganizeData` *Analysis Organization & Data Preparation*

Description

This function assesses the user supplied specifications and prepares data for analysis. In those cases where the user doesn't supply a needed specification, a basic option is supplied by this function.

Usage

```
analysisOrganizeData(df, analySpec = list(), reports = c(0, 1, 2, 3),
  parameterList = NA, stationMasterList = NA, layerLukup = NA)
```

Arguments

df	Data frame of water quality data
analySpec	Specifications for analysis
reports	Optional reports about parameters, layers and stations [default = c(0,1,2,3)]
parameterList	User-supplied parameter list [default = NA]
stationMasterList	User-supplied station list [default = NA]
layerLukup	User-supplied layer lookup list [default = NA]

Details

The supplied data frame, df, is a data frame with the variables station, date, and layer included along with multiple additional columns for a variety of water quality variables structured as "qw" objects. An example data frame, dataCensored, is included with baytrends as an example.

The argument, analySpec, is a list that includes basic specfications for performing GAM analyses. The components in analySpec are identified below. The user may create analySpec (which can include all or some of the below components; and pass the user-supplied analySpec to this function. Or, the user may accept the default argument and allow analysisOrganizeData to create and return analySpec. If the user passes a user-supplied analySpec, then analysisOrganizeData will "fill in" required arguments not provided by the user. The user can also adjust analySpec after it is returned from analysisOrganizeData although requirements for down selecting the data frame, df, or aggregating data by layer would need to be passed to analysisOrganizeData.

The default setting for the argument report is to provide tabular summary reports about the parameters, stations, and layers to be analyzed. Setting report=NA will suppress the tabular summary reports.

The user can supply their own parameterList, stationMasterList, or layerLukup data sets; or the user can use the example data frames included with baytrends.

The following steps are performed by analysisOrganizeData:

1) Review user supplied specifications through the analySpec argument. Fill in with default values. For example, if the user doesn't specify a list of stations, then all stations identified in the data set stationMasterList are used. Some other default values include the following: date range (1/1/1984-present), parameter list (all parameters in data set parameterList), layers (all layers in data set layerLukup), layer aggregation option (0, no aggregation), minimum number of observations to perform a GAM analysis (60), GAM formulas (Linear Trend with Seasonality, Non-linear Trend with Seasonality, and Non-linear trend with Seasonality (plus Interactions)), GAM alpha level for plots and output (0.05), periods for multi-time period analyses using GAM (Full Record, 1999/00-Present, and 2005/06-Present), and seasons for sub-annual analyses using GAM (All, Spring1, Spring2, Summer1, Summer2, SAV1, SAV2, Winter, and Fall.)

2) Based on the settings in analySpec, the data frame df, is down selected based on parameters, stations, dates, and layers. A dependent variable list (depVarList) is created that includes variable

descriptions, units, and log transformation selections. A station list (stationList) is created that includes the station ID, a selected USGS gage for correlating flow, and latitude/longitude.

3) Aggregate data layers. If analySpec$layerAggOption is equal to 0, then there is no aggregation. The analySpec$layerAggOption of 1 would result in averaging (mean) surface and above pycnocline data. In this example, records with layer = "S" and layer = "AP" are relabeled as "SAP". Other layerAggOption values are 2) "B"&"BP"; 3) "S"&"AP" and "B"&"BP"; 4) all layers; and 5) "S"&"B", respectively. A layer list (layerList) is created and returned.

4) Data are then averaged (mean) by date.

5) Date features are added. Columns for year, day of year (doy), decimal year (dyear), and month are added based on date. Note that the doy is based on a 366 day calendar regardless of leap year.

6) Reports on the number of records (0), parameters (1), layers (2) and stations (3) can be controlled with the reports option.

Value

Returns a list. Use dfr[["df"]] and dfr[["analySpec"]] to extract updated data frame and updated (or created) analySpec. analySpec is a list that includes the following components:

analyTitle - Analysis trend title

parameterFilt - Parameter filter used for data down selection

stationFilt - Station filter used for data down selection

dateFilt - Date filter for data down selection

setTZ - time zone (default = "America/New_York")

layerFilt - Layer filter

layerAggOption - Layer averaging option (default = 0)

obsMin - Minimum number of observations required to allow GAM analysis to proceed for a specific station, parameter, and layer combination (default = 60).

gamAlpha - Alpha level used GAM analyses (default = 0.05).

censorTrim - Values to apply for trimming data due to too much censoring (default = c(0.5, 0.4)).

gamModels - model formulations

gamDiffPeriods - list of time periods (years) used for computing changes (differences). The default options are: full record, 1999/00-present, and 2005/06-present.

gamDiffSeasons - list of seasons used for sub-annual analyses. The default options include the following: All (months 1:12), Spring1 (3:5), Spring2 (months: 4:6)), Summer1 (months: 6:9)), Summer2 (months: 7:9)), SAV1 (months: 4:10)), SAV2 (months: 3:5,9:11)), Winter (months: 1:2)), and Fall (months: 10:12))).

gamPenalty - allow the user to set the mgcv::gam select argument to TRUE, FALSE, or baytrend algorithm default (default = NA). When the default option is specified, then the mgcv::gam select argument is set to FALSE when none of the data are censored; otherwise (when some censoring exists in the data set) the select argument is set to TRUE

gamPenaltyCrit - edf and F-stat values used to flag ANOVA table results (default = c(1, 9e9))

gamCoeffDeltaMaxCrit - convergence criteria for expectation maximization (default = 1e-6)

gamFlw_Sal.Wgt.Perc - percentiles of flow [or salinity] to use for computing flow [salinity] averaged result (default = c(0.05, 0.25, 0.50, 0.75, 0.95))

gamLegend - default settings for gam figure legend

idVar - primary key for data frame returned as df

depVarList - data frame of dependent variables (useful for setting up gam analyses in for loops)

stationList - data frame of stations (useful for setting up gam analyses in for loops)

layerList - data frame of layers (useful for setting up gam analyses in for loops)

