Package ‘BioMonTools’

October 9, 2025

Type Package

Title Biomonitoring and Bioassessment Calculations

Version 1.2.4

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

Description An aid for manipulating data associated with biomonitoring and bioassessment. Calculations include metric calculation, marking of excluded taxa,
subsampling, and multimetric index calculation. Targeted communities are benthic macroinvertebrates, fish, periphyton, and coral. As described in the Revised Rapid Bioassessment Protocols (Barbour et al. 1999) <`https://archive.epa.gov/water/archive/web/html/index-14.html`>.

License MIT + file LICENSE

Encoding UTF-8

LazyData true

Depends R (>= 3.5)

Imports dplyr, maps, rlang, stats, tidyselect, tidyr

Suggests DataExplorer, DT, ggplot2, knitr, lazyeval, readxl, reshape2, rmarkdown, testthat, shiny, shinydashboard, shinydashboardPlus, shinyjs, shinyWidgets, utils, writexl, shinyalert

URL `https://github.com/leppott/BioMonTools`

BugReports `https://github.com/leppott/BioMonTools/issues`

VignetteBuilder knitr

RoxygenNote 7.3.3

NeedsCompilation no

Author Erik W. Leppo [aut, cre] (ORCID:
<`https://orcid.org/0000-0001-8026-7474`>),
Jen Stamp [ctb],
John van Sickles [ctb],
Ben Block [ctb]

Repository CRAN

Date/Publication 2025-10-09 12:40:02 UTC

Contents

`assign_IndexClass` *Assign Index_Class*

Description

Assign Index_Class for based on user input fields. If use the same name of an existing field the information will be overwritten.

Multiple criteria are treated as "AND" so all must be met to be assigned to a particular Index_Class.

Internally uses ‘tidyr‘ and ‘dplyr‘

If Index_Class is included in data then it is renamed Index_Class_Orig and returned in the output data frame.

Usage

```
assign_IndexClass(
  data,
  criteria,
  name_indexclass = "INDEX_CLASS",
  name_indexname = "INDEX_NAME",
  name_siteid = "SITEID",
  data_shape = "WIDE"
)
```

Arguments

`data`	Data frame (wide format) with metric values to be evaluated.
`criteria`	Data frame of metric thresholds to check.
`name_indexclass`	Name for new Index_Class column. Default = INDEX_CLASS
`name_indexname`	Name for Index Name column. Default = INDEX_NAME
`name_siteid`	Name for Site ID column. Default = SITEID
`data_shape`	Shape of data; wide or long. Default is 'wide'

Details

Requires use of reference file with criteria.

Value

Returns a data frame with new column added.

Examples

```
# Packages
library(readxl)

# EXAMPLE 1
# Create Example Data
df_data <- data.frame(SITEID = paste0("Site_", LETTERS[1:10]),
                      INDEX_NAME = "BCG_MariNW_Bugs500ct",
                      GRADIENT = round(stats::runif(10, 0.5, 1.5), 1),
                      ELEVATION = round(stats::runif(10, 700, 800), 1))

# Import Checks
df_criteria <- read_excel(system.file("extdata/IndexClass.xlsx",
                                      package = "BioMonTools"),
                          sheet = "Index_Class")

# Run Function
df_results <- assign_IndexClass(df_data, df_criteria, "INDEX_CLASS")

# Results
```

```
df_results
```

`data_benthos_MBSS` *Benthic macroinvertebrate taxa data; MBSS*

Description

A data set with example benthic macroinvertebrate data. Calculate metrics then statistics. Data from MBSS.

Usage

```
data_benthos_MBSS
```

Format

A data frame with 5,666 observations on the following 40 variables.

`INDEX_NAME` a character vector

`SAMPLEID` a character vector

`DATE` a character vector

`TAXAID` a character vector

`N_TAXA` a numeric vector, count

`N_GRIDS` a numeric vector, number of grids in subsample (max = 30)

`EXCLUDE` a character vector, whether taxon should be excluded from taxa richness metrics

`INDEX_CLASS` a character vector, index region

`Phylum` a character vector

`Class` a character vector

`Order` a character vector

`Family` a character vector

`Genus` a character vector

`Other_Taxa` a character vector

`Tribe` a character vector

`FFG` a character vector

`FAM_TV` a numeric vector

`Habit` a character vector

`TOLVAL` a numeric vector

`TOLVAL2` a numeric vector

`UFC` a numeric vector

`UFC_Comment` a character vector

`SUBPHYLUM` a character vector

`SUBCLASS` a character vector

`INFRAORDER` a character vector

`SUBFAMILY` a character vector

`LIFE_CYCLE` a character vector

`BCG_ATTR` a character vector

`THERMAL_INDICATOR` a character vector

`LONGLIVED` a character vector

`NOTEWORTHY` a character vector

`FFG2` a character vector

`HABITAT` a character vector

`ELEVATION_ATTR` a character vector

`GRADIENT_ATTR` a character vector

`WSAREA_ATTR` a character vector

`HABSTRUCT` a character vector

`BCG_ATTR2` a character vector

`NONTARGET` a logical vector

`AIRBREATHER` a logical vector

Source

example data from MBSS

`data_benthos_PacNW` *Benthic macroinvertebrate taxa data; Pacific Northwest*

Description

A dataset with example (demonstration only) taxa data and attributes for calculating metric values.

This dataset is an example only. DO NOT USE for any analyses.

Usage

```
data_benthos_PacNW
```

Format

A data frame with 598 observations on the following 38 variables.

`INDEX_NAME` a character vector

`INDEX_CLASS` a character vector

`SampleID` a character vector

`TaxaID` a character vector

`N_TAXA` a numeric vector

`Exclude` a logical vector

`NonTarget` a logical vector

`Phylum` a character vector

`Class` a character vector

`Order` a character vector

`Family` a character vector

`Subfamily` a character vector

`Tribe` a character vector

`Genus` a character vector

`BCG_Attr` a numeric vector

`Thermal_Indicator` a character vector

`FFG` a character vector

`Clinger` a character vector

`LongLived` a logical vector

`Noteworthy` a logical vector

`Habitat` a character vector

`SubPhylum` a character vector

`InfraOrder` a character vector

`Habit` a logical vector

`Life_Cycle` a logical vector

`TolVal` a logical vector

`FFG2` a logical vector

`TolVal2` a logical vector

`UFC` a character vector

`UFC_Comment` a numeric vector

`SubClass` a character vector

`Elevation_Attr` a character vector

`Gradient_Attr` a character vector

`WSArea_Attr` a character vector

`HabStruct` a character vector

`BCG_Attr2` a character vector

`AirBreather` a logical vector

Source

example data

`data_bio2rarify` *rarify example data*

Description

A dataset with example benthic macroinvertebrate data (600 count) to be used with the rarify function. Includes 12 samples.

Usage

```
data_bio2rarify
```

Format

A data frame with 223 rows and 28 variables:

SampleID Sample ID

TaxaID unique taxonomic identifier

N_Taxa number of individuals in sample

Source

example data

`data_coral_bcg_metric_dev` *Coral taxa data; Florida BCG.*

Description

A data set with example coral data. Calculate metrics. Data from Florida BCG providers.

Usage

```
data_coral_bcg_metric_dev
```

Format

A data frame with 2138 observations on the following 25 variables.

`DataSource` a character vector

`SampleID` a character vector

`TotTranLngth_m` a numeric vector

`SampDate` a Date

`TAXAID` a character vector

`CommonName` a character vector

`Juvenile` a logical vector

`DiamMax_cm` a numeric vector

`DiamPerp_cm` a numeric vector

`Height_cm` a numeric vector

`TotMort_pct` a numeric vector

`BCG_ATTR` a character vector

`Weedy` a character vector

`LRBC` a logical vector

`MorphConvFact` a numeric vector

`Phylum` a character vector

`Class` a character vector

`SubClass` a character vector

`Order` a character vector

`Family` a character vector

`Genus` a character vector

`SubGenus` a character vector

`Species` a character vector

`INDEX_NAME` a character vector

`INDEX_CLASS` a character vector

Source

example coral data from Florida BCG

`data_coral_bcg_metric_qc` *Coral metric value data; Florida BCG.*

Description

A data set with coral metric value data. Used to compare to metric value calculations. Data from Florida BCG providers.

Usage

```
data_coral_bcg_metric_qc
```

Format

A data frame with 100 observations on the following 19 variables.

`SAMPLEID` a character vector

`INDEX_NAME` a character vector

`INDEX_CLASS` a character vector

`transect_area_m2` a numeric vector

`ncol_total` a numeric vector

`lcol_total` a numeric vector

`nt_total` a numeric vector

`ncol_Acropora` a numeric vector

`ncol_AcroOrbi_m2` a numeric vector

`pcol_Acropora` a numeric vector

`nt_BCG_att123` a numeric vector

`nt_BCG_att1234` a numeric vector

`nt_BCG_att5` a numeric vector

`pt_BCG_att5` a numeric vector

`LCSA3D_samp_m2` a numeric vector

`LCSA3D_BCG_att1234_m2` a numeric vector

`LCSA3D_LRBC_m2` a numeric vector

`ncol_SmallWeedy` a numeric vector

`pcol_SmallWeedy` a numeric vector

Source

example coral metric results from Florida BCG

`data_diatom_mmi_dev` *Diatom taxa data; Indiana DEM*

Description

A data set with example diatom data. Calculate metrics. Data from IDEM.

Usage

```
data_diatom_mmi_dev
```

Format

A data frame with 24797 observations on the following 38 variables.

`INDEX_NAME` a character vector

`INDEX_CLASS` a character vector

`STATIONID` a character vector

`COLLDATE` a Date

`SAMPLEID` a character vector

`TAXAID` a character vector

`EXCLUDE` a logical vector

`NONTARGET` a logical vector

`N_TAXA` a numeric vector

`ORDER` a character vector

`FAMILY` a character vector

`GENUS` a character vector

`BC_USGS` a character vector

`TROPHIC_USGS` a character vector

`SAP_USGS` a character vector

`PT_USGS` a character vector

`O_USGS` a character vector

`SALINITY_USGS` a character vector

`BAHLS_USGS` a character vector

`P_USGS` a character vector

`N_USGS` a character vector

`HABITAT_USGS` a character vector

`N_FIXER_USGS` a character vector

`MOTILITY_USGS` a character vector

`SIZE_USGS` a character vector

`HABIT_USGS` a character vector

`MOTILE2_USGS` a character vector

`TOLVAL` a numeric vector

`DIATOM_ISA` a character vector

`DIAT_CL` a numeric vector

`POLL_TOL` a numeric vector

`BEN_SES` a numeric vector

`DIATAS_TP` a numeric vector

`DIATAS_TN` a numeric vector

`DIAT_COND` a numeric vector

`DIAT_CA` a numeric vector

`MOTILITY` a numeric vector

`NF` a numeric vector#'

`PHYLUM` a character vector

Source

example data from IDEM

`data_diatom_mmi_qc` *Diatom metric value data; Indiana DEM*

Description

A data set with diatom metric value data. Used to compare to metric value calculations. Data from IDEM.

