Package 'tidywater'

November 14, 2026

Type Package

Title Water Quality Models for Drinking Water Treatment Processes

Version 0.11.0

URL https://github.com/BrownandCaldwell-Public/tidywater,
https://brownandcaldwell-public.github.io/tidywater/

BugReports https://github.com/BrownandCaldwell-Public/tidywater/issues

Description Provides multiple water chemistry-
based models and published empirical models in one standard format.
As many models have been included as possible, however, users should be aware that models have varying degrees of accuracy and applicability.
To learn more, read the references provided below for the models implemented.
Functions can be chained together to model a complete treatment process and are designed to work in a 'tidyverse' workflow.
Models are primarily based on these sources:
Benjamin, M. M. (2002, ISBN:147862308X),
Crittenden, J. C., Trussell, R., Hand, D., Howe, J. K., & Tchobanoglous, G., Borchardt, J. H. (2012, ISBN:9781118131473),
USEPA. (2001) <https://www.epa.gov/sites/default/files/2017-03/documents/wtp_model_v._2.0_manual_508.pdf>.

License Apache License (>= 2) | MIT + file LICENSE

Encoding UTF-8

LazyData true

Imports dplyr, tidyr, forcats, knitr, ggplot2, ggrepel, magrittr,
methods, rlang, deSolve, lifecycle, cli

Depends R (>= 4.1)

Suggests rmarkdown, purrr, testthat (>= 3.0.0)

Config/testthat/edition 3

VignetteBuilder knitr

Date 2026-10-29

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.3

NeedsCompilation no

Author Sierra Johnson [aut, cre],
Libby McKenna [aut],
Riley Mulhern [aut] (ORCID: <https://orcid.org/0000-0001-6293-3672>),
Chris Corwin [aut] (ORCID: <https://orcid.org/0000-0002-9462-0352>),
Rachel Merrifield [ctb],
Mayuri Namasivayam [ctb],
Phoebe Chen [ctb],
Jiaming Yuan [ctb],
Atticus Fretz [ctb],
Bengu Kilinc [ctb],
Hania Zanib [ctb],
USEPA [cph] (Copyright holder of included TELSS fragments (dissolve_pb function)),
Brown and Caldwell [fnd, cph]

Maintainer Sierra Johnson <sjohnson2@brwncald.com>

Repository CRAN

Date/Publication 2026-11-14 00:30:02 UTC

Contents

balance_ions . . . 3
biofilter_toc . . . 5
blend_waters . . . 7
bromatecoeffs . . . 9
calculate_activity . . . 10
calculate_corrosion . . . 10
calculate_hardness . . . 13
check_water . . . 13
chemdosetoccoeff . . . 14
chemdose_chloramine . . . 15
chemdose_chlordecay . . . 17
chemdose_dbp . . . 20
chemdose_inorgdemand . . . 22
chemdose_ph . . . 24
chemdose_toc . . . 29
chloramine_conv . . . 32
cl2coeffs . . . 32
convert_flow . . . 33
convert_units . . . 33
convert_water . . . 34
correct_k . . . 35
dbpcoeffs . . . 36
dbp_correction . . . 37
decarbonate_ph . . . 37

define_chemical . . . 39
define_water . . . 40
discons . . . 46
dissolve_cu . . . 47
dissolve_cu_df . . . 48
dissolve_pb . . . 48
gacbv_toc . . . 51
gacrun_toc . . . 53
gac_toc . . . 55
leadsol_constants . . . 57
modify_water . . . 58
mweights . . . 59
opensys_ph . . . 60
ozonate_bromate . . . 61
pactoccoeffs . . . 63
pac_toc . . . 64
plot_ions . . . 65
plot_lead . . . 66
pluck_water . . . 67
regulate_toc . . . 68
scavenge_oh . . . 70
solvecost_chem . . . 72
solvecost_labor . . . 73
solvecost_power . . . 74
solvecost_solids . . . 74
solvect_chlorine . . . 76
solvect_o3 . . . 78
solvedose_alk . . . 79
solvedose_ph . . . 81
solvemass_chem . . . 82
solvemass_solids . . . 83
solveresid_o3 . . . 84
summarize_wq . . . 86
toc_compliance_table . . . 87
vlog_removalcts . . . 87
water_df . . . 88

Index **89**

`balance_ions` *Add an ion to balance overall charge in a water*

Description

This function takes a water defined by define_water and balances charge. For a single water use `balance_ions`; for a dataframe use `balance_ions_df`. Use `pluck_cols = TRUE` to get values from the output water as new dataframe columns.

Usage

```
balance_ions(water, anion = "cl", cation = "na")

balance_ions_df(
  df,
  input_water = "defined",
  output_water = "balanced",
  pluck_cols = FALSE,
  water_prefix = TRUE,
  anion = "cl",
  cation = "na"
)
```

Arguments

water	Water created with define_water, which may have some ions set to 0 when unknown
anion	Selected anion to use to for ion balance when more cations are present. Defaults to "cl". Choose one of c("cl", "so4").
cation	Selected cation to use to for ion balance when more anions are present. Defaults to "na". Choose one of c("na", "k", "ca", or "mg").
df	a data frame containing a water class column, which has already been computed using define_water_df
input_water	name of the column of water class data to be used as the input for this function. Default is "defined_water".
output_water	name of the output column storing updated water classes. Default is "balanced_water".
pluck_cols	Extract water slots modified by the function (selected cation and anion) into new numeric columns for easy access. Default to FALSE.
water_prefix	Append the output_water name to the start of the plucked columns. Default is TRUE.

Details

If more cations are needed, sodium will be added. User may specify which cation ("na", "k", "ca", or "mg") to use for balancing. If calcium and magnesium are not specified when defining a water with define_water, they will default to 0 and not be changed by this function unless specified in the cation argument. Anions are added by default with chloride. User may specify which anion ("cl", "so4") to use for balancing. This function is purely mathematical. User should always check the outputs to make sure values are reasonable for the input source water.

Value

`balance_ions` returns a single water class object with updated ions to balance water charge.

`balance_ions_df` returns a dataframe with a new column with the ion balanced water

Examples

```
water_defined <- define_water(7, 20, 50, 100, 80, 10, 10, 10, 10) %>%
  balance_ions()

water_defined <- define_water(7, 20, 50, tot_hard = 150) %>%
  balance_ions(anion = "so4")

example_df <- water_df %>%
  define_water_df() %>%
  balance_ions_df(anion = "so4", cation = "ca")
```

`biofilter_toc` *Determine TOC removal from biofiltration using Terry & Summers, Peterson & Summers, or Terry et al models*

Description

This function applies the Terry model to a water created by define_water to determine biofiltered DOC (mg/L). The Terry & Summers (2018) model does not adjust for ozone dose but assumes a constant biodegradable fraction based on whether or not the water was ozonated. When the reuse argument is TRUE, the Peterson and Summers (2021) model is used to determine biofiltered TOC (mg/L). The Peterson & Summers (2021) model does not adjust for temperature impacts due to a lack of collected temperature data in the metanalysis literature. Peterson & Summers (2021) is also a TOC-basis model, so the output water object's BDOC slot will not be updated. If a lab-measured ATP concentration is known, the Terry et al. (2024) model can be used to determine biofiltered DOC (mg/L) for both drinking water and reuse. The Terry et al. (2024) model has similar BDOC assumptions as Terry & Summers (2018), but the model ATP value can be adjusted for biofilter water temperature. For all three models, all particulate TOC is assumed to be removed so TOC = DOC. For a single water use `biofilter_toc`; for a dataframe use `biofilter_toc_df`. Use `pluck_cols = TRUE` to get values from the output water as new dataframe columns. For most arguments in the `_df` helper "use_col" default looks for a column of the same name in the dataframe. The argument can be specified directly in the function instead or an unquoted column name can be provided.

Usage

```
biofilter_toc(
  water,
  ebct,
  ozonated = TRUE,
  ozone_dose = 0,
  reuse = FALSE,
  media_type = "anthracite",
  atp = 0
)
```

```
biofilter_toc_df(
  df,
  input_water = "defined",
  output_water = "biofiltered",
  pluck_cols = FALSE,
  water_prefix = TRUE,
  ebct = "use_col",
  ozonated = "use_col",
  ozone_dose = "use_col",
  reuse = "use_col",
  media_type = "use_col",
  atp = "use_col"
)
```

Arguments

water	Source water object of class "water" created by define_water.
ebct	The empty bed contact time (min) used for the biofilter.
ozonated	Logical; TRUE if the water is ozonated (default), FALSE otherwise. Used by the Terry & Summers (2018) model and to select the BDOC fraction for reuse waters in the Terry et al. (2024) ATP model. Ignored by the Peterson & Summers (2021) model, which uses ozone_dose instead.
ozone_dose	The ozone dose (mg/L) applied to the water. Only used if reuse = TRUE.
reuse	Logical; TRUE if the water is effluent from a reuse treatment process, FALSE otherwise. If TRUE, the Peterson and Summers (2021) model is used to determine TOC removal. If FALSE, the Terry and Summers (2018) model is used to determine BDOC removal.
media_type	The type of media used in the biofilter. Default is "anthracite", other options are "gac" and "sand". Only used if reuse = TRUE.
atp	The measured adenosine triphosphate (ATP) concentration at the top of the biofilter (ng/mL at 22 degrees Celcius). Optional argument that will use Terry et al. (2024) model for both drinking water and reuse if nonzero value is provided. (This will also ignore ozone dose and media type for reuse models.) Defaults to 0.
df	a data frame containing a water class column, which has already been computed using define_water_df. The df may include a column indicating the EBCT or whether the water is ozonated.
input_water	name of the column of water class data to be used as the input for this function. Default is "defined".
output_water	name of the output column storing updated water class object. Default is "biofiltered".
pluck_cols	Extract water slots modified by the function (doc, toc, bdoc) into new numeric columns for easy access. Default to FALSE.
water_prefix	Append the output_water name to the start of the plucked columns. Default is TRUE.

Value

biofilter_toc returns water class object with modeled DOC removal from biofiltration.

biofilter_toc_df returns a data frame containing a water class column with updated DOC, TOC, and BDOC concentrations. Optionally, it also adds columns for each of those slots individually.