Examples

```
# run analysis relying on default specifications, examine analySpec for
# default options
dfr <- analysisOrganizeData(dataCensored)
df        <- dfr[["df"]]
analySpec <- dfr[["analySpec"]]

# analyze bottom dissolved oxygen at 2 stations
analySpec <-list()
analySpec$parameterFilt <- c('do')
analySpec$layerFilt     <- c('B')
analySpec$stationFilt   <- c('CB3.3C', 'CB5.4')
dfr <- analysisOrganizeData(dataCensored, analySpec)
df        <- dfr[["df"]]
analySpec <- dfr[["analySpec"]]
```

`baytrends` *baytrends: Long Term Water Quality Trend Analysis*

Description

The baytrends package was developed to enable users to evaluate long-term trends in the Chesapeake Bay using a Generalized Additive Modeling (GAM) approach. The model development includes selecting a GAM structure to describe nonlinear seasonally-varying changes over time, incorporation of hydrologic variability via either a river flow or salinity, the use of an intervention to deal with method or laboratory changes suspected to impact data values, and representation of left- and interval-censored data. This approach, which is fully transferable to other systems, allows for Chesapeake Bay water quality data to be evaluated in a statistically rigorous, yet flexible way to provide insights to a range of management- and research-focused questions.

closeOut *Document Processing Time and Other Session Time*

Description

Document Processing Time and Other Session Time

Usage

```
closeOut(timeProcess = TRUE, contInfo = TRUE, sessInfo = TRUE)
```

Arguments

timeProcess	Processing time
contInfo	Contact information
sessInfo	Session information

Value

Reports out processing time, contact information and session information

Examples

```
closeOut()
```

createResiduals *Calculate GAM residuals*

Description

Use GAM analysis to compute residuals. Relies on mgcv::gam to perform general additive model.

Usage

```
createResiduals(df, dep, residualModel = "doy_flw_sal",
  analySpec = analySpec, gamTable = FALSE, gamPlot = FALSE,
  flow.detrended = NA, salinity.detrended = NA, width = 10,
  height = 3.5, folder_r = "pltResiduals", ProjRoot)
```

Arguments

df	data frame
dep	variable
residualModel	which gam formula is used to compute . Default: 'doy_flw_sal'.
analySpec	analytical specifications
gamTable	gam table setting (set to FALSE to turn off table output)
gamPlot	gam plot setting (set to FALSE to turn off plotting)
flow.detrended	data generated by detrended.flow. Default = NA.
salinity.detrended	data generated by detrended.flow. Default = NA.
width	width of png figure (inches). Default = 10
height	height of png figure (inches). Default = 3.5
folder_r	folder to store residual plots
ProjRoot	Root folder for project.

Value

Returns a vector with results

---

dataCensored *Chesapeake Bay Program Monitoring Data, 1985-2016*

Description

Selected 1985-2016 data for eight (8) stations from the Chesapeake Bay Monitoring Program. Water quality variables are stored as class qw that allows for left- and interval-censored data.

Usage

```
dataCensored
```

Format

A data frame with 13,062 rows and 17 variables:

station station identifier

date sample date

layer sample layer

secchi Secchi depth [m]

salinity Salinity [ppt]

do Dissolved Oxygen [mg/L]

wtemp Water Temperature [deg C]

chla Corrected Chlorophyll-a [ug/L]

tn Total Nitrogen [mg/L]

tp Total Phosphorus [mg/L]

tss Total Suspended Solids [mg/L]

din Dissolved Inorganic Nitrogen [mg/L]

po4 Orthophosphorus [mg/L]

tdn Total Dissolved Nitrogen [mg/L]

tdp Total Dissolved Phosphorus [mg/L]

nh4 Ammonium [mg/L]

no23 Nitrite + Nitrate [mg/L]

`detrended.flow` *Create Seasonally Detrended Flow Data Set*

Description

This function creates a seasonally detrended flow data set for selected USGS gages. The created data set is used to support application of GAMs that include a hydrologic term as one of the independent variables. The output from this function should be stored as an .rda file for repeated use with baytrends.

Usage

```
detrended.flow(usgsGageID, siteName, yearStart, yearEnd,
  dvAvgWinSel = c(1, 5, 10, 15, 20, 30, 40, 50, 60, 90, 120, 150, 180,
  210), dvAvgWgtSel = "uniform", dvAvgSidesSel = 1, lowess.f = 0.2,
  span = 10, max.fill = 10)
```

Arguments

`usgsGageID`	USGS GageIDs (e.g., "01491000")
`siteName`	USGS SiteName (only used for plots)
`yearStart`	start year (recommended as at least one year before corresponding water quality data set)
`yearEnd`	end year
`dvAvgWinSel`	Averaging window (days) for smoothing the residuals of the seasonally adjusted daily flow values [default = c(1, 5, 10, 15, 20, 30, 40, 50, 60, 90, 120, 150, 180, 210)]
`dvAvgWgtSel`	Averaging method ("uniform", "weighted", or "centered") for creating weights. If using "weighted" then use dvAvgSidesSel=1. If using "centered" then use dvAvgSidesSel=2. [default = "uniform"]
`dvAvgSidesSel`	If dvAvgSidesSel=1 only past values are used, if dvAvgSidesSel=2 then values are centered around lag 0. [default = 1]

`lowess.f`	lowess smoother span applied to computed standard deviation (see Details). This gives the proportion of points which influence the smooth at each value. Larger values give more smoothness. [default = 0.2]
`span`	maximum number of observations on each side of range of missing values to use in filling in data [default = 10]
`max.fill`	maximum gap to fill in [default = 10]

Details

This function returns a list of seasonally detrended flow and companion statistics; and relies on USGS' dataRetrieval package to retrieve daily flow data.