Usage

```
data_diatom_mmi_qc
```

Format

A data frame with 497 observations on the following 250 variables.

`SAMPLEID` a character vector

`INDEX_NAME` a character vector

`INDEX_CLASS` a character vector

`ni_total` a numeric vector

`li_total` a numeric vector

`nt_total` a numeric vector

`nt_Achnan_Navic` a numeric vector

`nt_LOW_N` a numeric vector

`nt_HIGH_N` a numeric vector

`nt_LOW_P` a numeric vector

`nt_HIGH_P` a numeric vector

`nt_BC_1` a numeric vector

`nt_BC_2` a numeric vector

`nt_BC_3` a numeric vector

`nt_BC_4` a numeric vector

`nt_BC_5` a numeric vector

`nt_BC_12` a numeric vector

`nt_BC_45` a numeric vector

`nt_PT_1` a numeric vector

`nt_PT_2` a numeric vector

`nt_PT_3` a numeric vector

`nt_PT_4` a numeric vector

`nt_PT_5` a numeric vector

`nt_PT_12` a numeric vector

`nt_SALINITY_1` a numeric vector

`nt_SALINITY_2` a numeric vector

`nt_SALINITY_3` a numeric vector

`nt_SALINITY_4` a numeric vector

`nt_SALINITY_12` a numeric vector

`nt_SALINITY_34` a numeric vector

`nt_O_1` a numeric vector

`nt_O_2` a numeric vector

`nt_O_3` a numeric vector

`nt_O_4` a numeric vector

`nt_O_5` a numeric vector

`nt_O_345` a numeric vector

`nt_SESTONIC_HABIT` a numeric vector

`nt_BENTHIC_HABIT` a numeric vector

`nt_BAHLS_1` a numeric vector

`nt_BAHLS_2` a numeric vector

`nt_BAHLS_3` a numeric vector

`nt_TROPHIC_1` a numeric vector

`nt_TROPHIC_2` a numeric vector

`nt_TROPHIC_3` a numeric vector

`nt_TROPHIC_4` a numeric vector

`nt_TROPHIC_5` a numeric vector

`nt_TROPHIC_6` a numeric vector

`nt_TROPHIC_7` a numeric vector

`nt_TROPHIC_456` a numeric vector

`nt_SAP_1` a numeric vector

`nt_SAP_2` a numeric vector

`nt_SAP_3` a numeric vector

`nt_SAP_4` a numeric vector

`nt_SAP_5` a numeric vector

`nt_NON_N_FIXER` a numeric vector

`nt_N_FIXER` a numeric vector

`nt_HIGHLY_MOTILE` a numeric vector

`nt_MODERATELY_MOTILE` a numeric vector

`nt_NON_MOTILE` a numeric vector

`nt_SLIGHTLY_MOTILE` a numeric vector

`nt_WEAKLY_MOTILE` a numeric vector

`nt_BIG` a numeric vector

`nt_SMALL` a numeric vector

`nt_MEDIUM` a numeric vector

`nt_VERY_BIG` a numeric vector

`nt_VERY_SMALL` a numeric vector

`nt_ADNATE` a numeric vector

`nt_STALKED` a numeric vector

`nt_HIGHLY_MOTILE.1` a numeric vector

`nt_ARAPHID` a numeric vector

`nt_DIAT_CL_1` a numeric vector

`nt_DIAT_CL_2` a numeric vector

`nt_BEN_SES_1` a numeric vector

`nt_BEN_SES_2` a numeric vector

`nt_DIAT_CA_1` a numeric vector

`nt_DIAT_CA_2` a numeric vector

`nt_DIAT_COND_1` a numeric vector

`nt_DIAT_COND_2` a numeric vector

`nt_DIATAS_TN_1` a numeric vector

`nt_DIATAS_TN_2` a numeric vector

`nt_DIATAS_TP_1` a numeric vector

`nt_DIATAS_TP_2` a numeric vector

`nt_MOTILITY_1` a numeric vector

`nt_MOTILITY_2` a numeric vector

`nt_NF_1` a numeric vector

`nt_NF_2` a numeric vector

`pi_Achnan_Navic` a numeric vector

`pi_HIGH_N` a numeric vector

`pi_LOW_N` a numeric vector

`pi_HIGH_P` a numeric vector

`pi_LOW_P` a numeric vector

`pi_BC_1` a numeric vector

`pi_BC_2` a numeric vector

`pi_BC_3` a numeric vector

`pi_BC_4` a numeric vector

`pi_BC_5` a numeric vector

`pi_PT_1` a numeric vector

`pi_PT_2` a numeric vector

`pi_PT_3` a numeric vector

`pi_PT_4` a numeric vector

`pi_PT_5` a numeric vector

`pi_PT_45` a numeric vector

`pi_SALINITY_1` a numeric vector

`pi_SALINITY_2` a numeric vector

`pi_SALINITY_3` a numeric vector

`pi_SALINITY_4` a numeric vector

`pi_O_1` a numeric vector

`pi_O_2` a numeric vector

`pi_O_3` a numeric vector

`pi_O_4` a numeric vector

`pi_O_5` a numeric vector

`pi_SESTONIC_HABIT` a numeric vector

`pi_BENTHIC_HABIT` a numeric vector

`pi_BAHLS_1` a numeric vector

`pi_BAHLS_2` a numeric vector

`pi_BAHLS_3` a numeric vector

`pi_TROPHIC_1` a numeric vector

`pi_TROPHIC_2` a numeric vector

`pi_TROPHIC_3` a numeric vector

`pi_TROPHIC_4` a numeric vector

`pi_TROPHIC_5` a numeric vector

`pi_TROPHIC_6` a numeric vector

`pi_TROPHIC_7` a numeric vector

`pi_SAP_1` a numeric vector

`pi_SAP_2` a numeric vector

`pi_SAP_3` a numeric vector

`pi_SAP_4` a numeric vector

`pi_SAP_5` a numeric vector

`pi_NON_N_FIXER` a numeric vector

`pi_N_FIXER` a numeric vector

`pi_HIGHLY_MOTILE` a numeric vector

`pi_MODERATELY_MOTILE` a numeric vector

`pi_NON_MOTILE` a numeric vector

`pi_SLIGHTLY_MOTILE` a numeric vector

`pi_WEAKLY_MOTILE` a numeric vector

`pi_BIG` a numeric vector

`pi_SMALL` a numeric vector

`pi_MEDIUM` a numeric vector

`pi_VERY_BIG` a numeric vector

`pi_VERY_SMALL` a numeric vector

`pi_ADNATE` a numeric vector

`pi_STALKED` a numeric vector

`pi_HIGHLY_MOTILE.1` a numeric vector

`pi_ARAPHID` a numeric vector

`pi_DIAT_CL_1` a numeric vector

`pi_DIAT_CL_1_ASSR` a numeric vector

`pi_DIAT_CL_2` a numeric vector

`pi_BEN_SES_1` a numeric vector

`pi_BEN_SES_2` a numeric vector

`pi_DIAT_CA_1` a numeric vector

`pi_DIAT_CA_2` a numeric vector

`pi_DIAT_COND_1` a numeric vector

`pi_DIAT_COND_2` a numeric vector

`pi_DIATAS_TN_1` a numeric vector

`pi_DIATAS_TN_2` a numeric vector

`pi_DIATAS_TP_1` a numeric vector

`pi_DIATAS_TP_2` a numeric vector

`pi_MOTILITY_1` a numeric vector

`pi_MOTILITY_2` a numeric vector

`pi_NF_1` a numeric vector

`pi_NF_2` a numeric vector

`pt_Achnan_Navic` a numeric vector

`pt_HIGH_N` a numeric vector

`pt_LOW_N` a numeric vector

`pt_HIGH_P` a numeric vector

`pt_LOW_P` a numeric vector

`pt_BC_1` a numeric vector

`pt_BC_2` a numeric vector

`pt_BC_3` a numeric vector

`pt_BC_4` a numeric vector

`pt_BC_5` a numeric vector

`pt_BC_12` a numeric vector

`pt_BC_45` a numeric vector

`pt_PT_1` a numeric vector

`pt_PT_2` a numeric vector

`pt_PT_3` a numeric vector

`pt_PT_4` a numeric vector

`pt_PT_5` a numeric vector

`pt_PT_12` a numeric vector

`pt_SALINITY_1` a numeric vector

`pt_SALINITY_2` a numeric vector

`pt_SALINITY_3` a numeric vector

`pt_SALINITY_4` a numeric vector

`pt_SALINITY_34` a numeric vector

`pt_O_1` a numeric vector

`pt_O_2` a numeric vector

`pt_O_3` a numeric vector

`pt_O_4` a numeric vector

`pt_O_5` a numeric vector

`pt_O_345` a numeric vector

`pt_SESTONIC_HABIT` a numeric vector

`pt_BENTHIC_HABIT` a numeric vector

`pt_BAHLS_1` a numeric vector

`pt_BAHLS_2` a numeric vector

`pt_BAHLS_3` a numeric vector

`pt_TROPHIC_1` a numeric vector

`pt_TROPHIC_2` a numeric vector

`pt_TROPHIC_3` a numeric vector

`pt_TROPHIC_4` a numeric vector

`pt_TROPHIC_5` a numeric vector

`pt_TROPHIC_6` a numeric vector

`pt_TROPHIC_7` a numeric vector

`pt_TROPHIC_456` a numeric vector

`pt_SAP_1` a numeric vector

`pt_SAP_2` a numeric vector

`pt_SAP_3` a numeric vector

`pt_SAP_4` a numeric vector

`pt_SAP_5` a numeric vector

`pt_NON_N_FIXER` a numeric vector

`pt_N_FIXER` a numeric vector

`pt_HIGHLY_MOTILE` a numeric vector

`pt_MODERATELY_MOTILE` a numeric vector

`pt_NON_MOTILE` a numeric vector

`pt_SLIGHTLY_MOTILE` a numeric vector

`pt_WEAKLY_MOTILE` a numeric vector

`pt_BIG` a numeric vector

`pt_SMALL` a numeric vector

`pt_MEDIUM` a numeric vector

`pt_VERY_BIG` a numeric vector

`pt_VERY_SMALL` a numeric vector

`pt_ADNATE` a numeric vector

`pt_STALKED` a numeric vector

`pt_HIGHLY_MOTILE.1` a numeric vector

`pt_ARAPHID` a numeric vector

`pt_DIAT_CL_1` a numeric vector

`pt_DIAT_CL_2` a numeric vector

`pt_BEN_SES_1` a numeric vector

`pt_BEN_SES_2` a numeric vector

`pt_DIAT_CA_1` a numeric vector

`pt_DIAT_CA_2` a numeric vector

`pt_DIAT_COND_1` a numeric vector

`pt_DIAT_COND_2` a numeric vector

`pt_DIATAS_TN_1` a numeric vector

`pt_DIATAS_TN_2` a numeric vector

`pt_DIATAS_TP_1` a numeric vector

`pt_DIATAS_TP_2` a numeric vector

`pt_MOTILITY_1` a numeric vector

`pt_MOTILITY_2` a numeric vector

`pt_NF_1` a numeric vector

`pt_NF_2` a numeric vector

`nt_Sens_810` a numeric vector

`nt_RefIndicators` a numeric vector

`nt_Tol_13` a numeric vector

`pi_Sens_810` a numeric vector

`pi_RefIndicators` a numeric vector

`pi_Tol_13` a numeric vector

`pt_Sens_810` a numeric vector

`pt_RefIndicators` a numeric vector

`pt_Tol_13` a numeric vector

`wa_POLL_TOL` a numeric vector

Source

example metric value data from IDEM

`data_fish_MBSS` *Fish data, MBSS*

Description

A dataset with example fish taxa data for metric calculation.