Source

Terry and Summers 2018

Peterson and Summers 2021

Terry et al. 2024

Examples

```
water <- define_water(ph = 7, temp = 25, alk = 100, toc = 5.0, doc = 4.0, uv254 = .1) %>%
  biofilter_toc(ebct = 10, ozonated = FALSE)


example_df <- water_df %>%
  define_water_df() %>%
  biofilter_toc_df(input_water = "defined", ebct = c(10, 15), ozonated = FALSE)

example_df <- water_df %>%
  define_water_df() %>%
  dplyr::mutate(
    BiofEBCT = c(10, 10, 10, 15, 15, 15, 20, 20, 20, 25, 25, 25),
    ozonated = c(rep(TRUE, 6), rep(FALSE, 6))
  ) %>%
  biofilter_toc_df(input_water = "defined", ebct = BiofEBCT)
```

---

blend_waters *Determine blended water quality from multiple waters based on mass balance and acid/base equilibrium*

Description

This function takes a vector of waters defined by define_water and a vector of ratios and outputs a new water object with updated ions and pH. For a single blend use blend_waters; for a dataframe use blend_waters_df. Use pluck_water to get values from the output water as new dataframe columns.

Usage

```
blend_waters(waters, ratios)

blend_waters_df(df, waters, ratios, output_water = "blended")
```

Arguments

waters	Vector of source waters created by define_water. For df function, this can include quoted column names and/or existing single water objects unquoted.
ratios	Vector of ratios in the same order as waters. (Blend ratios must sum to 1). For df function, this can also be a list of quoted column names.
df	a data frame containing a water class column, which has already been computed using define_water_df
output_water	name of output column storing updated parameters with the class, water. Default is "blended_water".

Value

blend_waters returns a water class object with blended water quality parameters.

blend_waters_df returns a data frame with a water class column containing blended water quality

See Also

define_water

Examples

```
water1 <- define_water(7, 20, 50)
water2 <- define_water(7.5, 20, 100, tot_nh3 = 2)
blend_waters(c(water1, water2), c(.4, .6))


example_df <- water_df %>%
  dplyr::slice_head(n = 3) %>%
  define_water_df() %>%
  chemdose_ph_df(naoh = 22) %>%
  dplyr::mutate(
    ratios1 = .4,
    ratios2 = .6
  ) %>%
  blend_waters_df(
    waters = c("defined", "dosed_chem"),
    ratios = c("ratios1", "ratios2"), output_water = "Blending_after_chemicals"
  )


waterA <- define_water(7, 20, 100, tds = 100)
example_df <- water_df %>%
  dplyr::slice_head(n = 3) %>%
  define_water_df() %>%
  blend_waters_df(waters = c("defined", waterA), ratios = c(.8, .2))
```

`bromatecoeffs` *Data frame of bromate coefficients for predicting bromate formation during ozonation*

Description

A dataset containing coefficients for calculating ozone formation

Usage

```
bromatecoeffs
```

Format

A dataframe with 30 rows and 10 columns

model First author of source model

ammonia Either T or F, depending on whether the model applies to waters with ammonia present.

A First coefficient in bromate model

a Exponent in bromate model, associated with Br-

b Exponent in bromate model, associated with DOC

c Exponent in bromate model, associated with UVA

d Exponent in bromate model, associated with pH

e Exponent in bromate model, associated with Alkalinity

f Exponent in bromate model, associated with ozone dose

g Exponent in bromate model, associated with reaction time

h Exponent in bromate model, associated with ammonia (NH4+)

i Exponent in bromate model, associated with temperature

I Coefficient in bromate model, associated with temperature in the exponent. Either i or I are used, not both.

Source

Ozekin (1994), Sohn et al (2004), Song et al (1996), Galey et al (1997), Siddiqui et al (1994)

See references list at: https://github.com/BrownandCaldwell-Public/tidywater/wiki/References

calculate_activity *Calculate activity coefficients*

Description

This function calculates activity coefficients at a given temperature based on equation 5-43 from Davies (1967), Crittenden et al. (2012)

Usage

```
calculate_activity(z, is, temp)
```

Arguments

z	Charge of ions in the solution
is	Ionic strength of the solution
temp	Temperature of the solution in Celsius

Value

A numeric value for the activity coefficient.

Examples

```
calculate_activity(2, 0.1, 25)
```

calculate_corrosion *Calculate six corrosion and scaling indices (AI, RI, LSI, LI, CSMR, CCPP)*

Description

This function takes an object created by define_water and calculates corrosion and scaling indices. For a single water, use `calculate_corrosion`; to apply the calculations to a dataframe, use `calculate_corrosion_df`.

Usage

```
calculate_corrosion(
  water,
  index = c("aggressive", "ryznar", "langelier", "ccpp", "larsonskold", "csmr"),
  form = "calcite"
)
```

```
calculate_corrosion_df(
  df,
  input_water = "defined",
  water_prefix = TRUE,
  index = c("aggressive", "ryznar", "langelier", "ccpp", "larsonskold", "csmr"),
  form = "calcite"
)
```

Arguments

water	Source water of class "water" created by define_water
index	The indices to be calculated. Default calculates all six indices: "aggressive", "ryznar", "langelier", "ccpp", "larsonskold", "csmr" CCPP may not be able to be calculated sometimes, so it may be advantageous to leave this out of the function to avoid errors
form	Form of calcium carbonate mineral to use for modelling solubility in the Langelier and Ryznar indices: "calcite" (default), "aragonite", or "vaterite". CCPP always uses calcite.
df	a data frame containing a water class column, created using define_water
input_water	name of the column of water class data to be used as the input. Default is "defined".
water_prefix	append water name to beginning of output columns. Defaults to TRUE

Details

Aggressiveness Index (AI), unitless - the corrosive tendency of water and its effect on asbestos cement pipe.

Ryznar Index (RI), unitless - a measure of scaling potential.

Langelier Saturation Index (LSI), unitless - describes the potential for calcium carbonate scale formation. Equations use empirical calcium carbonate solubilities from Plummer and Busenberg (1982) and Crittenden et al. (2012) rather than calculated from the concentrations of calcium and carbonate in the water.

Larson-skold Index (LI), unitless - describes the corrosivity towards mild steel.

Chloride-to-sulfate mass ratio (CSMR), mg Cl/mg SO4 - indicator of galvanic corrosion for lead solder pipe joints.

Calcium carbonate precipitation potential (CCPP), mg/L as CaCO3 - a prediction of the mass of calcium carbonate that will precipitate at equilibrium. A positive CCPP value indicates the amount of CaCO3 (mg/L as CaCO3) that will precipitate. A negative CCPP indicates how much CaCO3 can be dissolved in the water. CCPP is always calculated against calcite solubility, following Merrill and Sanks (1977a, 1977b, 1978) and Trussell (1998); the `form` argument does not affect CCPP.

If chloride or sulfate is missing, the Larson-Skold index and CSMR are returned as `NA` with a warning, and the other requested indices are still calculated.

Value

`calculate_corrosion` returns a data frame with corrosion and scaling indices as individual columns.

`calculate_corrosion_df` returns the a data frame containing specified corrosion and scaling indices as columns.

Source

AWWA (1977)

Crittenden et al. (2012)

Langelier (1936)

Larson and Skold (1958)

Merrill and Sanks (1977a)

Merrill and Sanks (1977b)

Merrill and Sanks (1978)

Nguyen et al. (2011)

Plummer and Busenberg (1982)

Ryznar (1944)

Schock (1984)

Trussell (1998)

U.S. EPA (1980)

See reference list at https://github.com/BrownandCaldwell-Public/tidywater/wiki/References

Examples

```
water <- define_water(
  ph = 8, temp = 25, alk = 200, tot_hard = 200,
  tds = 576, cl = 150, so4 = 200
)
corrosion_indices <- calculate_corrosion(water)

water <- define_water(ph = 8, temp = 25, alk = 100, tot_hard = 50, tds = 200)
corrosion_indices <- calculate_corrosion(water, index = c("aggressive", "ccpp"))


example_df <- water_df %>%
  define_water_df() %>%
  calculate_corrosion_df(index = c("aggressive", "ccpp"))
```

calculate_hardness *Calculate hardness from calcium and magnesium*

Description

This function takes Ca and Mg in mg/L and returns hardness in mg/L as CaCO3

Usage

```
calculate_hardness(ca, mg, type = "total", startunit = "mg/L")
```

Arguments

ca	Calcium concentration in mg/L as Ca
mg	Magnesium concentration in mg/L as Mg
type	"total" returns total hardness, "ca" returns calcium hardness. Defaults to "total"
startunit	Units of Ca and Mg. Defaults to mg/L

Value

A numeric value for the total hardness in mg/L as CaCO3.

Examples

```
calculate_hardness(50, 10)

water_defined <- define_water(7, 20, 50, 100, 80, 10, 10, 10, 10, tot_po4 = 1)
calculate_hardness(water_defined@ca, water_defined@mg, "total", "M")
```

check_water *check a water object for common issues and applicable models*

Description

This function takes a previously defined water class input and checks common mismatch issues with water quality. It returns an unnamed list with the check, the value, and a flag if there is an issue. It will also check to see what other tidywater models are applicable to the input water unless the user specifies to skip that step.

Usage

```
check_water(water, skip_models = FALSE)
```

Arguments

water	a water class object
skip_models	a boolean to specify whether to skip the check for applicable models, defaults to FALSE

Details

This function serves as a heuristic check for users to check common issues with water data.

Value

an unnamed list with the check, value, and flag for any issues found. If skip_models = FALSE, also includes a check for applicable models with tidywater functions.

Examples

```
water1<-define_water(ph=7, alk = 70)
check_water(water1)

water2<-define_water( ph=7, alk = 70, ca = 20, na = 20, cl = 20, so4 = 20)
check_water(water2, skip_models = TRUE)
```

---

chemdosetoccoeff *Data frame of model coefficients for coagulation TOC removal*

Description

A dataset containing coefficients from the Edwards (1997) model for coagulation TOC removal, or Wilder & Korak (2025) model for reuse waters.

Usage

```
chemdosetoccoeff
```

Format

A dataframe with 9 rows and 11 columns. Columns not used by a model are NA:

ID Coefficient type

x3 x3 parameter

x2 x2 parameter

x1 x1 parameter

k1 k1 parameter

k2 k2 parameter

b b parameter

fns Fraction of non-sorbing DOC

z0 z0 parameter

z1 z1 parameter

z2 z2 parameter

Source

Edwards (1997) Table 2. Wilder & Korak (2025) Table 1.