It is the user responsibility to save the resulting list as **flow.detrended** for integration with baytrends.

For the purposes of baytrends, it is expected that the user would identify all USGS gages that are expected to be evaluated so that a single data file is created. To best match up with water quality data, we recommend retrieving flow data for one year prior to the first year of water quality data. This allows for creating a time-averaged flow data set and not loose the first few months of water quality data due to lack of matching flow data. Data retrievals should also be made in light of the time needed by the USGS to review and approve their flow records.

After retrieval, the following computation steps are performed to create a data frame for each USGS gage (the data frame naming convention is **qNNNNNNNN** where NNNNNNNN is the USGS gage ID):

1) The daily flow data are converted to cubic meters per second [cms] and stored as the variable **q**.

2) The day of year (**doy**) is added to the data set. We use a 366 day calendar regardless of leap year.

3) The Log (ln) flow is computed and stored as **LogQ**.

4) A seasonal GAM, i.e., gamoutput <- gam(LogQ ~ s(doy, bs='cc')) is evaluated and the predicted values stored as **qNNNNNNNN.gam**.

5) The GAM residuals, i.e., "residuals(gamoutput)" are extracted and stored as the variable, **d1**.

6) Based on the specifications for dvAvgWinSel, dvAvgWgtSel, and dvAvgSidesSel, the values of **d1** are time averaged and additional variables **dxxx** are added to the data frame where xxx corresponds to list of averaging windows specified in dvAvgWinSel. These values of **dxxx** are used in GAMs that include a hydrologic independent variable.

After the above data frame is created, the following four (4) additional data frames are created for each USGS gage and combined into a list named **qNNNNNNNN.sum**:

mean – For each doy (i.e., 366 days of year), the mean across all years for each value of d in the above data frame, qNNNNNNNN.

sd – For each doy (i.e., 366 days of year), the standard deviation across all years for each value of d in the above data frame, qNNNNNNNN.

nobs – For each doy (i.e., 366 days of year), the number of observations across all years for each value of d in the above data frame, qNNNNNNNN.

lowess.sd – Lowess smoothed standard deviations. (These values are used for computing confidence intervals in the flow averaged GAM.)

The process of creating the above data frame, **qNNNNNNNN**, and list, **qNNNNNNNN.sum**, is repeated for each USGS gage and combined together in a single list. The beginning of the list includes meta data documenting the retrieval parameters.

This function can be used in conjunction with an RMD file to knit (create) a report (DOCX or HTML).

Value

Returns a list of seasonally detrended flow data. You should save the resulting list as flow.detrended for use with baytrends. This function also creates diagnostic plots that can be saved to a report when this function is called from an .Rmd script.

Examples

```
## Not run:
# Define Function Inputs
usgsGageID    <- c("01491000", "01578310")
siteName      <- c("Choptank River near Greensboro, MD",
                   "Susquehanna River at Conowingo, MD")
yearStart     <- 1983
yearEnd       <- 2016
dvAvgWinSel   <- c(1, 5, 10, 15, 20, 30, 40, 50, 60, 90, 120, 150, 180, 210)
dvAvgWgtSel   <- "uniform"
dvAvgSidesSel <- 1
lowess.f      <- 0.2

# Run Function
flow.detrended <- detrended.flow(usgsGageID, siteName, yearStart, yearEnd
                                 , dvAvgWinSel, dvAvgWgtSel, dvAvgSidesSel
                                , lowess.f)

## End(Not run)
```

---

detrended.salinity *Create Seasonally Detrended Salinty Data Set*

Description

This function creates a seasonally detrended salinity data set for selected stations. The created data set is used to support application of GAMs that include a hydrologic term as one of the independent variables. The output from this function should be stored as an .rda file for repeated use with baytrends.

Usage

```
detrended.salinity(df.sal, dvAvgWinSel = 30, lowess.f = 0.2,
  minObs = 40, minObs.sd = 10)
```

Arguments

df.sal	data frame with salinty data (required variables in data frame are: station, date, layer, and salinity)
dvAvgWinSel	Averaging window (days) selection for pooling data to compute summary statistics
lowess.f	lowess smoother span applied to computed standard deviation (see Details). This gives the proportion of points which influence the smooth at each value. Larger values give more smoothness.
minObs	Minimum number of observations for performing analysis (default is 40)
minObs.sd	Minimum number of observations in averaging window for calculation of the standard deviation (default is 10)

Details

This function returns a list of seasonally detrended salinity and companion statistics; and relies on a user supplied data frame that contains the following variables: station, date, layer, and salinity. See structure of sal data in example below.

It is the user responsibility to save the resulting list as **salinity.detrended** for integration with baytrends.

For the purposes of baytrends, it is expected that the user would identify a data set with all salinity data that are expected to be evaluated so that a single data file is created. The following computation steps are performed:

1) Extract the list of stations, minimum year, and maximum year in data set. Initialize the **salinity.detrended** list with this information along with meta data documenting the retrieval parameters.

2) Downselect the input data frame to only include data where the layer is equal to 'S', 'AP', 'BP' or 'B'.

3) Average the 'S' and 'AP' salinity data; and the 'B' and 'BP salinity data together to create average salinity values for SAP (surface and above pycnocline) and BBP (bottom and below pycnocline), respectively. These values are stored as the variables, **salinity.SAP** and **salinity.BBP** together with the **date** and day of year (**doy**) in a data frame corresponding to the station ID.

4) For each station/layer combination with atleast **minObs** observations, a seasonal GAM, i.e., gamoutput <- gam(salinity ~ s(doy, bs='cc')) is evaluated and the predicted values stored in the above data frame as **salinity.SAP.gam** and **salinity.BBP.gam**.

5) The GAM residuals, i.e., "residuals(gamoutput)" are extracted and stored as the variable, **SAP** or **BBP** in the above data frame. (These are the values that are used for GAMs that include salinity.)