Usage

```
data_fish_MBSS
```

Format

A data frame with 1694 observations on the following 30 variables.

`SAMPLEID`	a character vector
`TAXAID`	a character vector
`N_TAXA`	a numeric vector
`TYPE`	a character vector
`TOLER`	a character vector
`NATIVE`	a character vector
`TROPHIC`	a character vector
`SILT`	a character vector
`INDEX_CLASS`	a character vector
`SAMP_LENGTH_M`	a numeric vector
`SAMP_WIDTH_M`	a numeric vector
`SAMP_BIOMASS`	a numeric vector
`INDEX_NAME`	a character vector
`EXCLUDE`	a logical vector
`BCG_ATTR`	a character vector#'
`DA_MI2`	a numeric vector
`N_ANOMALIES`	a numeric vector
`FAMILY`	a character vector
`GENUS`	a character vector
`THERMAL_INDICATOR`	a character vector
`ELEVATION_ATTR`	a character vector
`GRADIENT_ATTR`	a character vector
`WSAREA_ATTR`	a character vector
`REPRODUCTION`	a character vector
`HABITAT`	a character vector
`CONNECTIVITY`	a logical vector
`SCC`	a logical vector
`HYBRID`	a logical vector
`BCGATTR2`	a character vector
`TOLVAL2`	a numeric vector

Source

example data

`data_metval_scmb_ibi` *data_metval_scmb_ibi*

Description

Example data metrics

@format A data frame with 20 observations on the following 13 variables.

`INDEX_NAME`	a character vector
`INDEX_REGION`	a character vector
`SampID`	a character vector
`nt_total`	a numeric vector
`nt_Mol`	a numeric vector
`ni_Noto`	a numeric vector
`pi_intol`	a numeric vector
`qc_nt_total`	a numeric vector
`qc_nt_Mol`	a numeric vector
`qc_ni_Noto`	a numeric vector
`qc_pi_intol`	a numeric vector
`qc_sum`	a numeric vector
`qc_nar`	a character vector

Usage

```
data_metval_scmb_ibi
```

Format

An object of class `data.frame` with 20 rows and 13 columns.

Source

example data

`data_mmi_dev` *Metric data for metric stats for mmi development*

Description

A data set with example benthic macroinvertebrate data. Calculate metrics then statistics.

Usage

```
data_mmi_dev
```

Format

A data frame with 10,574 observations on the following 34 variables.

`Class` a character vector

`Ref_v1` a character vector

`CalVal_Class4` a character vector

`Unique_ID` a character vector

`BenSampID` a character vector

`CollDate` a character vector

`CollMeth` a character vector

`TaxaID` a character vector

`Individuals` a numeric vector

`Exclude` a logical vector

`NonTarget` a character vector

`Phylum` a character vector

`Benthic_MasterTaxa.Class` a character vector

`Order` a character vector

`Family` a character vector

`Subfamily` a character vector

`Tribe` a character vector

`Genus` a character vector

`TolVal` a character vector

`FFG` a character vector

`Habit` a character vector

`INDEX_NAME` a character vector

`SUBPHYLUM` a character vector

`CLASS` a character vector

`SUBCLASS` a character vector

`INFRAORDER` a character vector

`LIFE_CYCLE` a character vector

`BCG_ATTR` a character vector

`THERMAL_INDICATOR` a character vector

`LONGLIVED` a character vector

`NOTEWORTHY` a character vector

`FFG2` a character vector

`TOLVAL2` a character vector

`HABITAT` a numeric vector

Source

example data

`data_mmi_dev_small` *Metric data for metric stats for mmi development*

Description

A data set with example benthic macroinvertebrate data. Calculate metrics then statistics.

Usage

`data_mmi_dev_small`

Format

A data frame with 1,374 observations on the following 34 variables.

`Class` a character vector

`Ref_v1` a character vector

`CalVal_Class4` a character vector

`Unique_ID` a character vector

`BenSampID` a character vector

`CollDate` a character vector

`CollMeth` a character vector

`TaxaID` a character vector

`Individuals` a numeric vector

`Exclude` a logical vector

`NonTarget` a character vector

`Phylum` a character vector

`Benthic_MasterTaxa.Class` a character vector

`Order` a character vector

`Family` a character vector

`Subfamily` a character vector

`Tribe` a character vector

`Genus` a character vector

`TolVal` a character vector

`FFG` a character vector

`Habit` a character vector

`INDEX_NAME` a character vector

`SUBPHYLUM` a character vector

`CLASS` a character vector

`SUBCLASS` a character vector

`INFRAORDER` a character vector

`LIFE_CYCLE` a character vector

`BCG_ATTR` a character vector

`THERMAL_INDICATOR` a character vector

`LONGLIVED` a character vector

`NOTEWORTHY` a character vector

`FFG2` a character vector

`TOLVAL2` a character vector

`HABITAT` a numeric vector

Source

example data

`data_Taxa_MA` *Estuary taxa data*

Description

A dataset with example fish taxa data and locations for mapping.

Usage

```
data_Taxa_MA
```

Format

A data frame with 2,675 observations on the following 15 variables.

`estuary` a factor with levels `BOSTON HARBOR` `BUZZARDS BAY` `CAPE COD BAY` `MASSACHUSETTS BAY` `WAQUOIT BAY`

`CommonName` a factor with levels `ALEWIFE` `AMERICAN EEL` `AMERICAN LOBSTER` `AMERICAN PLAICE` `AMERICAN SAND LANCE` `AMERICAN SHAD` `ATLANTIC COD` `ATLANTIC CROAKER` `ATLANTIC HERRING` `ATLANTIC MACKEREL` `ATLANTIC MENHADEN` `ATLANTIC ROCK CRAB` `ATLANTIC SALMON` `ATLANTIC STINGRAY` `ATLANTIC STURGEON` `ATLANTIC TOMCOD` `BAY ANCHOVY` `BAY SCALLOP` `BLACK DRUM` `BLACK SEA BASS` `BLUE CRAB` `BLUE MUSSEL` `BLUEBACK HERRING` `BLUEFISH` `BROWN SHRIMP` `BUTTERFISH` `CHANNEL CATFISH` `COWNOSE RAY` `CUNNER` `DAGGERBLADE GRASS SHRIMP` `EASTERN OYSTER` `FOURSPINE STICKLEBACK` `GOBIES` `GREEN CRAB` `GREEN SEA URCHIN` `GRUBBY` `HADDOCK` `HOGCHOKER` `JONAH CRAB` `KILLIFISHES` `LONGHORN SCULPIN` `MULLETS` `MUMMICHOG` `NINESPINE STICKLEBACK` `NORTHERN KINGFISH` `NORTHERN PIPEFISH` `NORTHERN SEAROBIN` `NORTHERN SHRIMP` `OCEAN POUT` `OYSTER TOADFISH` `PINFISH` `POLLOCK` `QUAHOG` `RAINBOW SMELT` `RED DRUM` `RED HAKE` `ROCK GUNNEL` `SCUP` `SEA SCALLOP` `SEVENSPINE BAY SHRIMP` `SHEEPSHEAD MINNOW` `SHORTHORN SCULPIN` `SHORTNOSE STURGEON` `SILVER HAKE` `SILVERSIDES` `SKATES` `SMOOTH FLOUNDER` `SOFTSHELL CLAM` `SPINY DOGFISH` `SPOT` `SPOTTED SEATROUT` `STRIPED BASS` `SUMMER FLOUNDER` `TAUTOG` `THREESPINE STICKLEBACK` `WEAKFISH` `WHITE HAKE` `WHITE PERCH` `WINDOWPANE FLOUNDER` `WINTER FLOUNDER` `YELLOW PERCH` `YELLOWTAIL FLOUNDER`

`LifeStage` a factor with levels `ADULTS` `EGGS` `JUVENILES` `LARVAE` `MATING` `PARTURITION` `SPAWNING`

`SalZone` a factor with levels `>25 ppt` `0.5-25 ppt`

`Winter` a numeric vector

`Spring` a numeric vector

`Summer` a numeric vector

`Fall` a numeric vector

`All` a numeric vector

`TaxaName` Taxa Names for mapping

`State` a factor with levels `MA`

`Latitude` a numeric vector

`Longitude` a numeric vector

`Count` a numeric vector

`PctDensity` a numeric vector

Source

example data

`MapTaxaObs` *Taxa Observation Maps*

Description

Map taxonomic observations from a data frame. Input a dataframe with SampID, TaxaID, TaxaCount, Latitude, and Longitude. Other arguments are format (jpg vs. pdf), file name prefix, and output directory. Files are saved with the prefix "map.taxa." by default.

Usage

```
MapTaxaObs(
  df_obs,
  SampID,
  TaxaID,
  TaxaCount,
  Lat,
  Long,
  output_dir = NULL,
  output_prefix = "maps.taxa",
  output_type = "pdf",
  database,
  regions,
  map_grp = NULL,
  leg_loc = "right",
  verbose = FALSE,
  ...
)
```

Arguments

`df_obs`	Observation data frame
`SampID`	df_obs column name, Unique Sample identifier
`TaxaID`	df_obs column name, Unique Taxa identifier
`TaxaCount`	df_obs column name, Number of individuals for TaxonID and SampID
`Lat`	df_obs column name, Latitude
`Long`	df_obs column name, Longitude
`output_dir`	Directory to save output. Default is working directory.
`output_prefix`	Prefix to TaxaID for each file. Default = "map.taxa."
`output_type`	File format for output; jpg or pdf.
`database`	maps::map function database; world, usa, state, county
`regions`	maps::map function regions. Names pertinent to map_db.
`map_grp`	Map grouping variable from df_obs. Will generate legend and color code the points on the map. Default = NULL

`leg_loc`	Legend location text. Default = "right" Other values may not work properly.
`verbose`	Boolean value for if status messages are output to the console. Default = FALSE
`...`	Optional arguments to be passed to methods.

Details

The user will pass arguments for maps::map function that is used for the map. For example, 'database' and 'regions'. Without these arguments no map will be created.