See references list at: https://github.com/BrownandCaldwell-Public/tidywater/wiki/References

---

`chemdose_chloramine` *Calculate chlorine and chloramine Concentrations with the breakpoint cblorination approach*

Description

chemdose_chloramine, adopted from the U.S. EPA's Chlorine Breakpoint Curve Simulator, calculates chlorine and chloramine concentrations based on the two papers Jafvert & Valentine (Environ. Sci. Technol., 1992, 26 (3), pp 577-586) and Vikesland et al. (Water Res., 2001, 35 (7), pp 1766-1776). Required arguments include an object of class "water" created by define_water, chlorine dose, and reaction time. The function also requires additional water quality parameters defined in define_water including temperature, pH, and alkalinity. For a single water use `chemdose_chloramine`; for a dataframe use `chemdose_chloramine_df`. Use `pluck_cols = TRUE` to get values from the output water as new dataframe columns. For most arguments in the `_df` helper "use_col" default looks for a column of the same name in the dataframe. The argument can be specified directly in the function instead or an unquoted column name can be provided.

Usage

```
chemdose_chloramine(
  water,
  time,
  cl2 = 0,
  nh3 = 0,
  use_free_cl_slot = FALSE,
  use_tot_nh3_slot = FALSE
)

chemdose_chloramine_df(
  df,
  input_water = "defined",
  output_water = "chloraminated",
  pluck_cols = FALSE,
  water_prefix = TRUE,
  keep_dose_cols = FALSE,
```

```
  time = "use_col",
  cl2 = "use_col",
  nh3 = "use_col",
  use_free_cl_slot = "use_col",
  use_tot_nh3_slot = "use_col"
)
```

Arguments

water	Source water object of class "water" created by define_water
time	Reaction time (minutes). Time defined needs to be greater or equal to 1 minute.
cl2	Applied chlorine dose (mg/L as Cl2), defaults to 0.If not specified, use free_chlorine slot in water.
nh3	Applied ammonia dose (mg/L as N), defaults to 0. If not specified, use tot_nh3 slot in water.
use_free_cl_slot	Defaults to FALSE. If TRUE, uses free_chlorine slot in water. If TRUE AND there is a cl2 input, both the free_chlorine water slot and chlorine dose will be used.
use_tot_nh3_slot	Defaults to FALSE. If TRUE, uses tot_nh3 slot in water. If TRUE AND there is a nh3 input, both the tot_nh3 water slot and ammonia dose will be used.
df	a data frame containing a water class column, which has already been computed using define_water_df. The df may include a column named for the applied chlorine dose (cl2_dose), and a column for time in hours.
input_water	name of the column of water class data to be used as the input for this function. Default is "defined".
output_water	name of the output column storing updated water class object. Default is "chloraminated".
pluck_cols	Extract water slots modified by the function ("free_chlorine", "nh2cl", "nhcl2", "ncl3", "combined_chlorine", "tot_nh3") into new numeric columns for easy access. Default to FALSE.
water_prefix	Append the output_water name to the start of the plucked columns. Default is TRUE.
keep_dose_cols	Option to keep the original chemical dose columns in the output data frame. Default is FALSE.

Value

chemdose_chloramine returns a water class object with predicted chlorine and chloramine concentrations.

chemdose_chloramine_df returns a data frame containing water class column with updated chlorine/chloramine slots: free_chlorine, nh2cl, nhcl2, ncl3, combined_chlorine, tot_nh3. Optionally, it also adds columns for each of those slots individually.

Source

See references list at: https://github.com/BrownandCaldwell-Public/tidywater/wiki/References

Examples

```
breakpoint <- define_water(7.5, 20, 65, free_chlorine = 5, tot_nh3 = 1) %>%
  chemdose_chloramine(time = 40, cl2 = 2, nh3 = 1, use_free_cl_slot = TRUE)


breakpoint <- water_df %>%
  dplyr::mutate(free_chlorine = 5, tot_nh3 = 1) %>%
  define_water_df() %>%
  dplyr::mutate(
    time = 8,
    cl2dose = rep(c(2, 3, 4), 4)
  ) %>%
  chemdose_chloramine_df(
    output_water = "final",
    cl2 = cl2dose,
    use_free_cl_slot = TRUE,
    use_tot_nh3_slot = TRUE,
    pluck_cols = TRUE
  )
```

---

chemdose_chlordecay *Calculate chlorine decay*

Description

calculates the decay of chlorine or chloramine based on the U.S. EPA's Water Treatment Plant Model (U.S. EPA, 2001). For a single water use `chemdose_chlordecay`; for a dataframe use `chemdose_chlordecay_df`. Use `pluck_cols = TRUE` to get values from the output water as new dataframe columns. For most arguments in the `_df` helper "use_col" default looks for a column of the same name in the dataframe. The argument can be specified directly in the function instead or an unquoted column name can be provided.

Usage

```
chemdose_chlordecay(
  water,
  cl2_dose,
  time,
  treatment = "raw",
  cl_type = "chlorine",
  use_chlorine_slot = FALSE
)
```

```
chemdose_chlordecay_df(
  df,
  input_water = "defined",
  output_water = "disinfected",
  pluck_cols = FALSE,
  water_prefix = TRUE,
  cl2_dose = "use_col",
  time = "use_col",
  treatment = "use_col",
  cl_type = "use_col",
  use_chlorine_slot = "use_col"
)
```

Arguments

water	Source water object of class "water" created by define_water
cl2_dose	Applied chlorine or chloramine dose (mg/L as cl2). Model results are valid for doses between 0.995 and 41.7 mg/L for raw water, and for doses between 1.11 and 24.7 mg/L for coagulated water.
time	Reaction time (hours). Chlorine decay model results are valid for reaction times between 0.25 and 120 hours.Chloramine decay model does not have specified boundary conditions.
treatment	Type of treatment applied to the water. Options include "raw" for no treatment (default), "coag" for water that has been coagulated or softened.
cl_type	Type of chlorination applied, either "chlorine" (default) or "chloramine".
use_chlorine_slot	Defaults to FALSE. When TRUE, uses either free_chlorine or combined_chlorine slot in water (depending on cl_type). If 'cl2_dose' argument, not specified, chlorine slot will be used. If 'cl2_dose' specified and use_chlorine_slot is TRUE, all chlorine will be summed.
df	a data frame containing a water class column, which has already been computed using define_water_df. The df may include a column named for the applied chlorine dose (cl2), and a column for time in hours.
input_water	name of the column of water class data to be used as the input for this function. Default is "defined".
output_water	name of the output column storing updated water class object. Default is "disinfected".
pluck_cols	Extract water slots modified by the function (free_chlorine, combined_chlorine) into new numeric columns for easy access. Default to FALSE.
water_prefix	Append the output_water name to the start of the plucked columns. Default is TRUE.

Details

Required arguments include an object of class "water" created by define_water, applied chlorine/chloramine dose, type, reaction time, and treatment applied (options include "raw" for no

treatment, or "coag" for coagulated water). The function also requires additional water quality parameters defined in define_water including TOC and UV254. The output is a new "water" class with the calculated total chlorine value stored in the 'free_chlorine' or 'combined_chlorine' slot, depending on what type of chlorine is dosed. When modeling residual concentrations through a unit process, the U.S. EPA Water Treatment Plant Model applies a correction factor based on the influent and effluent residual concentrations (see U.S. EPA (2001) equation 5-118) that may need to be applied manually by the user based on the output.

Value

`chemdose_chlordecay` returns an updated disinfectant residual in the free_chlorine or combined_chlorine water slot in units of M. Use convert_units to convert to mg/L.

`chemdose_chlordecay_df` returns a data frame containing a water class column with updated free_chlorine or combined_chlorine residuals. Optionally, it also adds columns for each of those slots individually.

Source

U.S. EPA (2001)

See references list at: https://github.com/BrownandCaldwell-Public/tidywater/wiki/References

Examples

```
example_cl2 <- define_water(8, 20, 66, toc = 4, uv254 = 0.2) %>%
  chemdose_chlordecay(cl2_dose = 2, time = 8)

example_cl2 <- define_water(8, 20, 66, toc = 4, uv254 = 0.2, free_chlorine = 3) %>%
  chemdose_chlordecay(cl2_dose = 2, time = 8, use_chlorine_slot = TRUE)



example_df <- water_df %>%
  dplyr::mutate(br = 50) %>%
  define_water_df() %>%
  chemdose_chlordecay_df(input_water = "defined", cl2_dose = 4, time = 8)

example_df <- water_df %>%
  dplyr::mutate(
    br = 50,
    free_chlorine = 2
  ) %>%
  define_water_df() %>%
  dplyr::mutate(
    cl2_dose = seq(2, 24, 2),
    ClTime = 30
  ) %>%
  chemdose_chlordecay_df(
    time = ClTime,
    use_chlorine_slot = TRUE,
    treatment = "coag",
    cl_type = "chloramine",
```

```
  pluck_cols = TRUE
)
```

---

chemdose_dbp *Calculate DBP formation*

Description

Calculates disinfection byproduct (DBP) formation based on the U.S. EPA's Water Treatment Plant Model (U.S. EPA, 2001). Required arguments include an object of class "water" created by define_water chlorine dose, type, reaction time, and treatment applied (if any). The function also requires additional water quality parameters defined in define_water including bromide, TOC, UV254, temperature, and pH.

For a single water use `chemdose_dbp`; for a dataframe use `chemdose_dbp_df`. Use `pluck_cols = TRUE` to get values from the output water as new dataframe columns. For most arguments in the `_df` helper "use_col" default looks for a column of the same name in the dataframe. The argument can be specified directly in the function instead or an unquoted column name can be provided.