6) After the above data frame is created and appended to the list **salinity.detrended**, the following four (4) additional data frames are created for each station.

mean – For each doy (i.e., 366 days of year), the mean across all years for each value of d. Since samples are not collected on a daily basis it is necessary to aggregate data from within a +/- one-half of **dvAvgWinSel**-day window around d. (This includes wrapping around the calendar year. That is, the values near the beginning of the year, say January 2, would include values from the last part of December and the first part of January. The variables in the mean data frame are doy, SAP, and BBP.

sd – For each doy (i.e., 366 days of year), the standard deviation across all years for each value of d. (See mean calculations for additional details.)

nobs – For each doy (i.e., 366 days of year), the number of observations across all years for each value of d. (See mean calculations for additional details.)

lowess.sd – Lowess smoothed standard deviations. It is noted that some stations do not include regular sampling in all months of the year or for other reasons have few observations from which to compute standard deviations. Through visual inspection of plots, we found that the standard deviation could become unstable when the number of observations is small. For this reason, when the number of observations is less than **minObs.sd**, the corresponding value of lowess.sd is removed and interpolated from the remaining observations.

The above four data frames (mean, sd, nobs, and lowess.sd) are created, they are added to a list using a **station.sum** naming convention and appended to the list **salinity.detrended**.

Value

Returns a list of seasonally detrended salinity data. You should save the resulting list as salinity.detrended for use with baytrends. This function also creates diagnostic plots that can be saved to a report when this function is called from an .Rmd script.

Examples

```
## Not run:
# Show Example Dataset (sal)
str(sal)

# Define Function Inputs
df.sal        <- sal
dvAvgWinSel   <- 30
lowess.f      <- 0.2
minObs        <- 40
minObs.sd    <- 10

# Run Function
salinity.detrended <- detrended.salinity(df.sal, dvAvgWinSel,
                                 lowess.f, minObs, minObs.sd)

## End(Not run)
```

`gamDiff` *Compute an estimate of difference based on GAM results*

Description

Compute an estimate of difference based on GAM results

Usage

```
gamDiff(gamRslt, iSpec, analySpec, base.yr.set = NA, test.yr.set = NA,
  doy.set = NA, alpha = 0.05, flow.detrended = NA,
  salinity.detrended = NA)
```

Arguments

gamRslt	output from gam model
iSpec	data set specifications (see details for required content)
analySpec	analytical specifications
base.yr.set	vector of years used for baseline period
test.yr.set	vector of years used for test period
doy.set	vector of days used to establish sub-annual analyses (see details)
alpha	alpha level for computing confidence intervals
flow.detrended	data generated by detrended.flow. Default = flow.detrended.
salinity.detrended	data generated by detrended.salinity. Default = detrended.salinity.

Details

iSpec is a list containing information about the date range and transformations. Specifically, iSpec must include iSpec$yearBegin, iSpec$yearEnd, iSpec$centerYear corresponding to beginning year of record, ending year of record and centering year. Also, iSpec must include iSpec$transform and iSpec$logConst. (See online help for selectData for more information on these values.)

base.yr.set and test.yr.set represent two time periods used to compare differences. For example, base.yr.set=c(1999,2000) and test.yr.set=c(2013,2014) would compare GAM predictions from 1999-2000 versus 2013-2014. There is no particular limit to the number of years included in the specification for base.yr.set and test.yr.set. For example, a user could specify c(2001:2002,2004) to use the years 2001, 2002, and 2004, skipping 2003 because 2003 was an abnormal year (particularly wet, particularly dry, hurricanes, etc.).

base.yr.set and test.yr.set must be within the years specified by the range from iSpec$yearBegin to iSpec$yearEnd (inclusive). If not, this function defaults to using the first two years (or last two years) of record. If base.yr.set and test.yr.set are left to their default values of NA, then the first two and last two years will be used

doy.set represents the days of year for which GAM predictions are made and used to compute base.yr and test.yr means. For example doy.set= c(15, 46, 75, 106, 136, 167, 197, 228, 259, 289, 320, 350) would result in the 15th of each month being used in the analysis; whereas doy.set= c(15, 46, 75) would just use Jan-15, Feb-15, and Mar-15. (Keep in mind that this package uses a 366 day calendar every year such that Mar-1 is always day 61, regardless of leap year.) If doy.set is left to the default value of NA, then c(15, 46, 75, 106, 136, 167, 197, 228, 259, 289, 320, 350) is used.

The baseDay function has been added to this package from the smwrBase package.

Value

Returns a nest list that includes the base and test years, doys, period means in analyzed units, period means in observed units, percent change, difference estimate, difference estimate in observed units, standard error, confidence intervals, t statistic, p value, and alpha level. The alpha level corresponds to the confidence intervals. The first list (gamDiff.regular) uses the computed model to estimate differences and is applicable for GAM formulas that do not involve an intervention term. The second list (gamDiff.adjusted) performs computations by projecting the most recent intervention (e.g., the current lab method) to all time periods.