The map will have all points and colored points for each taxon. In addition the map will include the number of samples by taxon.

The example data is fish but can be used for benthic macroinvertebrates as well.

If use grouping variable colors are from grDevices::rainbow()

Jpg file names replace all non-alphanumeric characters with "_".

The R package maps is required for this function.

Value

Taxa maps to user defined directory as jpg or pdf.

Examples

```
df_obs     <- data_Taxa_MA
SampID     <- "estuary"
TaxaID     <- "TaxaName"
TaxaCount  <- "Count"
Lat        <- "Latitude"
Long       <- "Longitude"
output_dir <- tempdir()
output_prefix <- "maps.taxa."
output_type   <- "pdf"

myDB     <- "state"
myRegion <- "massachusetts"
myXlim   <- c(-(73+(30/60)), -(69+(56/60)))
myYlim   <- c((41+(14/60)),(42+(53/60)))

# Run function with extra arguments for map
MapTaxaObs(df_obs[1:500, ],
           SampID,
           TaxaID,
           TaxaCount,
           Lat,
           Long,
           output_dir,
           output_prefix,
           output_type,
           database = "state",
           regions = "massachusetts",
           map_grp = "estuary",
```

```
            leg_loc = "bottomleft",
            xlim = myXlim,
            ylim = myYlim,
            verbose = FALSE)
```

`markExcluded` *Mark "exclude" (non-distinct / non-unique / ambiguous) taxa*

Description

Takes as an input data frame with Sample ID, Taxa ID, and phlogenetic name fields and returns a similar dataframe with a column for "exclude" taxa (TRUE or FALSE).

Exclude taxa are refered to by multiple names; ambiguous, non-distinct, and non-unique. The "exclude" name was chosen so as to be consistent with "non-target" taxa. That is, taxa marked as "TRUE" are treated as undesireables. Exclude taxa are those that are present in a sample when taxa of the same group are present in the same sample are identified finer level. That is, the parent is marked as exclude when child taxa are present in the same sample.

Usage

```
markExcluded(
  df_samptax,
  SampID = "SAMPLEID",
  TaxaID = "TAXAID",
  TaxaCount = "N_TAXA",
  Exclude = "EXCLUDE",
  TaxaLevels,
  Exceptions = NA,
  verbose = FALSE
)
```

Arguments

`df_samptax`	Input data frame.
`SampID`	Column name in df_samptax for sample identifier. Default = "SAMPLEID".
`TaxaID`	Column name in df_samptax for organism identifier. Default = "TAXAID".
`TaxaCount`	Column name in df_samptax for organism count. Default = "N_TAXA".
`Exclude`	Column name for Exclude Taxa results in returned data frame. Default = "Exclude".
`TaxaLevels`	Column names in df_samptax that for phylogenetic names to be evaluated. Need to be in order from coarse to fine (i.e., Phylum to Species).
`Exceptions`	NA or two column data frame of synonyms or other exceptions. Default = NA Column 1 is the name used in the TaxaID column of df_samptax. Column 2 is the name used in the TaxaLevels columns of df_samptax.
`verbose`	Boolean value for if status messages are output to the console. Default = FALSE

Details

The exclude taxa are referenced in the metric values function. These taxa are removed from the taxa richness metrics. This is because these are coarser level taxa when fine level taxa are present in the same sample.

Exceptions is a 2 column data frame of synonyms or other exceptions. Column 1 is the name used in the TaxaID column the input data frame (df_samptax). Column 2 is the name used in the TaxaLevels columns of the input data frame (df_samptax). The phylogenetic columns (TaxaLevels) will be modified from Column 2 of the Exceptions data frame to match Column 1 of the Exceptions data frame. This ensures that the algorithm for markExcluded works properly. The changes will not be stored and the original names provided in the input data frame (df_samptax) will be returned in the final result. The function example below includes a practical case.

Taxa Levels are phylogenetic names that are to be checked. They should be listed in order from course (kingdom) to fine (species). Names not appearing in the data will be skipped.

The spelling of names must be consistent (including case) for this function to produce the intended output.

Value

Returns a data frame of df_samptax with an additional column, Exclude.

Examples

```
# Packages
library(readxl)
library(dplyr)
library(lazyeval)
library(knitr)

# Data
df_samps_bugs <- read_excel(system.file("./extdata/Data_Benthos.xlsx",
                                        package="BioMonTools"),
                            guess_max=10^6)

# Variables
SampID     <- "SampleID"
TaxaID     <- "TaxaID"
TaxaCount  <- "N_Taxa"
Exclude    <- "Exclude_New"
TaxaLevels <- c("Kingdom",
                "Phylum",
                "SubPhylum",
                "Class",
                "SubClass",
                "Order",
                "SubOrder",
                "SuperFamily",
                "Family",
                "SubFamily",
                "Tribe",
                "Genus",
```

```
                "SubGenus",
                "Species",
                "Variety")
# Taxa that should be treated as equivalent
Exceptions <- data.frame("TaxaID" = "Sphaeriidae",
                         "PhyloID" = "Pisidiidae")

# EXAMPLE 1
df_tst <- markExcluded(df_samps_bugs,
                       SampID = "SampleID",
                       TaxaID = "TaxaID",
                       TaxaCount = "N_Taxa",
                       Exclude = "Exclude_New",
                       TaxaLevels = TaxaLevels,
                       Exceptions = Exceptions)

# Compare
df_compare <- dplyr::summarise(dplyr::group_by(df_tst, SampleID),
                               Exclude_Import = sum(Exclude),
                               Exclude_R = sum(Exclude_New))
df_compare$Diff <- df_compare$Exclude_Import - df_compare$Exclude_R
#
tbl_diff <- table(df_compare$Diff)
kable(tbl_diff)
# sort
df_compare <- df_compare %>% arrange(desc(Diff))

# Number with issues
sum(abs(df_compare$Diff))
# total samples
nrow(df_compare)

# confusion matrix
tbl_results <- table(df_tst$Exclude, df_tst$Exclude_New, useNA = "ifany")
#
# Show differences
kable(tbl_results)
knitr::kable(df_compare[1:10, ])
knitr::kable(df_compare[672:678, ])
# samples with differences
samp_diff <- as.data.frame(df_compare[df_compare[,"Diff"] != 0, "SampleID"])
# results for only those with differences
df_tst_diff <- df_tst[df_tst[,"SampleID"] %in% samp_diff$SampleID, ]
# add diff field
df_tst_diff$Exclude_Diff <- df_tst_diff$Exclude - df_tst_diff$Exclude_New

# Classification Performance Metrics
class_TP <- tbl_results[2,2] # True Positive
class_FN <- tbl_results[2,1] # False Negative
class_FP <- tbl_results[1,2] # False Positive
class_TN <- tbl_results[1,1] # True Negative
class_n <- sum(tbl_results)  # total
#
```

```
# sensitivity (recall); TP / (TP+FN); measure model to ID true positives
class_sens <- class_TP / (class_TP + class_FN)
# precision; TP / (TP+FP); accuracy of model positives
class_prec <- class_TP / (class_TP + class_FP)
# specifity; TN / (TN + FP); measure model to ID true negatives
class_spec <- class_TN  / (class_TN + class_FP)
# overall accuracy; (TP + TN) / all cases; accuracy of all classifications
class_acc <- (class_TP + class_TN) / class_n
# F1; 2 * (class_prec*class_sens) / (class_prec+class_sens)
## balance of precision and recall
class_F1 <- 2 * (class_prec * class_sens) / (class_prec + class_sens)
#
results_names <- c("Sensitivity (Recall)",
                   "Precision",
                   "Specificity",
                   "Overall Accuracy",
                   "F1")
results_values <- c(class_sens,
                    class_prec,
                    class_spec,
                    class_acc,
                    class_F1)
#
tbl_class <- data.frame(results_names, results_values)
names(tbl_class) <- c("Performance Metrics", "Percent")
tbl_class$Percent <- round(tbl_class$Percent * 100, 2)
kable(tbl_class)

#~~~~~~~~~~~~~~~~~~~~~~~~~~~

# EXAMPLE 2
## No Exceptions

df_tst2 <- markExcluded(df_samps_bugs,
                        SampID = "SampleID",
                        TaxaID = "TaxaID",
                        TaxaCount = "N_Taxa",
                        Exclude = "Exclude_New",
                        TaxaLevels = TaxaLevels,
                        Exceptions = NA)

# Compare
df_compare2 <- dplyr::summarise(dplyr::group_by(df_tst2, SampleID),
                                Exclude_Import = sum(Exclude),
                                Exclude_R = sum(Exclude_New))
df_compare2$Diff <- df_compare2$Exclude_Import - df_compare2$Exclude_R
#
tbl_diff2 <- table(df_compare2$Diff)
kable(tbl_diff2)
# sort
df_compare2 <- df_compare2 %>% arrange(desc(Diff))

# Number with issues
```

```
sum(abs(df_compare2$Diff))
# total samples
nrow(df_compare2)

# confusion matrix
tbl_results2 <- table(df_tst2$Exclude, df_tst2$Exclude_New, useNA = "ifany")
#
# Show differences
kable(tbl_results2)
knitr::kable(df_compare2[1:10, ])
knitr::kable(tail(df_compare2))
# samples with differences
(samp_diff2 <- as.data.frame(df_compare2[df_compare2[, "Diff"] != 0,
                                         "SampleID"]))
# results for only those with differences
df_tst_diff2 <- filter(df_tst2, SampleID %in% samp_diff2$SampleID)
# add diff field
df_tst_diff2$Exclude_Diff <- df_tst_diff2$Exclude - df_tst_diff2$Exclude_New

# Classification Performance Metrics
class_TP2 <- tbl_results2[2,2] # True Positive
class_FN2 <- tbl_results2[2,1] # False Negative
class_FP2 <- tbl_results2[1,2] # False Positive
class_TN2 <- tbl_results2[1,1] # True Negative
class_n2 <- sum(tbl_results2)  # total
#
# sensitivity (recall); TP / (TP+FN); measure model to ID true positives
class_sens2 <- class_TP2 / (class_TP2 + class_FN2)
# precision; TP / (TP+FP); accuracy of model positives
class_prec2 <- class_TP2 / (class_TP2 + class_FP2)
# specifity; TN / (TN + FP); measure model to ID true negatives
class_spec2 <- class_TN2 / (class_TN2 + class_FP2)
# overall accuracy; (TP + TN) / all cases; accuracy of all classifications
class_acc2 <- (class_TP2 + class_TN2) / class_n2
# F1; 2 * (class_prec*class_sens) / (class_prec+class_sens)
## balance of precision and recall
class_F12 <- 2 * (class_prec2 * class_sens2) / (class_prec2 + class_sens2)
#
results_names2 <- c("Sensitivity (Recall)",
                    "Precision",
                    "Specificity",
                    "Overall Accuracy",
                    "F1")
results_values2 <- c(class_sens2,
                     class_prec2,
                     class_spec2,
                     class_acc2,
                     class_F12)
#
tbl_class2 <- data.frame(results_names2, results_values2)
names(tbl_class2) <- c("Performance Metrics", "Percent")
tbl_class2$Percent <- round(tbl_class2$Percent * 100, 2)
kable(tbl_class2)
```

`metric.scores` *Score metrics*

Description

This function calculates metric scores based on a Thresholds data frame. Can generate scores for categories n=3 (e.g., 1/3/5, ScoreRegime="Cat_135") or n=4 (e.g., 0/2/4/6, ScoreRegime="Cat_0246") or continuous (e.g., 0-100, ScoreRegime="Cont_0100").