Usage

```
chemdose_dbp(
  water,
  cl2,
  time,
  treatment = "raw",
  cl_type = "chorine",
  location = "plant",
  correction = TRUE,
  coeff = NULL
)

chemdose_dbp_df(
  df,
  input_water = "defined",
  output_water = "disinfected",
  pluck_cols = FALSE,
  water_prefix = TRUE,
  keep_dose_cols = FALSE,
  cl2 = "use_col",
  time = "use_col",
  treatment = "use_col",
  cl_type = "use_col",
  location = "use_col",
  correction = TRUE,
  coeff = NULL
)
```

Arguments

water	Source water object of class "water" created by define_water
cl2	Applied chlorine dose (mg/L as Cl2). Model results are valid for doses between 1.51 and 33.55 mg/L.
time	Reaction time (hours). Model results are valid for reaction times between 2 and 168 hours.
treatment	Type of treatment applied to the water. Options include "raw" for no treatment (default), "coag" for water that has been coagulated or softened, and "gac" for water that has been treated by granular activated carbon (GAC). GAC treatment has also been used for estimating formation after membrane treatment with good results.
cl_type	Type of chlorination applied, either "chlorine" (default) or "chloramine".
location	Location for DBP formation, either in the "plant" (default), or in the distributions system, "ds".
correction	Model calculations are adjusted based on location and cl_type. Default value is TRUE.
coeff	Optional input to specify custom coefficients to the dbp model. Must be a data frame with the following columns: ID, and the corresponding coefficients A, a, b, c, d, e, f, and ph_const for each dbp of interest. Default value is NULL.
df	a data frame containing a water class column, which has already been computed using define_water. The df may include columns for the other function arguments.
input_water	name of the column of water class data to be used as the input for this function. Default is "defined".
output_water	name of the output column storing updated water class object. Default is "disinfected".
pluck_cols	Extract primary water slots modified by the function (tthm, haa5) into new numeric columns for easy access with TRUE. Alternatively, specify "all" to get all DBP species in addition: (tthm, chcl3, chcl2br, chbr2cl, chbr3, haa5, mcaa, dcaa, tcaa, mbaa, dbaa) Defaults to FALSE.
water_prefix	Append the output_water name to the start of the plucked columns. Default is TRUE.
keep_dose_cols	Option to keep the original chemical dose columns in the output data frame. Default is FALSE.

Details

The function will calculate haloacetic acids (HAA) as HAA5, and total trihalomethanes (TTHM). Use `summarize_wq(water, params = c("dbps"))` to quickly tabulate the results.

Value

`chemdose_dbp` returns a single water class object with predicted DBP concentrations.

`chemdose_dbp_df` returns a data frame containing a water class column with updated tthm, chcl3, chcl2br, chbr2cl, chbr3, haa5, mcaa, dcaa, tcaa, mbaa, dbaa concentrations. Optionally, it also adds columns for those slots individually.

Source

TTHMs, raw: U.S. EPA (2001) equation 5-131

HAAs, raw: U.S. EPA (2001) equation 5-134

TTHMs, treated: U.S. EPA (2001) equation 5-139

HAAs, treated: U.S. EPA (2001) equation 5-142

See references list at: https://github.com/BrownandCaldwell-Public/tidywater/wiki/References

Examples

```
example_dbp <- define_water(8, 20, 66, toc = 4, uv254 = .2, br = 50) %>%
  chemdose_dbp(cl2 = 2, time = 8)
example_dbp <- define_water(7.5, 20, 66, toc = 4, uv254 = .2, br = 50) %>%
  chemdose_dbp(cl2 = 3, time = 168, treatment = "coag", location = "ds")


example_df <- water_df %>%
  dplyr::mutate(br = 50) %>%
  define_water_df() %>%
  chemdose_dbp_df(input_water = "defined", cl2 = 4, time = 8)

example_df <- water_df %>%
  dplyr::mutate(br = 50) %>%
  dplyr::slice_sample(n = 3) %>%
  define_water_df() %>%
  dplyr::mutate(
    cl2_dose = c(2, 3, 4),
    time = 30
  ) %>%
  chemdose_dbp_df(
    cl2 = cl2_dose, treatment = "coag", location = "ds",
    cl_type = "chloramine", pluck_cols = TRUE
  )
```

---

chemdose_inorgdemand *Inorganic Oxidant Demand*

Description

Calculates Cl residual from inorganic scavengers present in the input water object: Mg, Fe, and Ammonia. The output water will have updated `free_chlorine` slot to reflect the chlorine residual after accounting for the inorganic demand. The `mn`, `fe`, and `tot_nh3` slots will be set to 0, as they are assumed to be fully oxidized by the chlorine dose. Other products of the oxidation reactions will not be tracked or updated in the output water object. For a single water use `chemdose_inorgdemand`; to apply the model to a dataframe use `chemdose_inorgdemand_df`.

By default (use_chlorine_slot = FALSE), the output free_chlorine is cl2_dose minus the inorganic demand, and any free chlorine already in the water is overwritten (with a warning). Set use_chlorine_slot = TRUE to add the existing free_chlorine slot to cl2_dose before the demand is subtracted. If cl2_dose is not specified, the existing free_chlorine slot is used (with a warning when use_chlorine_slot = FALSE). The function errors if the inorganic demand is greater than the available chlorine.

The inorganic demand of ammonia is calculated assuming all ammonia is converted to nitrogen gas, not considering the conversion of ammonia to nitrate. For chloramine chemistry, use chemdose_chloramine if interested in more precise breakpoint chlorination calculations.

Usage

```
chemdose_inorgdemand(water, cl2_dose, use_chlorine_slot = FALSE)

chemdose_inorgdemand_df(
  df,
  input_water = "defined",
  output_water = "oxidized",
  pluck_cols = FALSE,
  water_prefix = TRUE,
  keep_dose_cols = FALSE,
  cl2_dose = "use_col",
  use_chlorine_slot = "use_col"
)
```

Arguments

water	Input water object with mn, fe, and tot_nh3 slots defined. The free_chlorine slot must also be defined when the slot is used (use_chlorine_slot = TRUE or cl2_dose not specified).
cl2_dose	Chlorine dose in mg/L Cl2.
use_chlorine_slot	Defaults to FALSE. When TRUE, the existing free_chlorine slot in the water is added to cl2_dose. If cl2_dose is not specified, only the chlorine slot will be used, regardless of use_chlorine_slot.
df	a data frame containing a water class column, which has already been computed using define_water_df. The df may include columns for the chlorine dose (cl2_dose) and use_chlorine_slot.
input_water	name of the column of water class data to be used as the input for this function. Default is "defined".
output_water	name of the output column storing updated water class object. Default is "oxidized".
pluck_cols	Extract the free_chlorine slot into a new numeric column for easy access. Default to FALSE.
water_prefix	Append the output_water name to the start of the plucked columns. Default is TRUE.

keep_dose_cols Option to keep the original chemical dose columns in the output data frame. Default is FALSE.

Value

chemdose_inorgdemand returns a water class object with updated free_chlorine slot that reflects the chlorine residual.

chemdose_inorgdemand_df returns a data frame containing a water class column with updated free_chlorine, mn, fe, and tot_nh3 slots. Optionally, it also adds a column for free_chlorine.

Examples

```
water1<-define_water(ph = 7, alk = 70, mn = 1, fe = 2, tot_nh3 = 1)
cl_residual <- chemdose_inorgdemand(water1, 20)

# Use the existing free chlorine in the water
water2 <- define_water(ph = 7, alk = 70, mn = 1, fe = 2, tot_nh3 = 1, free_chlorine = 12)
cl_residual <- chemdose_inorgdemand(water2, use_chlorine_slot = TRUE)


example_df <- water_df %>%
  dplyr::mutate(mn = 20, fe = 0.5, tot_nh3 = 0.2) %>%
  define_water_df() %>%
  chemdose_inorgdemand_df(cl2_dose = 5, pluck_cols = TRUE)

example_df <- water_df %>%
  dplyr::mutate(mn = 20, fe = 0.5, tot_nh3 = 0.2) %>%
  define_water_df() %>%
  dplyr::mutate(ClDose = seq(3, 14, 1)) %>%
  chemdose_inorgdemand_df(cl2_dose = ClDose, output_water = "chlorinated")

# Use the free_chlorine already in each water instead of a dose
example_df <- water_df %>%
  dplyr::mutate(mn = 20, fe = 0.5, tot_nh3 = 0.2, free_chlorine = 4) %>%
  define_water_df() %>%
  chemdose_inorgdemand_df(use_chlorine_slot = TRUE, pluck_cols = TRUE)
```

---

chemdose_ph *Calculate new pH and ion balance after chemical addition*

Description

Calculates the new pH, alkalinity, and ion balance of a water based on different chemical additions. For a single water use chemdose_ph; for a dataframe use chemdose_ph_df. Use pluck_cols = TRUE to get values from the output water as new dataframe columns. For most arguments in the _df helper "use_col" default looks for a column of the same name in the dataframe. The argument can be specified directly in the function instead or an unquoted column name can be provided.

Usage

```
chemdose_ph(
  water,
  hcl = 0,
  h2so4 = 0,
  h3po4 = 0,
  hno3 = 0,
  ch3cooh = 0,
  citric = 0,
  co2 = 0,
  naoh = 0,
  caoh2 = 0,
  mgoh2 = 0,
  na2co3 = 0,
  nahco3 = 0,
  caco3 = 0,
  caso4 = 0,
  caocl2 = 0,
  cacl2 = 0,
  cl2 = 0,
  naocl = 0,
  nh4oh = 0,
  nh42so4 = 0,
  alum = 0,
  ferricchloride = 0,
  ferricsulfate = 0,
  ach = 0,
  pacl = 0,
  kmno4 = 0,
  naf = 0,
  na3po4 = 0,
  softening_correction = FALSE
)

chemdose_ph_df(
  df,
  input_water = "defined",
  output_water = "dosed_chem",
  na_to_zero = TRUE,
  pluck_cols = FALSE,
  water_prefix = TRUE,
  keep_dose_cols = FALSE,
  hcl = "use_col",
  h2so4 = "use_col",
  h3po4 = "use_col",
  hno3 = "use_col",
  ch3cooh = "use_col",
  co2 = "use_col",
```

```
  naoh = "use_col",
  na2co3 = "use_col",
  nahco3 = "use_col",
  caoh2 = "use_col",
  mgoh2 = "use_col",
  caocl2 = "use_col",
  cacl2 = "use_col",
  cl2 = "use_col",
  naocl = "use_col",
  nh4oh = "use_col",
  nh42so4 = "use_col",
  caco3 = "use_col",
  caso4 = "use_col",
  alum = "use_col",
  ferricchloride = "use_col",
  ferricsulfate = "use_col",
  ach = "use_col",
  pacl = "use_col",
  kmno4 = "use_col",
  naf = "use_col",
  na3po4 = "use_col",
  softening_correction = "use_col"
)
```