Examples

```
# run analysisOrganizeData function to create the list analySpec
dfr <- analysisOrganizeData (dataCensored, report=NA)
df        <- dfr[["df"]]
analySpec <- dfr[["analySpec"]]

# set GAM models to just one model
analySpec$gamModels <- list(
  list(option=2, name= "Non-linear trend with Seasonality (plus Interactions)",
      model= "~ cyear + s(cyear) + s(doy,bs='cc')+ ti(cyear,doy,bs=c('tp','cc'))", deriv=FALSE))

# run GAM for a single water quality variable, station and layer
gamResult <- gamTest(df, 'tn', 'CB5.4', 'S', analySpec=analySpec)

# use gamDiff to replicate estimates of change calculated in the above
gamDiff(gamRslt=gamResult[["gamOutput2"]]$gamRslt,
        iSpec=gamResult$iSpec, analySpec=analySpec,
        base.yr.set = NA, test.yr.set = NA,
        doy.set = NA, alpha = 0.05)

# use gamDiff to calculate changes from 2005/06 to 2013/14
gamDiff(gamRslt=gamResult[["gamOutput2"]]$gamRslt,
        iSpec=gamResult$iSpec, analySpec=analySpec,
        base.yr.set = c(2004:2005), test.yr.set = c(2013:2014),
        doy.set = NA, alpha = 0.05)
```

`gamPlotDisp` *Plot censored gam fits vs. time*

Description

Plot censored gam fits vs. time

Usage

```
gamPlotDisp(gamResult = gamResult, analySpec = analySpec,
  fullModel = 2, seasAvgModel = 2, seasonalModel = 2,
  diffType = "regular", obserPlot = TRUE, interventionPlot = TRUE,
```

```
  seasAvgPlot = TRUE, seasAvgConfIntPlot = TRUE,
  seasAvgSigPlot = TRUE, fullModelPlot = TRUE, seasModelPlot = TRUE,
  BaseCurrentMeanPlot = TRUE, adjustedPlot = FALSE)
```

Arguments

gamResult	output from procedure gamTest
analySpec	analytical specifications
fullModel	GAM # for displaying full GAM (e.g., 0, 1, 2)
seasAvgModel	GAM # for displaying seasonally average GAM
seasonalModel	GAM # for displaying seasonal GAM
diffType	plot predicted baseline mean ('regular') or adjusted baseline mean ('adjusted')
obserPlot	logical field indicating whether to plot observations
interventionPlot	logical field indicating whether to plot interventions (e.g., method changes)
seasAvgPlot	logical field indicating whether to plot seasonal average GAM
seasAvgConfIntPlot	logical field indicating whether to plot confidence interval for seasonal average GAM
seasAvgSigPlot	logical field indicating whether to plot significant increasing and decreasing trends for seasonal average GAM
fullModelPlot	logical field indicating whether to plot full GAM
seasModelPlot	logical field indicating whether to plot seasonal GAM
BaseCurrentMeanPlot	logical field indicating whether to plot baseline and current mean
adjustedPlot	logical field indicating whether to plot adjusted model

---

gamTest *Perform GAM analysis*

Description

Perform GAM analysis. Relies on mgcv::gam to perform general additive model.

Usage

```
gamTest(df, dep, stat, layer = NA, analySpec, gamTable = TRUE,
  gamPlot = 10, gamDiffModel = NA, flow.detrended = NA,
  salinity.detrended = NA)
```

Arguments

df	data frame
dep	dependent variable
stat	station
layer	layer (optional)
analySpec	analytical specifications
gamTable	gam table setting (set to FALSE to turn off table output)
gamPlot	gam plot setting (set to FALSE to turn off plotting)
gamDiffModel	GAM model(s) used for computing differences on sub-annual/multi-period basis
flow.detrended	data generated by detrended.flow. Default = NA.
salinity.detrended	data generated by detrended.flow. Default = NA.

Details

The baseDay function has been added to this package from the smwrBase package.

Value

Returns a list with results

Examples

```
# specify parameter and station to analyze
dep        <- 'secchi'
stat       <- 'CB5.4'
layer      <- 'S'

#Using gamTest
dfr <- analysisOrganizeData (dataCensored)
df        <- dfr[[1]]
analySpec <- dfr[[2]]
gamResult <- gamTest(df, dep, stat, layer, analySpec=analySpec)
```

---

impute *Impute Censored Values*

Description

Impute value for multiply censored data.

Usage

```
impute(x, imputeOption = "mid")
```

Arguments

x	vector of type baytrends::qw
imputeOption	imputation method [default= "mid"], valid impute options are "lower", "upper", "mid", "norm", "lnorm"

Details

The imputeOption values of lower, upper and mid impute the lower limit, upper limit, and midpoint between the lower and upper limit. In the context of typical water quality data, these options would be eqivalent to zero, detection limit and 1/2 detection limit substitution. Options for substituting the normal ["norm"] or lognormal ["lnorm"] expectation can also be used.

Value

vector where x is transformed into a simple numeric variable

Examples

```
x  <- dataCensored[1:20,"tdp"]
x.lower <- impute(x,'lower')
x.mid   <- impute(x,'mid')
x.upper <- impute(x,'upper')
x.norm  <- impute(x,'norm')
x.lnorm <- impute(x,'lnorm')
```

---

layerLukup *Layer List*

Description

A lookup table of layer abbreviations and the corresponding names.

Usage

```
layerLukup
```

Format

A data frame with 10 rows and 3 variables:

layers Layer codes

order Analysis order

name Layer name

loadData *Load/Clean CSV and TXT Data File*

Description

Load and clean comma delimited (*.csv) or tab delimited (*.txt) file and perform some rudimentary data cleaning.

Usage

```
loadData(file = NA, folder = NA, pk = NA, remDup = TRUE,
  remNAcol = TRUE, remNArow = TRUE, convDates = TRUE,
  tzSel = "America/New_York", commChar = "#", naChar = NA)
```

Arguments

file	file (can use wildcards, e.g., "*.csv")
folder	folder (i.e., directory to look in, can use relative path)
pk	vector of columns that form the primary key for data set
remDup	logical field indicating whether duplicate rows are deleted
remNAcol	logical field indicating whether columns with all NA are deleted
remNArow	logical field indicating whether rows with all NA are deleted
convDates	vector or logical field indicating whether date-like columns should be converted to POSIXct format (see details)
tzSel	time zone to use for date conversions (default: "America/New_York")
commChar	character for comment line to be skipped
naChar	characters to treat as NA

Details

This function reads in a single comma delimited (*.csv) or tab delimited (*.txt) file using either utils::read.table or utils::read.csv based on the file extension. The user can use the wildcard feature for the file argument (e.g., file='*.csv') and the function will identify the most recently modified csv or txt file in the folder for importing.