Usage

```
metric.scores(
  DF_Metrics,
  col_MetricNames,
  col_IndexName,
  col_IndexClass,
  DF_Thresh_Metric,
  DF_Thresh_Index,
  col_ni_total = "ni_total",
  col_IndexRegion = NULL
)
```

Arguments

`DF_Metrics`	Data frame of metric values (as columns), Index Name, and Index Region (strata).
`col_MetricNames`	Names of columns of metric values.
`col_IndexName`	Name of column with index (e.g., MBSS.2005.Bugs)
`col_IndexClass`	Name of column with relevant bioregion or site class (e.g., COASTAL).
`DF_Thresh_Metric`	Data frame of Scoring Thresholds for metrics (INDEX_NAME, INDEX_CLASS, METRIC_NAME, Direction, Thresh_Lo, Thresh_Mid, Thresh_Hi, ScoreRegime , SingleValue_Add, NormDist_Tail_Lo, NormDist_Tail_Hi, CatGrad_xvar , CatGrad_InfPt, CatGrad_Lo_m, CatGrad_Lo_b, CatGrad_Mid_m, CatGrad_Mid_b , CatGrad_Hi_m, CatGrad_Hi_b).
`DF_Thresh_Index`	Data frame of Scoring Thresholds for indices (INDEX_NAME, INDEX_CLASS,METRIC_NAME, ScoreRegime, Thresh01, Thresh02 , Thresh03, Thresh04, Thresh05, Thresh06, Thresh07 , Nar01, Nar02, Nar03, Nar04, Nar05, Nar06).
`col_ni_total`	Name of column with total number of individuals. Used for cases where sample was collected but no organisms collected. Default = ni_total.#'
`col_IndexRegion`	Name of column with relevant bioregion or site class (e.g., COASTAL). Default = NULL. DEPRECATED

Details

The R library dplyr is needed for this function.

For all ScoreRegime cases at the index level a "sum_Index" field is computed that is the sum of all metric scores. Valid "ScoreRegime" values are:

* SUM = all metric scores added together.

* AVERAGE = all metric scores added and divided by the number of metrics. The index is on the same scale as the individual metric scores.

* AVERAGE_100 = AVERAGE is scaled 0 to 100.

FIX, 2024-01-29, v1.0.0.9060 Rename col_IndexRegion to col_IndexClass Add col_IndexRegion as variable at end to avoid breaking existing code Later remove it as an input variable but add code in the function to accept

Value

vector of scores

Examples

```
# Example data

library(readxl)
library(reshape2)

# Thresholds
fn_thresh <- file.path(system.file(package = "BioMonTools"),
                       "extdata",
                       "MetricScoring.xlsx")
df_thresh_metric <- read_excel(fn_thresh, sheet = "metric.scoring")
df_thresh_index <- read_excel(fn_thresh, sheet = "index.scoring")

#~~~~~~~~~~~~~~~~~~~~~~~~
# Pacific Northwest, BCG Level 1 Indicator Taxa Index
df_samps_bugs <- read_excel(system.file("extdata/Data_Benthos.xlsx"
                                        , package = "BioMonTools")
                            , guess_max = 10^6)

myIndex <- "BCG_PacNW_L1"
df_samps_bugs$Index_Name   <- myIndex
df_samps_bugs$Index_Class <- "ALL"
(myMetrics.Bugs <- unique(
  as.data.frame(df_thresh_metric)[df_thresh_metric[,
                                  "INDEX_NAME"] == myIndex, "METRIC_NAME"]))
# Run Function
df_metric_values_bugs <- metric.values(df_samps_bugs,
                                       "bugs",
                                       fun.MetricNames = myMetrics.Bugs)

# index to BCG.PacNW.L1
df_metric_values_bugs$INDEX_NAME <- myIndex
```

```
df_metric_values_bugs$INDEX_CLASS <- "ALL"

# SCORE Metrics
df_metric_scores_bugs <- metric.scores(df_metric_values_bugs,
                                       myMetrics.Bugs,
                                       "INDEX_NAME",
                                       "INDEX_CLASS",
                                       df_thresh_metric,
                                       df_thresh_index)

# QC, table
table(df_metric_scores_bugs$Index, df_metric_scores_bugs$Index_Nar)
# QC, plot
hist(df_metric_scores_bugs$Index,
     main = "PacNW BCG Example Data",
     xlab = "Level 1 Indicator Taxa Index Score")
abline(v = c(21,30), col = "blue")
text(21 + c(-2, +2), 200, c("Low", "Medium"), col = "blue")
```

`metric.stats` *Calculate metric statistics*

Description

This function calculates metric statistics for use with developing a multi-metric index.

Inputs are a data frame with

Usage

```
metric.stats(
  fun.DF,
  col_metrics,
  col_SampID = "SAMPLEID",
  col_RefStatus = "Ref_Status",
  RefStatus_Ref = "Ref",
  RefStatus_Str = "Str",
  RefStatus_Oth = "Oth",
  col_DataType = "Data_Type",
  DataType_Cal = "Cal",
  DataType_Ver = "Ver",
  col_Subset = NULL,
  Subset_Value = NULL
)
```

Arguments

`fun.DF`	Data frame.
`col_metrics`	Column names for metrics.

`col_SampID`	Column name for unique sample identifier. Default = "SAMPLEID".
`col_RefStatus`	Column name for Reference Status. Default = "Ref_Status"
`RefStatus_Ref`	Reference Status name for Reference used in col_ RefStatus. Default = “Ref”. Use NULL if you don’t use this value.
`RefStatus_Str`	Reference Status name for Stressed used in col_ RefStatus. Default = “Str”. Use NULL if you don’t use this value.
`RefStatus_Oth`	Reference Status name for Other used in col_ RefStatus. Default = “Oth”. Use NULL if you don’t use this value.
`col_DataType`	Column name for Data Type – Validation vs. Calibration. Default = "Data_Type"
`DataType_Cal`	Datatype name for Calibration used in col_DataType. Default = “Cal”. Use NULL if you don’t use this value.
`DataType_Ver`	Datatype name for Verification used in col_DataType. Default = “Ver”. Use NULL if you don’t use this value.
`col_Subset`	Column name to subset the data and run on each subset. Default = NULL. If NULL then no subset will be generated.
`Subset_Value`	Subset name to be used for creating subset. Default = NULL.

Details

Summary statistics for the data are calculated.

The data is filtered by the column Subset for only a single value given by the user. If need further subsets re-run the function. If no subset is given the entire data set is used.

Statistics will be generated for up to 6 combinations for RefStatus (Ref, Oth, Str) and DataType (Cal, Ver).

The resulting dataframe will have the statistics in columns with the first 4 columns as: INDEX_CLASS (if col_Subset not provided), col_RefStatus, col_DataType, and Metric_Name.

The following statistics are generated with na.rm = TRUE.

* n = number

* min = minimum

* max = maximum

* mean = mean

* median = median

* range = range (max - min)

* sd = standard deviation

* cv = coefficient of variation (sd/mean)

* q05 = quantile, 5

* q10 = quantile, 10

* q25 = quantile, 25

* q50 = quantile, 50

* q75 = quantile, 75

* q90 = quantile, 90

* q95 = quantile, 95

Value

data frame of metrics (rows) and statistics (columns). This is in long format with columns for INDEX_CLASS, RefStatus, and DataType.

Examples

```
# data, benthos
df_bugs <- data_mmi_dev_small

# Munge Names
names(df_bugs)[names(df_bugs) %in% "BenSampID"]   <- "SAMPLEID"
names(df_bugs)[names(df_bugs) %in% "TaxaID"]      <- "TAXAID"
names(df_bugs)[names(df_bugs) %in% "Individuals"] <- "N_TAXA"
names(df_bugs)[names(df_bugs) %in% "Exclude"]     <- "EXCLUDE"
names(df_bugs)[names(df_bugs) %in% "Class"]       <- "INDEX_CLASS"
names(df_bugs)[names(df_bugs) %in% "Unique_ID"]   <- "SITEID"

# Add Missing Columns
df_bugs$ELEVATION_ATTR <- NA_character_
df_bugs$GRADIENT_ATTR  <- NA_character_
df_bugs$WSAREA_ATTR    <- NA_character_
df_bugs$HABSTRUCT      <- NA_character_
df_bugs$BCG_ATTR2      <- NA_character_
df_bugs$AIRBREATHER    <- NA
df_bugs$UFC            <- NA_real_

# Calc Metrics
cols_keep <- c("Ref_v1",
               "CalVal_Class4",
               "SITEID",
               "CollDate",
               "CollMeth")
# INDEX_NAME and INDEX_CLASS kept by default
df_metval <- metric.values(df_bugs, "bugs", fun.cols2keep = cols_keep)

# Calc Stats
col_metrics   <- names(df_metval)[9:ncol(df_metval)]
col_SampID    <- "SAMPLEID"
col_RefStatus <- "REF_V1"
RefStatus_Ref <- "Ref"
RefStatus_Str <- "Strs"
RefStatus_Oth <- "Other"
col_DataType  <- "CALVAL_CLASS4"
DataType_Cal  <- "cal"
DataType_Ver  <- "verif"
col_Subset    <- "INDEX_CLASS"
Subset_Value  <- "CentralHills"

df_stats <- metric.stats(df_metval,
                         col_metrics,
                         col_SampID,
                         col_RefStatus,
```

```
                          RefStatus_Ref,
                          RefStatus_Str,
                          RefStatus_Oth,
                          col_DataType,
                          DataType_Cal,
                          DataType_Ver,
                          col_Subset,
                          Subset_Value)


# Save Results
write.table(df_stats,
            file.path(tempdir(), "metric.stats.tsv"),
            col.names = TRUE,
            row.names = FALSE,
            sep = "\t")
```

`metric.stats2` *Secondary metric statistics*

Description

This function calculates secondary statistics (DE and z-score) on metric statistics for use with developing a multi-metric index.

Usage

```
metric.stats2(
  data_metval,
  data_metstat,
  col_metval_RefStatus = "RefStatus",
  col_metval_DataType = "DataType",
  col_metval_Subset = "INDEX_CLASS",
  col_metstat_RefStatus = "RefStatus",
  col_metstat_DataType = "DataType",
  col_metstat_Subset = "INDEX_CLASS",
  RefStatus_Ref = "Ref",
  RefStatus_Str = "Str",
  RefStatus_Oth = "Oth",
  DataType_Cal = "Cal",
  DataType_Ver = "Ver",
  Subset_Value = NULL
)
```

Arguments

`data_metval`	Data frame of metric values.