Arguments

water	Source water object of class "water" created by define_water
hcl	Amount of hydrochloric acid added in mg/L: HCl -> H + Cl
h2so4	Amount of sulfuric acid added in mg/L: H2SO4 -> 2H + SO4
h3po4	Amount of phosphoric acid added in mg/L: H3PO4 -> 3H + PO4
hno3	Amount of nitric acid added in mg/L: HNO3 -> H + NO3
ch3cooh	Amount of acetic acid added in mg/L: CH3COOH -> H + CH3COO
citric	Amount of citric acid added in mg/L: C3H4OH(COOH)3 -> 3H+ C3H4OH(COO)3
co2	Amount of carbon dioxide added in mg/L: CO2 (gas) + H2O -> H2CO3*
naoh	Amount of caustic added in mg/L: NaOH -> Na + OH
caoh2	Amount of lime added in mg/L: Ca(OH)2 -> Ca + 2OH
mgoh2	Amount of magnesium hydroxide added in mg/L: Mg(OH)2 -> Mg + 2OH
na2co3	Amount of soda ash added in mg/L: Na2CO3 -> 2Na + CO3
nahco3	Amount of sodium bicarbonate added in mg/L: NaHCO3 -> Na + H + CO3
caco3	Amount of calcium carbonate added (or removed) in mg/L: CaCO3 -> Ca + CO3
caso4	Amount of calcium sulfate added (for post-RO condition) in mg/L: CaSO4 -> Ca + SO4
caocl2	Amount of Calcium hypochlorite added in mg/L as Cl2: CaOCl2 -> Ca + 2OCl

cacl2	Amount of calcium chloride added in mg/L: CaCl2 -> Ca2+ + 2Cl-
cl2	Amount of chlorine gas added in mg/L as Cl2: Cl2(g) + H2O -> HOCl + H + Cl
naocl	Amount of sodium hypochlorite added in mg/L as Cl2: NaOCl -> Na + OCl
nh4oh	Amount of ammonium hydroxide added in mg/L as N: NH4OH -> NH4 + OH
nh42so4	Amount of ammonium sulfate added in mg/L as N: (NH4)2SO4 -> 2NH4 + SO4
alum	Amount of hydrated aluminum sulfate added in mg/L: Al2(SO4)3*14H2O + 6HCO3 -> 2Al(OH)3(am) +3SO4 + 14H2O + 6CO2
ferricchloride	Amount of ferric Chloride added in mg/L: FeCl3 + 3HCO3 -> Fe(OH)3(am) + 3Cl + 3CO2
ferricsulfate	Amount of ferric sulfate added in mg/L: Fe2(SO4)3*8.8H2O + 6HCO3 -> 2Fe(OH)3(am) + 3SO4 + 8.8H2O + 6CO2
ach	Amount of aluminum chlorohydrate added in mg/L: Al2(OH)5Cl*2H2O + HCO3 -> 2Al(OH)3(am) + Cl + 2H2O + CO2
pacl	Amount of polyaluminum chloride added in mg/L as Al2O3 (assumed Cl:Al ratio = 0.9): Al2(OH)4.2Cl(1.8) + #HCO3 -> 2Al(OH)3(am) + 1.8Cl + #H2O + #CO2....
kmno4	Amount of potassium permanganate added in mg/L: KMnO4 -> K + MnO4
naf	Amount of sodium fluoride added in mg/L: NaF -> Na + F
na3po4	Amount of trisodium phosphate added in mg/L: Na3PO4 -> 3Na + PO4
softening_correction	Set to TRUE to correct post-softening pH (caco3 must be < 0). Default is FALSE. Based on WTP model equation 5-62
df	a data frame containing a water class column, which has already been computed using define_water_df The df may include columns named for the chemical(s) being dosed.
input_water	name of the column of water class data to be used as the input for this function. Default is "defined".
output_water	name of the output column storing updated water class object. Default is "dosed".
na_to_zero	option to convert all NA values in the data frame to zeros. Default value is TRUE.
pluck_cols	Extract primary water slots modified by the function (ph, alk) into new numeric columns for easy access. Default to FALSE.
water_prefix	Append the output_water name to the start of the plucked columns. Default is TRUE.
keep_dose_cols	Option to keep the original chemical dose columns in the output data frame. Default is FALSE.

Details

The function takes an object of class "water" created by define_water and user-specified chemical additions and returns a new object of class "water" with updated water quality. Units of all chemical additions are in mg/L as chemical (not as product).

chemdose_ph works by evaluating all the user-specified chemical additions and solving for what the new pH must be using uniroot to satisfy the principle of electroneutrality in pure water while correcting for the existing alkalinity of the water that the chemical is added to. Multiple chemicals can be added simultaneously or each addition can be modeled independently through sequential doses.

Value

chemdose_ph returns a water class object with updated pH, alkalinity, and ions post-chemical addition.

chemdose_ph_df returns a data frame containing a water class column with updated pH, alkalinity, and ions post-chemical addition.

See Also

define_water, convert_units

Examples

```
water <- define_water(ph = 7, temp = 25, alk = 10)
# Dose 1 mg/L of hydrochloric acid
dosed_water <- chemdose_ph(water, hcl = 1)

# Dose 1 mg/L of hydrochloric acid and 5 mg/L of alum simultaneously
dosed_water <- chemdose_ph(water, hcl = 1, alum = 5)

# Softening:
water2 <- define_water(ph = 7, temp = 25, alk = 100, tot_hard = 350)
dosed_water2 <- chemdose_ph(water2, caco3 = -100, softening_correction = TRUE)


example_df <- water_df %>%
  define_water_df() %>%
  dplyr::slice_head(n = 3) %>%
  dplyr::mutate(
    hcl = c(2, 4, 6),
    Caustic = 20
  ) %>%
  chemdose_ph_df(input_water = "defined", mgoh2 = c(20, 55), co2 = 4, naoh = Caustic)

example_df <- water_df %>%
  define_water_df() %>%
  chemdose_ph_df(naoh = 5, pluck_cols = TRUE)
```

chemdose_toc *Determine TOC removal from coagulation*

Description

This function applies the Edwards (1997) model to a water created by define_water to determine coagulated DOC. Reuse waters with effluent organic matter (EfOM) can be modeled by selecting one of the Wilder & Korak (2025) models with `coeff`. Reuse models apply only to aluminum coagulants (alum, ACH, PACl). Model assumes all particulate TOC is removed; therefore TOC = DOC in output. Coagulated UVA is from U.S. EPA (2001) equation 5-80 for Edwards (1997) model and a regression for the Wilder & Korak (2025) model. Note that the models rely on pH of coagulation. If only raw water pH is known, utilize chemdose_ph first. For a single water use `chemdose_toc`; for a dataframe use `chemdose_toc_df`. Use `pluck_cols = TRUE` to get values from the output water as new dataframe columns. For most arguments in the `_df` helper "use_col" default looks for a column of the same name in the dataframe. The argument can be specified directly in the function instead or an unquoted column name can be provided.

Usage

```
chemdose_toc(
  water,
  alum = 0,
  ferricchloride = 0,
  ferricsulfate = 0,
  ach = 0,
  pacl = 0,
  coeff = "Alum",
  caoh2 = 0,
  fns = NA_real_
)

chemdose_toc_df(
  df,
  input_water = "defined",
  output_water = "coagulated",
  pluck_cols = FALSE,
  water_prefix = TRUE,
  keep_dose_cols = FALSE,
  alum = "use_col",
  ferricchloride = "use_col",
  ferricsulfate = "use_col",
  ach = "use_col",
  pacl = "use_col",
  caoh2 = "use_col",
  coeff = "use_col",
  fns = "use_col"
)
```

Arguments

water	Source water object of class "water" created by define_water. Water must include ph, doc, and uv254
alum	Amount of hydrated aluminum sulfate added in mg/L: Al2(SO4)3*14H2O + 6HCO3 -> 2Al(OH)3(am) +3SO4 + 14H2O + 6CO2
ferricchloride	Amount of ferric chloride added in mg/L: FeCl3 + 3HCO3 -> Fe(OH)3(am) + 3Cl + 3CO2
ferricsulfate	Amount of ferric sulfate added in mg/L: Fe2(SO4)3*8.8H2O + 6HCO3 -> 2Fe(OH)3(am) + 3SO4 + 8.8H2O + 6CO2
ach	Amount of aluminum chlorohydrate added in mg/L: Al2(OH)5Cl*2H2O + HCO3 -> 2Al(OH)3(am) + Cl + 2H2O + CO2
pacl	Amount of polyaluminum chloride added in mg/L as Al2O3 (assumed Cl:Al ratio = 0.9): Al2(OH)4.2Cl(1.8) + #HCO3 -> 2Al(OH)3(am) + 1.8Cl + #H2O + #CO2....
coeff	String specifying the Edwards coefficients to be used from "Alum", "Ferric", "General Alum", "General Ferric", or "Low DOC" or data frame of coefficients, which must include: k1, k2, x1, x2, x3, b. For reuse waters (aluminum coagulants only), Wilder & Korak string specifying coefficients can be used from "Reuse General Langmuir", "Reuse Site Langmuir", "Reuse General Linear", "Reuse Site Linear", or a dataframe of coefficients that must include: fns, x1, x2, x3, b for Langmuir or fns, z0, z1, z2 for Linear. Reuse models were validated for coagulation pH 5.5 - 8 and 20 - 200 mg/L alum. See reference list for more information.
caoh2	Option to add caoh2 in mg/L to soften the water. Will predict DOC, TOC, UV254 using a modified equation (see reference list). Defaults to zero.
fns	Fraction of non-sorbing DOC (0 - 1), required for the site-specific reuse models ("Reuse Site Langmuir" and "Reuse Site Linear"). Measured as the fraction of DOC remaining after coagulation with 200 mg/L alum at pH 6 (Wilder & Korak 2025). The general reuse models use fixed values of 0.649 (Langmuir) and 0.650 (Linear); a supplied fns is ignored with a warning. Not used by Edwards models.
df	a data frame containing a water class column, which has already been computed using define_water_df. The df may include a column named for the coagulant being dosed, and a column named for the set of coefficients to use.
input_water	name of the column of water class data to be used as the input for this function. Default is "defined".
output_water	name of the output column storing updated water class object. Default is "coagulated".
pluck_cols	Extract water slots modified by the function (doc, toc, uv254) into new numeric columns for easy access. Default to FALSE.
water_prefix	Append the output_water name to the start of the plucked columns. Default is TRUE.
keep_dose_cols	Option to keep the original chemical dose columns in the output data frame. Default is FALSE.

Value

chemdose_toc returns a single water class object with an updated DOC, TOC, and UV254 concentration.

chemdose_toc_df returns a data frame containing a water class column with updated DOC, TOC, and UV254 concentrations. Optionally, it also adds columns for each of those slots individually.