Some specific features of this function include the following:

1. Leading '0's in character strings that would otherwise be trimmed and treated as numeric variables (e.g., USGS flow gages, state and county FIPS codes) are preserved. To effectively use this functionality, data maintained in a spreadsheet would be enclosed in quotes (e.g., "01578310"). When exported to csv or txt files the field would be in triple quotes (e.g., """01578310"""). Any column read in as integer is converted to numeric.

2. Rows and columns with no data (i.e., all NA) are deleted unless default settings for remNAcol and remNArow are changed to FALSE.

3. Completely duplicate rows are deleted unless default setting for remDup is changed to FALSE.

4. Rows beginning with '#' are skipped unless commChar set to ""

5. If a primary key (either single or multiple columns) is selected, the function enforces the primary key by deleting duplicate entries based on the primary key. Columns corresponding to the primary key (when specified) are moved to the first columns.

6. If convDates is a vector (i.e., c('beginDate', 'endDate')), then a date conversion is attempted for the corresponding columns found in the input file. If TRUE, then a date conversion is attempted for all columns found in the input file with 'date' in the name, If FALSE, no date conversion is attempted.

Some other common time zones include the following: America/New_York, America/Chicago, America/Denver, America/Los_Angeles, America/Anchorage, America/Honolulu, America/Jamaica, America/Managua, America/Phoenix, America/Metlakatla

A brief table reporting the results of the import are printed.

Note that columns containing just F, T, FALSE, TRUE are stored as logical fields

Value

Returns data frame

`loadExcel` *Load/Clean Excel sheet*

Description

Load and clean one sheet from an Excel file

Usage

```
loadExcel(folder = NA, file = NA, sheet = 1, pk = NA,
  remDup = TRUE, remNAcol = TRUE, remNArow = TRUE,
  convDates = TRUE, tzSel = "America/New_York")
```

Arguments

`folder`	folder (i.e., directory to look in, can use relative path)
`file`	file (can use wildcards, e.g., "*.csv")
`sheet`	worksheet name to load from Excel file
`pk`	vector of columns that form the primary key for data set
`remDup`	logical field indicating whether duplicate rows are deleted
`remNAcol`	logical field indicating whether columns with all NA are deleted
`remNArow`	logical field indicating whether rows with all NA are deleted
`convDates`	vector or logical field indicating whether date-like columns should be converted to POSIXct format (see details)
`tzSel`	time zone to use for date conversions (default: "America/New_York")

Details

This function reads in a single sheet from an Excel file using readxl::read_excel to load the data

After reading data in with readxl::read_excel, some specific additional steps are implemented:

1. Double quotes are removed from beginning and ending of all fields. The purpose is to maintaion leading zeroes (e.g., USGS flow gages). To effectively use this functionality, data maintained in a spreadsheet would be enclosed in quotes (e.g., "01578310"). If exported to csv or txt files the field would be in triple quotes (e.g., """01578310"""). Any column read in as integer is converted to numeric.

2. Rows and columns with no data (i.e., all NA) are deleted unless default settings for remNAcol and remNArow are changed to FALSE.

3. Completely duplicate rows are deleted unless default setting for remDup is changed to FALSE.

4. If a primary key (either single or multiple columns) is selected, the function enforces the primary key by deleting duplicate entries based on the primary key. Columns corresponding to the primary key (when specified) are moved to the first columns.

5. If convDates is a vector (i.e., c('beginDate', 'endDate')), then a date conversion is attempted for the corresponding columns found in the input file. If TRUE, then a date conversion is attempted for all columns found in the input file with 'date' in the name, If FALSE, no date conversion is attempted.

Some other common time zones include the following: America/New_York, America/Chicago, America/Denver, America/Los_Angeles, America/Anchorage, America/Honolulu, America/Jamaica, America/Managua, America/Phoenix, America/Metlakatla

A brief table reporting the results of the import are printed.

Note that columns containing just F, T, FALSE, TRUE are stored as logical fields

Value

Returns data frame

`loadModels` *Load Built-in GAM formulas*

Description

Returns built-in GAM formulas

Usage

```
loadModels(gamSelect = "gam4")
```

Arguments

`gamSelect`	character vector of models (Current options include gam0, gam1, gam2, gam3, gam4, gam5)

Value

Returns a list with GAM formulas

---

loadModelsResid *Load Built-in GAM formulas for calculating residuals*

Description

Load Built-in GAM formulas for calculating residuals

Usage

```
loadModelsResid(gamSelect = "doy_flw_sal")
```

Arguments

gamSelect	character vector of models (Current options include 'doy', 'doy_flw_sal', 'doy_flw_sal_int')

Value

Returns a list with GAM formulas

---

parameterList *Parameter List*

Description

A lookup table of water quality parameters

Usage

```
parameterList
```

Format

A data frame with 81 rows and 13 variables:

parm "preferred" parameter ID

parmSource parmamter field as may be found in sample data sets

parmCat Parameter Group Header

parmName Full parameter name

parmCalc indicator if parameter is calculated

parmNamelc parameter name in lower case

parmUnits units

parmRecensor numerical value used in log plots if data are reported as <= 0

parmRO1 Parameter sort order level 1

parmRO2 Parameter sort order level 2

parmTrend TRUE/FALSE for whether parameter should be analyzed for trend

logTrans TRUE/FALSE for whether parameter should be analyzed in log transposed

trendIncrease Should an increase in conccentration be interpreted as 'improving', 'degrading', 'increasing', or 'decreasing'

`qw.export` *qw.export*

Description

Export the contents of a QW object to comma delimited file (csv).

Usage

```
qw.export(df.qw, dir.output = getwd(), fn.output = NULL)
```

Arguments

`df.qw`	Data frame with qw fields to be exported.
`dir.output`	Directory where output files will be saved. Default is working directory.
`fn.output`	Name of output file.