`data_metstat`	Data frame of metric statistics
`col_metval_RefStatus`	Column name for Reference Status. Default = "Ref_Status"
`col_metval_DataType`	Column name for Data Type – Validation vs. Calibration. Default = "Data_Type"
`col_metval_Subset`	Column name for INDEX_CLASS in data_metstats. Default = INDEX_CLASS
`col_metstat_RefStatus`	Column name for Reference Status. Default = "Ref_Status"
`col_metstat_DataType`	Column name for Data Type – Validation vs. Calibration. Default = "Data_Type"
`col_metstat_Subset`	Column name for INDEX_CLASS in data_metstats. Default = xx.
`RefStatus_Ref`	RefStatus value for Reference. Default = "Ref"
`RefStatus_Str`	RefStatus value for Stressed. Default = "Str"
`RefStatus_Oth`	RefStatus value for Other. Default = "Oth"
`DataType_Cal`	DataType value for Calibration. Default = "Cal"
`DataType_Ver`	DataType value for Verification. Default = "Ver"
`Subset_Value`	Subset value of INDEX_CLASS (site class). Default = NULL

Details

Secondary metrics statistics for the data are calculated.

Inputs are metric values and metric stats outputs.

Metric values is a wide format with columns for each metric. Assumes only a single Subset.

Metrics stats is a wide format with columns for each statistic with metrics in a single column. Assumes only a single Subset.

Required fields are RefStatus, DataType, and INDEX_CLASS. The user is allowed to enter their own values for these fields for each input file.

The two statistics calculated are z-score and discrimination efficiency (DE) for each metric within each DataType (cal / val).

Z-scores are calculated using the calibration (or development) data set for a given INDEX_CLASS (or Site Class).

* (mean Ref - mean Str) / sd Ref

DE is calculated without knowing the expected direction of response for each metric for a given INDEX_CLASS (or Site Class). DE is the percentage (0-100) of **stressed** samples that fall **below** the **25th** quantile (for decreaser metrics, e.g., total taxa) or **above** the **75th** quantile (for increaser metrics, e.g., HBI) of the **reference** samples.

A data frame of the metric.stats input is returned with new columns (z_score, DE25 and DE75). The z-score is added for each Ref_Status. DE25 and DE75 are only added where Ref_Status is labeled as Stressed.

Value

A data frame of the metric.stats input is returned with new columns (z_score, DE25 and DE75).

Examples

```
# data, benthos
df_bugs <- data_mmi_dev_small

# Munge Names
names(df_bugs)[names(df_bugs) %in% "BenSampID"]   <- "SAMPLEID"
names(df_bugs)[names(df_bugs) %in% "TaxaID"]      <- "TAXAID"
names(df_bugs)[names(df_bugs) %in% "Individuals"] <- "N_TAXA"
names(df_bugs)[names(df_bugs) %in% "Exclude"]     <- "EXCLUDE"
names(df_bugs)[names(df_bugs) %in% "Class"]       <- "INDEX_CLASS"
names(df_bugs)[names(df_bugs) %in% "Unique_ID"]   <- "SITEID"

# Add Missing Columns
df_bugs$ELEVATION_ATTR <- NA_character_
df_bugs$GRADIENT_ATTR  <- NA_character_
df_bugs$WSAREA_ATTR    <- NA_character_
df_bugs$HABSTRUCT      <- NA_character_
df_bugs$BCG_ATTR2      <- NA_character_
df_bugs$AIRBREATHER    <- NA
df_bugs$UFC            <- NA_real_

# Calc Metrics
cols_keep <- c("Ref_v1",
               "CalVal_Class4",
               "SITEID",
               "CollDate",
               "CollMeth")
# INDEX_NAME and INDEX_CLASS kept by default
df_metval <- metric.values(df_bugs, "bugs", fun.cols2keep = cols_keep)

# Calc Stats
col_metrics   <- names(df_metval)[9:ncol(df_metval)]
col_SampID    <- "SAMPLEID"
col_RefStatus <- "REF_V1"
RefStatus_Ref <- "Ref"
RefStatus_Str <- "Strs"
RefStatus_Oth <- "Other"
col_DataType  <- "CALVAL_CLASS4"
DataType_Cal  <- "cal"
DataType_Ver  <- "verif"
col_Subset    <- "INDEX_CLASS"
Subset_Value  <- "CentralHills"
df_stats <- metric.stats(df_metval,
                         col_metrics,
                         col_SampID,
                         col_RefStatus,
                         RefStatus_Ref,
                         RefStatus_Str,
```

```
                         RefStatus_Oth,
                         col_DataType,
                         DataType_Cal,
                         DataType_Ver,
                         col_Subset,
                         Subset_Value)

# Calc Stats2 (z-scores and DE)
data_metval           <- df_metval
data_metstat          <- df_stats
col_metval_RefStatus  <- "REF_V1"
col_metval_DataType   <- "CALVAL_CLASS4"
col_metval_Subset     <- "INDEX_CLASS"
col_metstat_RefStatus <- "REF_V1"
col_metstat_DataType  <- "CALVAL_CLASS4"
col_metstat_Subset    <- "INDEX_CLASS"
RefStatus_Ref         <- "Ref"
RefStatus_Str         <- "Strs"
RefStatus_Oth         <- "Other"
DataType_Cal          <- "cal"
DataType_Ver          <- "verif"
Subset_Value          <- "CentralHills"
df_stats2 <- metric.stats2(data_metval,
                           data_metstat,
                           col_metval_RefStatus,
                           col_metval_DataType,
                           col_metval_Subset,
                           col_metstat_RefStatus,
                           col_metstat_DataType,
                           col_metstat_Subset,
                           RefStatus_Ref,
                           RefStatus_Str,
                           RefStatus_Oth,
                           DataType_Cal,
                           DataType_Ver,
                           Subset_Value)


# Save Results
write.table(df_stats2,
            file.path(tempdir(), "metric.stats2.tsv"),
            col.names = TRUE,
            row.names = FALSE,
            sep = "\t")
```

`metric.values` *Calculate metric values*

Description

This function calculates metric values for bugs, fish, algae , and coral. Inputs are a data frame with SampleID and taxa with phylogenetic and autecological information (see below for required fields by community). The dplyr package is used to generate the metric values.

Usage

```
metric.values(
  fun.DF,
  fun.Community,
  fun.MetricNames = NULL,
  boo.Adjust = FALSE,
  fun.cols2keep = NULL,
  boo.marine = FALSE,
  boo.Shiny = FALSE,
  verbose = FALSE,
  metric_subset = NULL,
  taxaid_dni = NULL
)
```

Arguments

`fun.DF`	Data frame of taxa (list required fields)
`fun.Community`	Community name for which to calculate metric values (bugs, fish, algae, or coral)
`fun.MetricNames`	Optional vector of metric names to be returned. If none are supplied then all will be returned. Default=NULL
`boo.Adjust`	Optional boolean value on whether to perform adjustments of values prior to scoring. Default = FALSE but may be TRUE for certain metrics.
`fun.cols2keep`	Column names of fun.DF to retain in the output. Uses column names.
`boo.marine`	Should estuary/marine metrics be included. Ignored if fun.MetricNames is not null. Default = FALSE.
`boo.Shiny`	Boolean value for if the function is accessed via Shiny. Default = FALSE.
`verbose`	Include messages to track progress. Default = FALSE
`metric_subset`	Subset of metrics to be generated. Internal function. Default = NULL
`taxaid_dni`	Taxa names to be included in DNI (Do Not Include) metrics (n = 3) but dropped for all other metrics. Only for benthic metrics. Default = NULL

Details

All percent metric results are 0-100.

No manipulations of the taxa are performed by this routine. All benthic macroinvertebrate taxa should be identified to the appropriate operational taxonomic unit (OTU).

Any non-count taxa should be identified in the "Exclude" field as "TRUE". These taxa will be excluded from taxa richness metrics (but will count for all others).

Any non-target taxa should be identified in the "NonTarget" field as "TRUE". Non-target taxa are those that are not part of your intended #' capture list; e.g., fish, herps, water column taxa, or water surface taxa in a benthic sample. The target list will vary by program. The non-target taxa will be removed prior to any calculations.

Excluded taxa are ambiguous taxa (on a sample basis), i.e., the parent taxa when child taxa are present. For example, the parent taxa Chironomidae would be excluded when the child taxa Tanytarsini is present. Both would be excluded when Tanytarsus is present. The markExcluded function can be used to populated this field.

There are a number of required fields (see below) for metric to calculation. If any fields are missing the user will be prompted as to which are missing and if the user wants to continue or quit. If the user continues the missing fields will be added but will be filled with zero or NA (as appropriate). Any metrics based on the missing fields will not be valid.

A future update may turn these fields into function parameters. This would allow the user to tweak the function inputs to match their data rather than having to update their data to match the function.

Required fields, all communities:

* SAMPLEID (character or number, must be unique)

* TAXAID (character or number, must be unique)

* N_TAXA

* INDEX_NAME

* INDEX_CLASS (BCG or MMI site category; e.g., for BCG PacNW valid values are "hi" or "lo")

Additional Required fields, bugs:

* EXCLUDE (valid values are TRUE and FALSE)

* NONTARGET (valid values are TRUE and FALSE)

* PHYLUM, SUBPHYLUM, CLASS, SUBCLASS, INFRAORDER, ORDER, FAMILY, SUBFAMILY, TRIBE, GENUS

* FFG, HABIT, LIFE_CYCLE, TOLVAL, BCG_ATTR, THERMAL_INDICATOR, FFG2, TOLVAL2, LONGLIVED, NOTEWORTHY, HABITAT, UFC, ELEVATION_ATTR, GRADIENT_ATTR, WSAREA_ATTR, HABSTRUCT

Additional Required fields, fish:

* N_ANOMALIES

* SAMP_BIOMASS (biomass total for sample, funciton uses max in case entered for all taxa in sample)

* NATIVE: NATIVE or other text values

* DA_MI2, SAMP_WIDTH_M, SAMP_LENGTH_M, , TYPE, TOLER, TROPHIC, SILT, FAMILY, GENUS, HYBRID, BCG_ATTR, THERMAL_INDICATOR, ELEVATION_ATTR, GRADIENT_ATTR, WSAREA_ATTR, REPRODUCTION, HABITAT, CONNECTIVITY, SCC

Additional Required fields, algae:

* EXCLUDE, NONTARGET, PHYLUM, ORDER, FAMILY, GENUS, BC_USGS, TROPHIC_USGS, SAP_USGS, PT_USGS, O_USGS, SALINITY_USGS, BAHLS_USGS, P_USGS, N_USGS, HABITAT_USGS, N_FIXER_USGS, MOTILITY_USGS, SIZE_USGS, HABIT_USGS, MOTILE2_USGS, TOLVAL, DIATOM_ISA, DIAT_CL, POLL_TOL, BEN_SES, DIATAS_TP, DIATAS_TN, DIAT_COND, DIAT_CA, MOTILITY, NF

Valid values for fields:

* FFG: CG, CF, PR, SC, SH

* HABIT: BU, CB, CN, SP, SW

* LIFE_CYCLE: UNI, SEMI, MULTI

* THERMAL_INDICATOR: STENOC, COLD, COOL, WARM, STENOW, EURYTHERMAL , COWA, NA

* LONGLIVED: TRUE, FALSE

* NOTEWORTHY: TRUE, FALSE

* HABITAT: BRAC, DEPO, GENE, HEAD, RHEO, RIVE, SPEC, UNKN

* UFC: integers 1:6 (taxonomic uncertainty frequency class)

* ELEVATION_ATTR: LOW, HIGH

* GRADIENT_ATTR: LOW, MOD, HIGH

* WSAREA_ATTR: SMALL, MEDIUM, LARGE, XLARGE

* REPRODUCTION: BROADCASTER, SIMPLE NEST, COMPLEX NEST, BEARER, MIGRATORY

* CONNECTIVITY: TRUE, FALSE

* SCC (Species of Conservation Concern): TRUE, FALSE

'Columns to keep' are additional fields in the input file that the user wants retained in the output. Fields need to be those that are unique per sample and not associated with the taxa. For example, the fields used in qc.check(); Area_mi2, SurfaceArea, Density_m2, and Density_ft2.