Source

Edwards (1997)

Wilder & Korak (2025)

U.S. EPA (2001)

See reference list at: https://github.com/BrownandCaldwell-Public/tidywater/wiki/References

See Also

chemdose_ph

Examples

```
water <- define_water(ph = 7, temp = 25, alk = 100, toc = 3.7, doc = 3.5, uv254 = .1)
dosed_water <- chemdose_ph(water, alum = 30) %>%
  chemdose_toc(alum = 30, coeff = "Alum")

dosed_water <- chemdose_ph(water, alum = 10, h2so4 = 10) %>%
  chemdose_toc(alum = 10, coeff = data.frame(
    x1 = 280, x2 = -73.9, x3 = 4.96, k1 = -0.028, k2 = 0.23, b = 0.068
  ))

dosed_reuse <- chemdose_ph(water, alum = 10, h2so4 = 10) %>%
  chemdose_toc(alum = 40, coeff = "Reuse Site Langmuir", fns = 0.6)


example_df <- water_df %>%
  define_water_df() %>%
  dplyr::mutate(FerricDose = seq(1, 12, 1)) %>%
  chemdose_toc_df(ferricchloride = FerricDose, coeff = "Ferric")

example_df <- water_df %>%
  define_water_df() %>%
  dplyr::mutate(ferricchloride = seq(1, 12, 1)) %>%
  chemdose_toc_df(coeff = "Ferric", pluck_cols = TRUE)
```

chloramine_conv *Data frame of conversion factors for estimating DBP formation from chloramines*

Description

A dataset containing conversion factors for calculating DBP formation

Usage

```
chloramine_conv
```

Format

A dataframe with 17 rows and 3 columns

ID abbreviation of dbp species

alias full name of dbp species

percent specifies the percent of DBP formation predicted from chloramines compared to chlorine, assuming the same chlorine dose applied

Source

U.S. EPA (2001), Table 5-10

See references list at: https://github.com/BrownandCaldwell-Public/tidywater/wiki/References

cl2coeffs *Data frame of Cl2 decay coefficients*

Description

A dataset containing coefficients for calculating Cl2 decay

Usage

```
cl2coeffs
```

Format

A dataframe with 3 rows and 4 columns

treatment Specifies the treatment applied to the water

a Coefficient in chlorine decay model, associated with chlorine dose and time

b Coefficient in chlorine decay model, associated with chlorine dose & organics

c Exponent in chlorine decay model, associated with chlorine dose & organics

Source

U.S. EPA (2001)

`convert_flow` *Convert flow rate*

Description

Given units for flow rate, convert to different units of flow rate.

Usage

```
convert_flow(value, startunit, endunit, sg = 1)
```

Arguments

`value`	Value of flow rate to be converted
`startunit`	Current units of flow rate
`endunit`	Desired units of flow rate
`sg`	Optional argument for specific gravity, default to 1. Required for converting between mass and volumetric flows.

Value

A numeric value for the converted flow

Examples

```
flow <- 40 #in MGD

convert_flow(flow, "MGD", "gpm")
```

`convert_units` *Calculate unit conversions for common compounds*

Description

This function takes a value and converts units based on compound name.

Usage

```
convert_units(value, formula = NULL, startunit = "mg/L", endunit = "M")
```

Arguments

value	Value to be converted
formula	Chemical formula of compound; accepts compounds in mweights. Optional when converting within the same unit type and basis (e.g., mg/L to g/L, mM to M, meq/L to eq/L, or mg/L CaCO3 to g/L CaCO3).
startunit	Units of current value, currently accepts g/L; g/L CaCO3; g/L N; g/L P; g/L Cl2; M; eq/L; and the same units with "m", "u", "n" prefixes
endunit	Desired units, currently accepts same as start units

Details

Mass concentrations may be expressed as CaCO3, N, P, or molecular chlorine (Cl2). Mass as CaCO3 is an equivalents basis: 1 eq of the compound = 50 g as CaCO3, so 61 mg/L HCO3 is 50 mg/L as CaCO3. Mass as N or P counts the atoms in the compound: 1 mol of (NH4)2SO4 is 28 g as N. Mass as Cl2 counts oxidizing capacity: 1 mol of NHCl2 is 2 mol as Cl2. "mg/L Cl" is treated as "mg/L Cl2".

Value

A numeric value for the converted parameter.

Examples

```
convert_units(50, "ca") # converts from mg/L to M by default
convert_units(50, "ca", "mg/L", "mg/L CaCO3")
convert_units(50, "ca", startunit = "mg/L", endunit = "eq/L")
```

---

convert_water *Convert* water *class object to a dataframe*

Description

This converts a water class to a dataframe with individual columns for each slot (water quality parameter) in the water. This is useful for one-off checks. For typical applications, use pluck_cols = TRUE in any _df function or pluck_water to choose relevant slots.

Use convert_water to keep all slots in the same units as the water.

Use convert_watermg to convert to more typical units. Converts all non-NA water slots with ions to mg/L, with the exception of br and mn, which are converted to ug/L. Total silicate is converted to mg/L as SiO2, and citrate species are converted to mg/L as citric acid, matching the define_water inputs. All other values remain unchanged.

The ks slot (a data frame of corrected equilibrium constants) is not included in the output. Access it directly with water@ks.

Usage

```
convert_water(water)

convert_watermg(water)
```

Arguments

water	A water class object

Value

A data frame containing columns for all non-NA water slots, except ks.

A data frame containing columns for all non-NA water slots with ions in mg/L.

Examples

```
# Generates 1 row dataframe
example_df <- define_water(ph = 7, temp = 20, alk = 100) %>%
  convert_water()

example_df <- water_df %>%
  define_water_df() %>%
  dplyr::mutate(to_dataframe = purrr::map(defined, convert_water)) %>%
  tidyr::unnest(to_dataframe) %>%
  dplyr::select(-defined)

water_defined <- define_water(7, 20, 50, 100, 80, 10, 10, 10, 10, tot_po4 = 1) %>%
  convert_watermg()
```

---

correct_k *Correct acid dissociation constants*

Description

This function calculates the corrected equilibrium constant for temperature and ionic strength

Usage

```
correct_k(water)
```

Arguments

water	Defined water with values for temperature and ion concentrations

Value

A dataframe with equilibrium constants for co3, po4, so4, ocl, and nh4.

Examples

```
water_defined <- define_water(7, 20, 50, 100, 80, 10, 10, 10, 10, tot_po4 = 1)
correct_k(water_defined)
```

`dbpcoeffs` *Data frame of DBP coefficients for predicting DBP formation*

Description

A dataset containing coefficients for calculating DBP formation

Usage

```
dbpcoeffs
```

Format

A dataframe with 30 rows and 10 columns

ID abbreviation of dbp species

alias full name of dbp species

water_type specifies which model the constants apply to, either treated or untreated water

A First coefficient in DBP model

a Second coefficient in DBP model, associated with TOC or DOC

b Third coefficient in DBP model, associated with Cl2

c Fourth coefficient in DBP model, associated with Br-

d Fifth coefficient in DBP model, associated with temperature

e Sixth coefficient in DBP model, associated with pH

f Seventh coefficient in DBP model, associated with reaction time

Source

U.S. EPA (2001)

See references list at: https://github.com/BrownandCaldwell-Public/tidywater/wiki/References

dbp_correction *Data frame of correction factors for estimating DBP formation as a function of location*

Description

A dataset containing correction factors for calculating DBP formation

Usage

```
dbp_correction
```

Format

A dataframe with 17 rows and 4 columns

ID abbreviation of dbp species

alias full name of dbp species

plant specifies the correction factor for modelling DBP formation within a treatment plant

ds specifies the correction factor for modelling DBP formation within the distribution system

Source

U.S. EPA (2001), Table 5-7

See references list at: https://github.com/BrownandCaldwell-Public/tidywater/wiki/References

decarbonate_ph *Apply decarbonation to a water*

Description

Calculates the new water quality (pH, alkalinity, etc) after a specified amount of CO2 is removed (removed as bicarbonate). The function takes an object of class "water" and a fraction of CO2 removed, then returns a water class object with updated water slots. For a single water, use `decarbonate_ph`; to apply the model to a dataframe, use `decarbonate_ph_df`. For a single water use `chemdose_toc`; for a dataframe use `chemdose_toc_df`. Use `pluck_cols = TRUE` to get values from the output water as new dataframe columns. For most arguments in the `_df` helper "use_col" default looks for a column of the same name in the dataframe. The argument can be specified directly in the function instead or an unquoted column name can be provided.

Usage

```
decarbonate_ph(water, co2_removed)

decarbonate_ph_df(
  df,
  input_water = "defined",
  output_water = "decarbonated",
  pluck_cols = FALSE,
  water_prefix = TRUE,
  co2_removed = "use_col"
)
```

Arguments

water	Source water of class "water" created by define_water
co2_removed	Fraction of CO2 removed
df	a data frame containing a water class column, which has already been computed using define_water_df. The df may include a column with names for each of the chemicals being dosed.
input_water	name of the column of water class data to be used as the input for this function. Default is "defined".
output_water	name of the output column storing updated water class object. Default is "decarbonated".
pluck_cols	Extract water slots modified by the function (ph, alk) into new numeric columns for easy access. Default to FALSE.
water_prefix	Append the output_water name to the start of the plucked columns. Default is TRUE.

Details

decarbonate_ph uses water@h2co3 to determine the existing CO2 in water, then applies chemdose_ph to match the CO2 removal.

Value

A water with updated pH/alk/etc.

decarbonate_ph_df returns a data frame containing a water class column with updated ph and alk (and pH dependent ions). Optionally, it also adds columns for each of those slots individually.

See Also

chemdose_ph

Examples

```
water <- define_water(ph = 4, temp = 25, alk = 5) %>%
  decarbonate_ph(co2_removed = .95)


example_df <- water_df %>%
  define_water_df() %>%
  decarbonate_ph_df(
    input_water = "defined", output_water = "decarb",
    co2_removed = .95, pluck_cols = TRUE
  )
```

---

define_chemical *Create a chemical class object given a chemical's parameters*

Description

Creates an S4 chemical class object that stores a chemical's formula, dose, strength, dose units, hydration, cost, and solids formula.

Usage

```
define_chemical(
  formula,
  dose = 0,
  strength = 100,
  dose_unit = "mg/L as chemical",
  hydration = NA_real_,
  cost = NA_real_,
  solids_formula = NA_character_
)
```

Arguments

formula	Associated chemical formula
dose	Dosage of the chemical
strength	from 100% to 1%, measure of the amount of chemical to its solution
dose_unit	Units the chemical is in, defaults to mg/L as chemical
hydration	Waters of hydration
cost	How much the chemical costs
solids_formula	formula for solids

Value

An S4 chemical class object with user defined parameters.