Details

This internal function will export each qw field of a given data frame to separate columns in a single comma separated file (csv). The left header information will be used "as is" but the data fields will be named with the qw column name and the qw slot names .Data.values, .Data.value2, and remark.codes (e.g., secchi..Data.values). These slots names will be renamed _lo, _hi, _symbol (e.g., secchi_lo). The remaning qw slot values will not be exported.

The exported file will be in a directory specified by the user (defaults to working directory). The file will be named as specified by the user.

Value

Returns a user named tab delimited files to the user defined directory with all of the qw slot data.

Examples

```
# define data frame with qw column classes
myDF <- dataCensored

# directory to save output
dir.save <- getwd()

# prefix for file names ()
fn.out <- "dataCensored_TEST.csv"

# run function
qw.export(myDF, dir.save, fn.out)
```

---

qw.import *qw.import*

Description

Import the contents of tab delimited text files to a QW object.

Usage

```
qw.import(fn.import, qw.names, rounding = c(3, 4))
```

Arguments

fn.import	Filename (with path) containing data. Comma separated (CSV) format only.
qw.names	Names of parameters for QW columns.
rounding	OPTIONAL. Rounding values for QW columns. Default is c(3,4) but can be changed.

Details

This internal function will import data from a comma delimited (CSV) file to create a data frame with columns of the qw class.

Only three slots are recognized by this function; ".Data" and "remark.codes" The slot name ".Data" includes ".Data.values" and ".Data.value2".

These three slots are marked as "_lo", "_hi", and "_symbol" in the import file the prefix of the parameter name to be used in the qw object; e.g., secchi_lo, secchi_hi, and secchi_symbol.

To view all qw slot names use the command slotNames("qw").

The function returns a data frame with QW enabled columns. The user must save the file. Any modification of column classes needs to be handled by the user (e.g., character to date or POSIXct).

Value

Returns a QW enabled data frame.

Examples

```
# Use internal function to export dataCensored as example for import
qw.export(dataCensored, getwd(), "dataCensored_TEST.csv")


# Import Test file as a qw object

# Define function parameters
fn.import <- file.path(".", "dataCensored_TEST.csv")
qw.names <- c("secchi", "chla", "do", "tn", "tp", "po4", "tdp"
              , "no23", "nh4", "tdn", "tss")
rounding <- c(3, 4)

# Import
dataCensored.test<- qw.import(fn.import, qw.names, rounding)

# Check for qw class
str(dataCensored.test)

# Save
save(dataCensored.test, file="dataCensored.test.rda")


# Show slot names for a qw object.
slotNames("qw")

####
# convert date field to POSIXct
#dataCensored.test[,"date"] <- as.POSIXct(dataCensored.test[,"date"])
# str(dataCensored.test)
# as.numeric() and as.integer() can be used to convert columns of those types.
```

---

rbindQW *Combine Data by Rows*

Description

Combines a sequence of data frame arguments and combine by rows. This is a specialized version of rbind that works for data frames that contain columns of class "qw."

Usage

```
rbindQW(...)
```

Arguments

...	any number of data frames with identical columns. The missing value NA is permitted as a special case to allow the addition of missing values.

Value

A data frame with all columns combined in the order specified in ....

See Also

rbind

---

sal *Salinity data*

Description

Salinity data, 1984 to 2016, for 8 stations.

Usage

```
sal
```

Format

A data frame with 51,092 rows and 4 variables:

station CBP Station ID

date date, YYYY-MM-DD

layer sample layer

salinity Measured salinity

---

saveDF *Save R object to disk*

Description

Saves R object to disk using csv and/or Rdata format.

Usage

```
saveDF(rObj, note = NULL, rData = FALSE, csv = TRUE, attr = FALSE,
  timeStamp = TRUE, folder = "_saveDF")
```

Arguments

rObj	Name of R object to save.
note	Suffix to include in file name.
rData	Logical field to save rObj as an rData file (FALSE [default]).
csv	Logical field to save rObj as an a csv file (TRUE [default]).
attr	Logical field to save data frame attributes as a text file (FALSE [default]).
timeStamp	Logical field to include date/time stamp in file name (TRUE [default]).
folder	Subdirectory for saving file ('_saveDF' is default)

Details

Output files are saved with an "rObj_note_YYYY_MM_DD_HHMMSS" naming convetion. By default, files are saved as csv files to a '_saveDF' subdirectory relative to the working directory and include a time stamp in the file name using utils::write.csv. The default folder can be changed with the folder argument. Inclusion of a time stamp in the file name enables saving the same object at multiple steps through an R script, but can be turned off with the timeStamp argument. To also save object as rData file, set rData=TRUE.

Value

n/a

Examples

```
df <- data.frame(x=c(1:100))
saveDF(df,'test_note')
```

---

selectData *Select data for analysis from a larger data frame*

Description

Select data for analysis from a larger data frame based on dependent variable, station, and layer. Removing records with missing values, performing log-transformations, and adding a centering date are performed based on settings.

Usage

```
selectData(df, dep, stat, layer = NA, transform = TRUE,
  remMiss = TRUE, analySpec)
```

Arguments

df	data frame
dep	dependent variable
stat	station
layer	layer (optional)
transform	logical field to return log-transformed value (TRUE [default])
remMiss	logical field to remove records where dependent variable, dep, is a missing value (TRUE [default])
analySpec	analytical specifications

Details

The returned data frame will include dyear and cyear. dyear is the decimal year computed using smwrBase::baseDay2decimal and smwrBase::baseDay. From this, the minimum and maximum 'dyear' are averaged. This averaged value, centerYear, is used to compute the centering date, cyear, using cyear = dyear - centerYear.