If fun.MetricNames is provided only those metrics will be returned in the provided order. This variable can be used to sort the metrics per the user's preferences. By default the metric names will be returned in the groupings that were used for calculation.

The fields TOLVAL2 and FFG2 are provided to allow the user to calculate metrics based on alternative scenarios. For example, including both HBI and NCBI where the NCBI uses a different set of tolerance values (TOLVAL2).

If TAXAID is 'NONE' and N_TAXA is '0' then metrics **will** be calculated with that record. Other values for TAXAID with N_TAXA = 0 will be removed before calculations.

For 'Oligochete' metrics either Class or Subclass is required for calculation.

The parameter boo.Shiny can be set to TRUE when accessing this function in Shiny. Normally the QC check for required fields is interactive. Setting boo.Shiny to TRUE will always continue. The default is FALSE.

The parameter 'taxaid_dni' denotes taxa to be included in Do Not Include (DNI) metrics but dropped from all other metrics. Only for benthic metrics.

Breaking change from 0.5 to 0.6 with change from Index_Name to Index_Class.

Value

data frame of SampleID and metric values

Examples

```
# Example 1, data already in R

df_metval <- metric.values(BioMonTools::data_benthos_PacNW,
                           "bugs")

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
# Example 2, specific metrics or metrics in a specific order
## reuse df_samps_bugs from above

# metric names to keep (in this order)
myMetrics <- c("ni_total",
               "nt_EPT",
               "nt_Ephem",
               "pi_tv_intol",
               "pi_Ephem",
               "nt_ffg_scrap",
               "pi_habit_climb")

# Run Function
df_metval_myMetrics <- metric.values(BioMonTools::data_benthos_PacNW,
                                     "bugs",
                                     fun.MetricNames = myMetrics)
```

`metvalgrpxl` *Metric values Groups to Excel*

Description

The output of metric.values() is saved to Excel with different groups of metrics on different worksheets.

Usage

```
metvalgrpxl(
  fun.DF.MetVal,
  fun.DF.xlMetNames = NULL,
  fun.Community,
  fun.MetVal.Col2Keep = c("SAMPLEID", "INDEX_NAME", "INDEX_CLASS"),
  fun.xlGrpCol = "Sort_Group",
  file.out = NULL
)
```

Arguments

`fun.DF.MetVal`	Data frame of metric values.

`fun.DF.xlMetNames`	Data frame of metric names and groups. Default (NULL) will use the verion of MetricNames.xlsx that is in the BioMonTools package.
`fun.Community`	Community name of calculated metric values (bugs, fish, or algae)
`fun.MetVal.Col2Keep`	Column names in metric values to keep. Default = c("SAMPLEID", "INDEX_NAME", "INDEX_CLASS")
`fun.xlGrpCol`	Column name from Excel metric names to use for Groupings. Default = Sort_Group
`file.out`	Output file name. Default (NULL) will generate a file name based on the data and time (e.g., MetricValuesGroups_bugs_20220201.xlsx)

Details

This function will save the output of metric.values() into groups by worksheet as defined by the user.

The Excel file MetricNames.xlsx provided in the extdata folder has a column named 'Groups' that can be used as default groupings. If no groupings are provided (the default) all metrics are saved to a single worksheet. Within each group the 'sort_order' is used to sort the metrics. If this column is blank then the metrics are sorted in the order they appear in the output from metric.values() (i.e., in fun.DF).

The MetricNames data frame must include the following fields:

* Metric_Name

* Community

* Sort_Group (user defined)

Value

Saves Excel file with metrics grouped by worksheet

Examples

```
# Example 1, bugs
## Community
comm <- "bugs"
## Calculate Metrics
df_metval <- metric.values(BioMonTools::data_benthos_PacNW, comm)
## Metric Names and Groups
df_metnames <- readxl::read_excel(system.file("extdata/MetricNames.xlsx",
                                              package="BioMonTools"),
                                  guess_max = 10^6,
                                  sheet = "MetricMetadata",
                                  skip = 4)
## Columns to Keep
col2keep <- c("SAMPLEID", "INDEX_NAME", "INDEX_CLASS")
## Grouping Column
col_Grp <- "Sort_Group"
## File Name
file_out <- file.path(tempdir(), paste0("MetValGrps_", comm, ".xlsx"))
```

```
## Run Function
metvalgrpxl(df_metval, df_metnames, comm, col2keep, col_Grp, file_out)
```

---

qc.checks *QC checks on metric values*

Description

Apply "QC checks" on calculated metrics and station/sample attributes to "flag" samples for the user. Examples include watershed size or total number of individuals. Can have checks for both high and low values. Checks are stored in separate file. For structure see df.checks in example.

Usage

```
qc.checks(df.metrics, df.checks, input.shape = "wide")
```

Arguments

`df.metrics`	Wide data frame with metric values to be evaluated.
`df.checks`	Data frame of metric thresholds to check.
`input.shape`	Shape of df.metrics; wide or long. Default is wide.

Details

used reshape2 package

Value

Returns a data frame of SampleID checks and results; Pass and Fail.

Examples

```
library(readxl)

# Calculate Metrics
df.samps.bugs <- read_excel(system.file("./extdata/Data_Benthos.xlsx",
                                        package="BioMonTools"),
                            guess_max = 10^6)

# Columns to keep
myCols <- c("Area_mi2", "SurfaceArea", "Density_m2", "Density_ft2")

# Run Function
myDF <- df.samps.bugs
df.metric.values.bugs <- metric.values(myDF, "bugs", fun.cols2keep = myCols)

# Import Checks
df.checks <- read_excel(system.file("./extdata/MetricFlags.xlsx",
```

```
                                    package="BioMonTools"),
                              sheet="Flags")

# Run Function
df.flags <- qc.checks(df.metric.values.bugs, df.checks)

# Summarize Results
table(df.flags[, "CHECKNAME"], df.flags[, "FLAG"], useNA = "ifany")
```

`qc_taxa` *Quality Control Check on User Data Against Master Taxa List*

Description

This function compares the user's data frame to a data frame with the official (or user supplied) master taxa list (benthic macroinvertebrates).

Usage

```
qc_taxa(
  DF_User,
  DF_Official = NULL,
  fun.Community = NULL,
  useOfficialTaxaInfo = "only_Official"
)
```

Arguments

`DF_User`	User taxa data.
`DF_Official`	Official master taxa list. Can be a local file or from a URL. Default is NULL. A NULL value will use the official online files.
`fun.Community`	Community name for which to compare the master taxa list (bugs or fish).
`useOfficialTaxaInfo`	Select how to handle new/different taxa. See 'Details' for more information. Valid values are "only_Official", "only_user", "add_new". Default = "only_Official".

Details

Output is a data frame with matches.

Messages are output to the console with the number of matches and which user taxa did not match the official list.

The official list is stored online but the user can input their own saved copy.

Any columns in the user input file that match the official master taxa list will be renamed with the "_NonOfficial" suffix.

New/different taxa in the user data are handled by the 'useOfficialTaxaInfo' parameter. For taxa that did not match the master taxa list the user has options on how to handle the differences for the phylogeny (e.g., columns for phylum, class, family, etc.) and autecology (e.g., columns for FFG, habit, tolerance value, etc.). The options are below.

* only_official = use only official master taxa information. Any non-matching taxa will not have any master taxa information.

* only_user = only use the information provided by the user. Information from the 'Official' will not be used. This should only be used for non-official calculations.

* add_new = hybrid approach that uses official master taxa information, when present, but includes user information for non-matching taxa if the column names match.

Default master taxa lists are saved as CSV files online at:

https://github.com/leppott/MBSStools_SupportFiles

The files can be downloaded with the following code.

Benthic Macroinvertebrate

url_mt_bugs <- "https://github.com/leppott/MBSStools_SupportFiles/raw/master/Data/CHAR_Bugs.csv"
df_mt_bugs <- read.csv(url_mt_bugs)

The master taxa files are periodically updated. Update dates will be logged on the GitHub repository.

Expected fields include:

Benthic Macroinvertebrates

+ TAXON, Phylum, Class, Order, Family, Genus, Other_Taxa, Tribe, FFG, FAM_TV, Habit, FinalTolVal07, Comment

Value

input data frame with master taxa information added to it.

Examples

```
# Example 1, Master Taxa List, Bugs
url_mt_bugs <- "https://github.com/leppott/MBSStools_SupportFiles/raw/master/Data/CHAR_Bugs.csv"
df_mt_bugs  <- read.csv(url_mt_bugs)

# User data
DF_User <- data_benthos_MBSS
DF_Official <- NULL   # NULL df_mt_bugs
fun.Community <- "bugs"
useOfficialTaxaInfo <- "only_Official"
# modify taxa id column
DF_User[, "TAXON"] <- DF_User[, "TAXAID"]

df_qc_taxa_bugs <- qc_taxa(DF_User,
                           DF_Official,
                           fun.Community,
                           useOfficialTaxaInfo)

# QC input/output
dim(DF_User)
dim(df_qc_taxa_bugs)
```

```
names(DF_User)
names(df_qc_taxa_bugs)
```

`rarify` *Rarify (subsample) biological sample to fixed count*

Description

Takes as an input a 3 column data frame (SampleID, TaxonID , Count) and returns a similar dataframe with revised Counts.