Examples

```
naoh_50p <- define_chemical(formula="naoh",dose=12, strength = 50, dose_unit = "M")
```

`define_water` *Create a water class object given water quality parameters*

Description

This function takes user-defined water quality parameters and creates an S4 "water" class object that forms the input and output of all tidywater models.

This function allows define_water to be added to a piped data frame. Its output is a `water` class, and can therefore be chained with "downstream" tidywater functions.

Usage

```
define_water(
  ph,
  temp = 25,
  alk,
  tot_hard,
  ca,
  mg,
  na,
  k,
  cl,
  so4,
  mno4,
  free_chlorine = 0,
  combined_chlorine = 0,
  tot_po4 = 0,
  tot_nh3 = 0,
  tot_ch3coo = 0,
  tot_bo4 = 0,
  tot_citric = 0,
  tot_sio4 = 0,
  tds,
  cond,
  toc,
  doc,
  uv254,
  br,
  f,
  fe,
  al,
  mn,
```

```
  no3,
  no2
)

define_water_df(
  df,
  output_water = "defined",
  pluck_cols = FALSE,
  water_prefix = TRUE,
  ph = "use_col",
  temp = "use_col",
  alk = "use_col",
  tot_hard = "use_col",
  ca = "use_col",
  mg = "use_col",
  na = "use_col",
  k = "use_col",
  cl = "use_col",
  so4 = "use_col",
  mno4 = "use_col",
  free_chlorine = "use_col",
  combined_chlorine = "use_col",
  tot_po4 = "use_col",
  tot_nh3 = "use_col",
  tot_bo4 = "use_col",
  tot_sio4 = "use_col",
  tot_ch3coo = "use_col",
  tot_citric = "use_col",
  tds = "use_col",
  cond = "use_col",
  toc = "use_col",
  doc = "use_col",
  uv254 = "use_col",
  br = "use_col",
  f = "use_col",
  fe = "use_col",
  al = "use_col",
  mn = "use_col",
  no3 = "use_col",
  no2 = "use_col"
)
```

Arguments

ph	Water pH
temp	Temperature in degree C
alk	Alkalinity in mg/L as CaCO3
tot_hard	Total hardness in mg/L as CaCO3

ca	Calcium in mg/L Ca2+
mg	Magnesium in mg/L Mg2+
na	Sodium in mg/L Na+
k	Potassium in mg/L K+
cl	Chloride in mg/L Cl-
so4	Sulfate in mg/L SO42-
mno4	Permanganate in mg/L MnO4-
free_chlorine	Free chlorine in mg/L as Cl2. Used when a starting water has a free chlorine residual.
combined_chlorine	Combined chlorine (chloramines) in mg/L as Cl2. Used when a starting water has a chloramine residual.
tot_po4	Phosphate in mg/L as PO4 3-. Used when a starting water has a phosphate residual.
tot_nh3	Total ammonia in mg/L as N
tot_ch3coo	Total acetate in mg/L
tot_bo4	Total borate (B(OH)4 -) in mg/L as B
tot_citric	Total citric acid in mg/L
tot_sio4	Total silicate in mg/L as SiO2
tds	Total Dissolved Solids in mg/L (optional if ions are known)
cond	Electrical conductivity in uS/cm (optional if ions are known)
toc	Total organic carbon (TOC) in mg/L
doc	Dissolved organic carbon (DOC) in mg/L
uv254	UV absorbance at 254 nm (cm-1)
br	Bromide in ug/L Br-
f	Fluoride in mg/L F-
fe	Iron in mg/L Fe3+
al	Aluminum in mg/L Al3+
mn	Manganese in ug/L Mn2+
no3	Nitrate in mg/L as N
no2	Nitrite in mg/L as N
df	a data frame containing columns with all the desired parameters with column names matching argument names in define_water
output_water	name of the output column storing updated parameters with the class, water. Default is "defined".
pluck_cols	Extract primary water slots (ph, alk, doc, uv254) into new numeric columns for easy access. Default to FALSE.
water_prefix	Append the output_water name to the start of the plucked columns. Default is TRUE.

Details

Carbonate balance is calculated and units are converted to mol/L. Ionic strength is determined from ions, TDS, or conductivity. Missing values are handled by defaulting to 0 or NA. Calcium defaults to 65 percent of the total hardness when not specified. DOC defaults to 95 percent of TOC. If `alk` is less than the non-carbonate alkalinity at the given pH (for example, hydroxide alone at high pH), carbonate species are set to 0 with a warning.

Value

define_water outputs a water class object where slots are filled or calculated based on input parameters. Water slots have different units than those input into the define_water function, as listed below.

ph pH, numeric, in standard units (SU).

temp temperature, numeric, in °C.

alk alkalinity, numeric, mg/L as CaCO3.

tds total dissolved solids, numeric, mg/L.

cond electrical conductivity, numeric, uS/cm.

tot_hard total hardness, numeric, mg/L as CaCO3.

kw dissociation constant for water, numeric, unitless.

alk_eq total alkalinity as equivalents, numeric, equivalent (eq).

carbonate_alk_eq carbonate alkalinity as equivalents, numeric, equivalent (eq).

phosphate_alk_eq phosphate alkalinity as equivalents, numeric, equivalent (eq).

ammonium_alk_eq ammonium alkalinity as equivalents, numeric, equivalent (eq).

borate_alk_eq borate alkalinity as equivalents, numeric, equivalent (eq).

silicate_alk_eq silicate alkalinity as equivalents, numeric, equivalent (eq).

hypochlorite_alk_eq hypochlorite alkalinity as equivalents, numeric, equivalent (eq).

citric_alk_eq citrate alkalinity as equivalents, numeric, equivalent (eq).

toc total organic carbon, numeric, mg/L.

doc dissolved organic carbon, numeric, mg/L.

bdoc biodegradable organic carbon, numeric, mg/L.

uv254 light absorption at 254 nm, numeric, cm-1.

dic dissolved inorganic carbon, numeric, mg/L as C.

is ionic strength, numeric, mol/L.

na sodium, numeric, mols/L.

ca calcium, numeric, mols/L.

mg magnesium, numeric, mols/L.

k potassium, numeric, mols/L.

cl chloride, numeric, mols/L.

so4 sulfate, numeric, mols/L.

mno4 permanganate, numeric, mols/L.

no3 nitrate, numeric, mols/L.

no2 nitrite, numeric, mols/L.

h2co3 carbonic acid (H2CO3*, the sum of dissolved CO2 and H2CO3), numeric, mols/L.

co2 dissolved carbon dioxide (H2CO3*, calculated from h2co3), numeric, mg/L as CO2.

hco3 bicarbonate, numeric, mols/L.

co3 carbonate, numeric, mols/L.

h2po4 dihydrogen phosphate, numeric, mols/L.

hpo4 hydrogen phosphate, numeric, mols/L.

po4 phosphate, numeric, mols/L.

h2citric dihydrogen citrate, numeric, mols/L.

h1citric hydrogen citrate, numeric, mols/L.

citric citrate ion, numeric, mols/L.

nh4 ammonium, numeric, mol/L as N.

h4bo4 borate, numeric, mol/L.

h3bo3 boric acid, numeric, mol/L.

h3sio4 trihydrogen silicate, numeric, mol/L.

h2sio4 dihydrogen silicate, numeric, mol/L.

ch3coo acetate, numeric, mol/L.

h hydrogen ion, numeric, mol/L.

oh hydroxide ion, numeric, mol/L.

tot_po4 total phosphate, numeric, mol/L.

tot_nh3 total ammonia, numeric, mol/L.

tot_co3 total carbonate, numeric, mol/L.

tot_bo4 total borate, numeric, mol/L.

tot_sio4 total silicate, numeric, mol/L.

tot_ch3coo total acetate, numeric, mol/L.

tot_citric total citric acid, numeric, mol/L.

br bromide, numeric, mol/L.

bro3 bromate, numeric, mol/L.

f fluoride, numeric, mol/L.

fe iron, numeric, mol/L.

al aluminum, numeric, mol/L.

mn manganese, numeric, mol/L.

free_chlorine free chlorine, numeric, mol/L.

ocl hypochlorite ion, numeric, mol/L.

combined_chlorine sum of chloramines, numeric, mol/L.

nh2cl monochloramine, numeric, mol/L.

nhcl2 dichloramine, numeric, mol/L.

ncl3 trichloramine, numeric, mol/L.

estimated list of parameters that were estimated based on other inputs.

ks data frame with corrected equilibrium constants given defined water initial temperature and ionic strength

chcl3 chloroform, numeric, ug/L.

chcl2br bromodichloromethane, numeric, ug/L.

chbr2cl dibromodichloromethane, numeric, ug/L.

chbr3 bromoform, numeric, ug/L.

tthm total trihalomethanes, numeric, ug/L.

mcaa chloroacetic acid, numeric, ug/L.

dcaa dichloroacetic acid, numeric, ug/L.

tcaa trichloroacetic acid, numeric, ug/L.

mbaa bromoacetic acid, numeric, ug/L.

dbaa dibromoacetic acid, numeric, ug/L.

haa5 sum of haloacetic acids, numeric, ug/L.

bcaa bromochloroacetic acid, numeric, ug/L.

cdbaa chlorodibromoacetic acid, numeric, ug/L.

dcbaa dichlorobromoacetic acid, numeric, ug/L.

tbaa tribromoacetic acid, numeric, ug/L.

A data frame containing a water class column.

Source

Crittenden et al. (2012) equation 5-38 - ionic strength from TDS

Snoeyink & Jenkins (1980) - ionic strength from conductivity

Lewis and Randall (1921), Crittenden et al. (2012) equation 5-37 - ionic strength from ion concentrations

Harned and Owen (1958), Crittenden et al. (2012) equation 5-45 - Temperature correction of dielectric constant (relative permittivity)

See Also

define_water

Examples

```
water_missingions <- define_water(ph = 7, temp = 15, alk = 100, tds = 10)
water_defined <- define_water(7, 20, 50, 100, 80, 10, 10, 10, 10, tot_po4 = 1)


example_df <- water_df %>%
  define_water_df() %>%
  balance_ions_df()

example_df <- water_df %>%
  define_water_df(output_water = "This is a column of water") %>%
  balance_ions_df(input_water = "This is a column of water")
```

`discons` *Dissociation constants and standard enthalpy for weak acids/bases*

Description

Equilibrium constants (k) and corresponding standard enthalpy of reaction values (deltah) for significant acids in water influencing pH at equilibrium. Includes carbonate, sulfate, phosphate, and hypochlorite. Standard enthalpy of reaction is calculated by taking the sum of the enthalpy of formation of each individual component minus the enthalpy of formation of the final product. e.g., the standard enthalpy of reaction for water can be calculated as: deltah_h2o = deltah_f_oh + deltah_f_h - deltah_f_h2o = -230 + 0 - (-285.83) = 55.83 kJ/mol. See MWH (2012) example 5-5 and Benjamin (2002) eq. 2.96.