The variable identified by dep is copied to the variable name dep+".orig" (e.g., chla.orig) allowing the user to track the original concentrations. A new column, recensor, is added. The value of recensor is FALSE unless the value of dep.orig was <=0. In the cases where dep.orig is <= 0, recensor is set to TRUE and the value of dep is set to "less-than" a small positive value which is stored as iSpec$recensor. If transform=TRUE, the returned data frame will also include a variable "ln"+dep (i.e., "lnchla" for log transformed chla).

The data frame will include a column, intervention, which is a factor identifying different periods of record such as when different laboratory methods were used and is based on the data frame methodsList that is loaded into the global environment. This column is set to "A" with only 1 level if the data frame methodsList has not been loaded into the global environment.

The data frame will include a column, lowCensor, to indicate whether the data record occurs in a year with a low level of censoring over that particular year. The function gamTest uses this column to identify years of record (i.e., when lowCensor==FALSE) that should not be used in analyses.

If remMiss=TRUE, then the returned data frame will be down selected by removing records where the variable identified in 'dep' is missing; otherwise, no down selection is performed.

iSpec contains a large list of information

dep - name of column where dependent variable is stored, could be "ln"+dep for variables that will be analyzed after natural log transformation

depOrig - name of original dependent variable, could be same as dep if no transformation is used

stat - name of station

stationMethodGroup - name of station group that the station belongs to, derived from station list (stationMasterList) and used to identify interventions specified in methodsList table

intervenNum - number of interventions found for this station and dependent variable as derived from methodsList table, a value of 1 is assigned if no methodsList entry is found

intervenList - data frame of interventions identified by beginning and ending date and labeled consecutively starting with "A"

layer - layer

layerName - layer name derived from layerLukup

transform - TRUE/FALSE indicating whether log transformations were taken

trendIncrease - an indicator for interpretation of an increasing concentration

logConst - not currently used

recensor - small value that observations <=0 are recensored to as "less than" the small value

censorFrac - data frame indicating the yearly number of observations and fraction of observations reported as less than, uncensored, interval censored, less than zero, and recensored; also includes a 'lowCensor' field indicating which years will be dropped by gamTest due to high yearly censoring

yearRangeDropped - year range of data that will be dropped due to censoring

censorFracSum - censoring overall summary

centerYear - centering year

parmName - parameter name

parmNamelc - parameter name in lower case

parmUnits - parameter units

statLayer - station/layer label, e.g., "LE3.1 (S)"

usgsGageID - USGS gage used for flow adjustments

usgsGageName - USGS gage used for flow adjustments

numObservations - number of observations

dyearBegin - begin date in decimal form

dyearEnd - end date in decimal form

dyearLength - period of record length

yearBegin - period of record begin year

yearend - period of record end year

dateBegin - begin date

dateEnd - end date

The baseDay and baseDay2decimal functions have been added to this package from the smwrBase package.

Value

A nest list is returned. The first element of the nest list is the down-selected data frame. The second element is the list, iSpec, contains specifications for data extraction. See examples for usage and details for further discussion of the data processing and components of each element.

`stationMasterList` *Chesapeake Bay Program long-term tidal monitoring stations*

Description

Chesapeake Bay Program long-term tidal monitoring stations

Usage

```
stationMasterList
```

Format

A data frame with 145 rows and 19 variables:

station Water quality station code

state State location

locationType As identified in the state trend reports, whether this station is in the mainstem or tributary monitoring

waterbody Location as identified in the CBP CIMS database

latitude From the CBP CIMS database, and verified in a GIS map

longitude From the CBP CIMS database, and verified in a GIS map

cbSeg92 Match to CB 92 Segmentation scheme (for 303d list)

usgsGageName Manual match to a USGS fall-line monitoring station (See usgsGages$siteName)

usgsGageID USGS station code (See usgsGages$siteNumber)

usgsGageMatch Identifies how the USGS-to-TribStation match was made. Direct: If the station falls within a tributary with a USGS station, or in the mainstem. Indirect: If the station is in a small sub-tributary of one of the major tributaries. The USGS station may not be that representative, but it is better than matching to the Susquehanna. SusDefault: If there was no clear match, the tidal station was matched to the Susquehanna River.

stationRO1 Station sort order level 1

stationRO2 Station sort order level 2

stationGrpName Station group header

stationMethodGroup Foreign key to methodsList table

hydroTerm "flow" or "salinity" to indicate whether to model flow or salinity effects (if missing, "flow" assumed)

flwAvgWin list of averaging windows if flow is selected in hydroTerm (options are: 1, 5, 10, 15, 20, 30, 40, 50, 60, 90, 120, 150, 180, 210)

flwParms list of parameters to model with flow (regardless of hydroTerm value), each parameter separated by a space

salParms list of parameters to model with salinity (regardless of hydroTerm value), each parameter separated by a space

notes Optional note to track updates

usgsGages *USGS Gages*

Description

List of core USGS gages for CBP trend analyses

Usage

```
usgsGages
```

Format

A data frame with 9 rows and 2 variables:

usgsGageID USGS Gage ID

siteName USGS Site Name

Index

∗Topic **datasets**
dataCensored, 8
layerLukup, 18
parameterList, 22
sal, 26
stationMasterList, 30
usgsGages, 31
∗Topic **data**
rbindQW, 25

analysisOrganizeData, 3

baytrends, 6
baytrends-package *(*baytrends*)*, 6

closeOut, 7
createResiduals, 7

dataCensored, 8
detrended.flow, 9
detrended.salinity, 11

gamDiff, 13
gamPlotDisp, 15
gamTest, 16

impute, 17

layerLukup, 18
loadData, 19
loadExcel, 20
loadModels, 21
loadModelsResid, 22

parameterList, 22

qw.export, 23
qw.import, 24

rbind, *26*
rbindQW, 25

sal, 26
saveDF, 26
selectData, 27
stationMasterList, 30

usgsGages, 31