The other inputs are subsample size (target number of organisms in each sample) and seed. The seed is given so the results can be reproduced from the same input file. If no seed is given a random seed is used.

Usage

```
rarify(inbug, sample.ID, abund, subsiz, mySeed = NA, verbose = FALSE)
```

Arguments

`inbug`	Input data frame. Needs 3 columns (SampleID, taxonomicID , Count).
`sample.ID`	Column name in inbug for sample identifier.
`abund`	Column name in inbug for organism count.
`subsiz`	Target subsample size for each sample.
`mySeed`	Seed for random number generator. If provided the results with the same inbug file will produce the same results. Default = NA (random seed will be used.)
`verbose`	Boolean value for if status messages are output to the console. Default = FALSE

Details

rarify function: R function to rarify (subsample) a macroinvertebrate sample down to a fixed count; by John Van Sickle, USEPA. email: VanSickle.John@epa.gov ; Version 1.0, 06/10/05;

Value

Returns a data frame with the same three columns but the abund field has been modified so the total count for each sample is no longer above the target (subsiz).

Examples

```
# Subsample to 500 organisms (from over 500 organisms) for 12 samples.

# load bio data
df_biodata <- data_bio2rarify
dim(df_biodata)

# subsample
```

```
mySize  <- 500
Seed_OR <- 18590214
Seed_WA <- 18891111
Seed_US <- 17760704
bugs_mysize <- rarify(inbug = df_biodata,
                      sample.ID = "SampleID",
                      abund = "N_Taxa",
                      subsiz = mySize,
                      mySeed = Seed_US,
                      verbose = FALSE)

# view results
dim(bugs_mysize)

# Compare pre- and post- subsample counts
df_compare <- merge(df_biodata,
                    bugs_mysize,
                    by = c("SampleID", "TaxaID"),
                    suffixes = c("_Orig","_500"))
df_compare <- df_compare[, c("SampleID",
                             "TaxaID",
                             "N_Taxa_Orig",
                             "N_Taxa_500")]

# compare totals
tbl_totals <- aggregate(cbind(N_Taxa_Orig, N_Taxa_500) ~ SampleID,
                        df_compare,
                        sum)


# save the data
write.table(bugs_mysize,
            file.path(tempdir(), paste("bugs", mySize, "txt", sep = ".")),
            sep = "\t")
```

`TaxaMaster_Ben_BCG_PacNW`

TaxaMaster_Ben_BCG_PacNW

Description

Example data

Usage

```
TaxaMaster_Ben_BCG_PacNW
```

Format

A data frame with 684 observations on the following 20 variables.

`TaxaID` a character vector

`Phylum` a character vector

`SubPhylum` a character vector

`Class` a character vector

`SubClass` a character vector

`Order` a character vector

`SuperFamily` a character vector

`Family` a character vector

`Tribe` a character vector

`Genus` a character vector

`SubGenus` a character vector

`Species` a character vector

`BCG_Attr` a character vector

`NonTarget` a logical vector

`Thermal_Indicator` a character vector

`Long_Lived` a character vector

`FFG` a character vector

`Habit` a character vector

`Life_Cycle` a character vector

`TolVal` a numeric vector

Source

example master taxa from BCG Pacific Northwest

`taxa_translate` *Taxa Translate*

Description

Convert user taxa names to those in an official project based name list.

Usage

```
taxa_translate(
  df_user = NULL,
  df_official = NULL,
  df_official_metadata = NULL,
  taxaid_user = "TAXAID",
  taxaid_official_match = NULL,
  taxaid_official_project = NULL,
  taxaid_drop = NULL,
  col_drop = NULL,
  sum_n_taxa_boo = FALSE,
  sum_n_taxa_col = NULL,
  sum_n_taxa_group_by = NULL,
  trim_ws = FALSE,
  match_caps = FALSE
)
```

Arguments

`df_user`	User taxa data
`df_official`	Official project taxa data (master taxa list).
`df_official_metadata`	Metadata for official project taxa data. Default is NULL
`taxaid_user`	Taxonomic identifier in user data. Default is "TAXAID".
`taxaid_official_match`	Taxonomic identifier in official data user to match with user data. This is not the project taxanomic identifier.
`taxaid_official_project`	Taxonomic identifier in official data that is specific to a project, e.g., after operational taxonomic unit (OTU) applied.
`taxaid_drop`	Official taxonomic identifier that signals a record should be dropped; e.g., DNI (Do Not Include) or -999. Default = NULL
`col_drop`	Columns to remove in output. Default = NULL
`sum_n_taxa_boo`	Boolean value for if the results should be summarized Default = FALSE DEPRECATED, values will be ignored
`sum_n_taxa_col`	Column name for number of individuals for user data when summarizing. This column will be summed. Default = NULL (suggestion = N_TAXA) DEPRECATED, values will be ignored
`sum_n_taxa_group_by`	Column names for user data to use for grouping the data when summarizing the user data. Suggestions are SAMPID and TAXA_ID. Default = NULL DEPRECATED, values will be ignored
`trim_ws`	Boolean value for taxaid to have leading and trailing white space removed. Non-braking spaces (e.g., from ITIS) also removed (including inside text). Default = FALSE
`match_caps`	Boolean value to match user and official TaxaIDs after converting to ALL CAPS. Default = FALSE

Details

Merges user file with official file. The official file has phylogeny, autecology, and other project specific fields.

The inputs for the function uses existing data frames (or tibbles).

Any fields that match between the user file and the official file the official data column name have the 'official' version retained.

The 'col_drop' parameter can be used to remove unwanted columns; e.g., the other taxa id fields in the 'official' data file.

By default, taxa are not collapsed to the official taxaid. That is, if multiple taxa in a sample have the same name the rows will not be combined. If collapsing is desired set the parameter 'sum_n_taxa_boo' to TRUE. Will also need to provide 'sum_n_taxa_col' and 'sum_n_taxa_group_by'. This feature was DEPRECATED in v1.0.2.9040 (2024-06-12). The parameters will remain and could be reinstituted in a future version.

Slightly different than 'qc_taxa' since no options in 'taxa_translate' for using one field over another and is more generic.

The parameter 'taxaid_drop' is used to drop records that matched to a new name that should not be included in the results. Examples include "999" or "DNI" (Do Not Include). Default is NULL so no action is taken. "NA"s are always removed.

Optional parameter 'trim_ws' is used to invoke the function 'trimws' to remove from the taxa matching field any leading and trailing white space. Default is FALSE (no action). All horizontal and vertical white space characters are removed. See ?trimws for additional information. Additionally, non-breaking spaces (nbsp) inside the text string will be replaced with a normal space. This cuts down on the number of permutations need to be added to the translation table.

Optional parameter 'match_caps' is used to convert user and official taxaid values to ALL CAPS before matching. Any non-ascii characters will cause this to fail. A message is output to the console for any taxaid values that contain non-ascii characters. In the event that 'match_caps' is set to TRUE and non-ascii characters are present the matching will be done without converting to upper case as this would cause the function to fail.

The taxa list and metadata file names will be added to the results as two new columns.

Another output is the unique taxa with old and new names.

Value

A list with four elements. The first (merge) is the user data frame with additional columns from the official data appended to it. Names from the user data that overlap with the official data have the suffix '_User'. The second element (nonmatch) of the list is a vector of the non-matching taxa from the user data. The third element (metadata) includes the metadata for the official data (if provided). The fourth element (unique) is a data frame of the unique taxa names old and new.

Examples

```
# Example 1, PacNW
## Input Parameters
df_user <- BioMonTools::data_benthos_PacNW
fn_official <- file.path(system.file("extdata", package = "BioMonTools"),
                         "taxa_official",
```

```
                         "ORWA_TAXATRANSLATOR_20221219b.csv")
df_official <- read.csv(fn_official)
fn_official_metadata <- file.path(system.file("extdata",
                                              package = "BioMonTools"),
                                  "taxa_official",
                                  "ORWA_ATTRIBUTES_METADATA_20221117.csv")
df_official_metadata <- read.csv(fn_official_metadata)
taxaid_user <- "TaxaID"
taxaid_official_match <- "Taxon_orig"
taxaid_official_project <- "OTU_MTTI"
taxaid_drop <- "DNI"
col_drop <- c("Taxon_v2", "OTU_BCG_MariNW") # non desired ID cols in Official
sum_n_taxa_boo <- TRUE
sum_n_taxa_col <- "N_TAXA"
sum_n_taxa_group_by <- c("INDEX_NAME", "INDEX_CLASS", "SampleID", "TaxaID")
## Run Function

taxatrans <- taxa_translate(df_user,
                            df_official,
                            df_official_metadata,
                            taxaid_user,
                            taxaid_official_match,
                            taxaid_official_project,
                            taxaid_drop,
                            col_drop,
                            sum_n_taxa_boo,
                            sum_n_taxa_col,
                            sum_n_taxa_group_by)
## View Results
taxatrans$nonmatch


#~~~~~
# Example 2, Multiple Stages
# Create data
TAXAID <- c(rep("Agapetus", 3), rep("Zavrelimyia", 2))

N_TAXA <- c(rep(33, 3), rep(50, 2))
STAGE <- c("A", "L", "P", "X", "")
df_user <- data.frame(TAXAID, N_TAXA, STAGE)
df_user[, "INDEX_NAME"]  <- "BCG_MariNW_Bugs500ct"
df_user[, "INDEX_CLASS"] <- "HiGrad-HiElev"
df_user[, "SAMPLEID"]    <- "Test2023"
df_user[, "STATIONID"]   <- "Test"
df_user[, "DATE"]        <- "2023-01-16"
## Input Parameters
fn_official <- file.path(system.file("extdata", package = "BioMonTools"),
                         "taxa_official",
                         "ORWA_TAXATRANSLATOR_20221219b.csv")
df_official <- read.csv(fn_official)
fn_official_metadata <- file.path(system.file("extdata",
                                              package = "BioMonTools"),
                                  "taxa_official",
```

```
                                  "ORWA_ATTRIBUTES_20221212.csv")
df_official_metadata <- read.csv(fn_official_metadata)
taxaid_user <- "TAXAID"
taxaid_official_match <- "Taxon_orig"
taxaid_official_project <- "OTU_BCG_MariNW"
taxaid_drop <- NULL
col_drop <- c("Taxon_v2", "OTU_MTTI") # non desired ID cols in Official
sum_n_taxa_boo <- TRUE
sum_n_taxa_col <- "N_TAXA"
sum_n_taxa_group_by <- c("INDEX_NAME", "INDEX_CLASS", "SAMPLEID", "TAXAID")
## Run Function
taxatrans <- taxa_translate(df_user,
                            df_official,
                            df_official_metadata,
                            taxaid_user,
                            taxaid_official_match,
                            taxaid_official_project,
                            taxaid_drop,
                            col_drop,
                            sum_n_taxa_boo,
                            sum_n_taxa_col,
                            sum_n_taxa_group_by)
## View Results (before and after)
df_user
taxatrans$merge
```

Index