Usage

```
discons
```

Format

A dataframe with 8 rows and 3 columns

ID Coefficient type

k Equilibrium constant

deltah Standard enthalpy in J/mol

Source

Benjamin (2015) Appendix A.1 and A.2.

See references list at: https://github.com/BrownandCaldwell-Public/tidywater/wiki/References

dissolve_cu *Calculate Dissolved Copper Concentration*

Description

This function takes a water defined by defined_water and output a column of dissolved copper. It is an empirical model developed based on bench-scale copper solubility testing that can be used to predict copper levels as a function of pH, DIC, and orthophosphate. For a single water, use `dissolve_cu`; to apply the model to a dataframe use `dissolve_cu_df`.

Usage

```
dissolve_cu(water)
```

Arguments

`water`	Source water object of class "water" created by `define_water`. Water must include ph and dic

Details

Dissolved copper is a function of pH, DIC (mg/L as C), and orthophosphate (mg/L as PO4). Output units are in mg/L. The model was fit on waters with pH 6.48 to 8.52, orthophosphate 0.2 to 3.1 mg/L as PO4, and DIC of 10, 50, and 100 mg/L as C (Lytle et al. 2018, equation 1); a warning is given for inputs outside these ranges.

DIC is not an input: `define_water` calculates it from pH and alkalinity. Lytle et al. (2018) give no alkalinity range, and the alkalinity that corresponds to a given DIC depends on pH. At 25 C, the 10 to 100 mg/L as C DIC range is roughly 24 to 244 mg/L as CaCO3 alkalinity at pH 6.5, 34 to 340 at pH 7, 39 to 390 at pH 7.5, and 41 to 410 at pH 8.

Value

`dissolve_cu` returns a column containing dissolved copper concentration in mg/L.

Source

Lytle et al (2018)

Examples

```
example_cu <- define_water(ph = 7.5, alk = 125, tot_po4 = 2) %>%
  dissolve_cu()
```

dissolve_cu_df *Calculate Dissolved Copper Concentration*

Description

Calculate Dissolved Copper Concentration

Usage

```
dissolve_cu_df(df, input_water = "defined", water_prefix = TRUE)
```

Arguments

df	a data frame containing a water class column, which has already been computed using define_water_df
input_water	name of the column of Water class data to be used as the input for this function. Default is "defined_water".
water_prefix	Append the output_water name to the start of the plucked columns. Default is TRUE.

Value

dissolve_cu_df returns a data frame containing the original data frame and a column for dissolved copper in mg/L.

Examples

```
cu_calc <- water_df %>%
  define_water_df() %>%
  dissolve_cu_df()
```

dissolve_pb *Simulate contributions of various lead solids to total soluble lead*

Description

This function takes a water data frame defined by define_water and outputs a dataframe of the controlling lead solid and total lead solubility. Lead solid solubility is calculated based on controlling solid. Total dissolved lead species (tot_dissolved_pb, M) are calculated based on lead complex calculations. For a single water, use dissolve_pb; to apply the model to a dataframe, use dissolve_pb_df. For most arguments, the _df "use_col" default looks for a column of the same name in the dataframe. The argument can be specified directly in the function instead or an unquoted column name can be provided.

Usage

```
dissolve_pb(
  water,
  hydroxypyromorphite = "Schock",
  pyromorphite = "Topolska",
  laurionite = "Nasanen"
)

dissolve_pb_df(
  df,
  input_water = "defined",
  output_col_solid = "controlling_solid",
  output_col_result = "pb",
  hydroxypyromorphite = "Schock",
  pyromorphite = "Topolska",
  laurionite = "Nasanen",
  water_prefix = TRUE
)
```

Arguments

water	Source water object of class "water" created by define_water. Water must include alk and is. If po4, cl, and so4 are known, those should also be included.
hydroxypyromorphite	defaults to "Schock", the constant, K, developed by Schock et al (1996). Can also use "Zhu".
pyromorphite	defaults to "Topolska", the constant, K, developed by Topolska et al (2016). Can also use "Xie".
laurionite	defaults to "Nasanen", the constant, K, developed by Nasanen & Lindell (1976). Can also use "Lothenbach".
df	a data frame containing a water class column, which has already been computed using define_water_df
input_water	name of the column of water class data to be used as the input. Default is "defined_water".
output_col_solid	name of the output column storing the controlling lead solid. Default is "controlling_solid".
output_col_result	name of the output column storing dissolved lead in M. Default is "pb".
water_prefix	name of the input water used for the calculation, appended to the start of output columns. Default is TRUE. Change to FALSE to remove the water prefix from output column names.

Details

The solid with lowest solubility will form the lead scale (controlling lead solid). Some lead solids have two k-constant options. The function will default to the EPA's default constants. The user

may change the constants to hydroxypyromorphite = "Zhu" or pyromorphite = "Xie" or laurionite = "Lothenbach"

Make sure that total dissolved solids, conductivity, or ca, na, cl, so4 are used in `define_water` so that an ionic strength is calculated.

Value

`dissolve_pb` returns a one row data frame containing only the controlling lead solid and modeled dissolved lead concentration.

`dissolve_pb_df` returns a data frame containing the controlling lead solid and modeled dissolved lead concentration as new columns.

Source

Code is from EPA's TELSS lead solubility dashboard https://github.com/USEPA/TELSS which is licensed under MIT License: Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions: The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

Wahman et al. (2021)

See references list at: https://github.com/BrownandCaldwell-Public/tidywater/wiki/References

Examples

```
example_pb <- define_water(
  ph = 7.5, temp = 25, alk = 93, cl = 240,
  tot_po4 = 0, so4 = 150, tds = 200
) %>%
  dissolve_pb()
example_pb <- define_water(
  ph = 7.5, temp = 25, alk = 93, cl = 240,
  tot_po4 = 0, so4 = 150, tds = 200
) %>%
  dissolve_pb(pyromorphite = "Xie")


example_df <- water_df %>%
  define_water_df() %>%
  dissolve_pb_df(output_col_result = "dissolved_lead", pyromorphite = "Xie")
```

gacbv_toc *Calculate maximum bed volumes to stay below target DOC*

Description

Calculates GAC filter bed volumes to achieve target effluent DOC according to the model developed in "Modeling TOC Breakthrough in Granular Activated Carbon Adsorbers" by Zachman and Summers (2010), or the USEPA WTP Model v. 2.0 Manual (2001). For a single water use `gacbv_toc`; for a dataframe use `gacbv_toc_df`. For most arguments in the `_df` helper "use_col" default looks for a column of the same name in the dataframe. The argument can be specified directly in the function instead or an unquoted column name can be provided.

Water must contain a DOC value (or TOC, from which define_water estimates DOC).

Usage

```
gacbv_toc(
  water,
  ebct = 10,
  model = "Zachman",
  media_size = "12x40",
  target_doc
)

gacbv_toc_df(
  df,
  input_water = "defined",
  model = "use_col",
  media_size = "use_col",
  ebct = "use_col",
  target_doc = "use_col",
  water_prefix = TRUE
)
```

Arguments

`water`	Source water object of class "water" created by define_water
`ebct`	Empty bed contact time (minutes). Model results are valid for 10 or 20 minutes. Default is 10 minutes.
`model`	Specifies which GAC TOC removal model to apply. Options are Zachman and WTP.
`media_size`	Size of GAC filter mesh. Model includes 12x40 and 8x30 mesh sizes. Default is 12x40.
`target_doc`	Target effluent DOC concentration (mg/L). Can be a vector to return a bed volume for each target.

df	a data frame containing a water class column, which has already been computed using define_water_df The df may include columns named for the chemical(s) being dosed.
input_water	name of the column of water class data to be used as the input for this function. Default is "defined".
water_prefix	Append the output_water name to the start of the plucked columns. Default is TRUE.

Details

GAC model for TOC removal

The function will calculate bed volume required to achieve given target DOC values. Both models were published in terms of TOC. GAC influent is normally coagulated and filtered, so particulate organic carbon is negligible and TOC = DOC; tidywater therefore applies the models to influent DOC.

Value

gacbv_toc returns a data frame of bed volumes that achieve the target DOC.

gacbv_toc_df returns a data frame with columns for bed volumes.

Source

See references list at: https://github.com/BrownandCaldwell-Public/tidywater/wiki/References

Zachman and Summers (2010)

USEPA (2001)

Examples

```
water <- define_water(ph = 8, toc = 2.5, uv254 = .05, doc = 1.5)
bed_volume <- gacbv_toc(water, media_size = "8x30", ebct = 20, model = "Zachman", target_doc = 0.8)


library(dplyr)

example_df <- water_df %>%
  define_water_df() %>%
  dplyr::mutate(
    model = "WTP",
    media_size = "8x30",
    ebct = 10,
    target_doc = rep(c(0.5, 0.8, 1), 4)
  ) %>%
  gacbv_toc_df()
```

gacrun_toc *Calculate TOC Concentration in GAC system*

Description

Returns a data frame with a breakthrough curve based on the DOC concentration after passing through GAC treatment, according to the model developed in "Modeling TOC Breakthrough in Granular Activated Carbon Adsorbers" by Zachman and Summers (2010), or the USEPA WTP Model v. 2.0 Manual (2001).

Water must contain a DOC value (or TOC, from which define_water estimates DOC).

Usage

```
gacrun_toc(
  water,
  ebct = 10,
  model = "Zachman",
  media_size = "12x40",
  bvs = c(2000, 20000, 100)
)

gacrun_toc_df(
  df,
  input_water = "defined",
  water_prefix = TRUE,
  ebct = "use_col",
  model = "use_col",
  media_size = "use_col",
  bvs = "use_col"
)
```

Arguments

water	Source water object of class "water" created by define_water
ebct	Empty bed contact time (minutes). Model results are valid for 10 or 20 minutes. Default is 10 minutes.
model	Specifies which GAC TOC removal model to apply. Options are Zachman and WTP.
media_size	Size of GAC filter mesh. If model is Zachman, can choose between 12x40 and 8x30 mesh sizes, otherwise leave as default. Defaults to 12x40.
bvs	If using WTP model, option to run the WTP model for a specific sequence of bed volumes, otherwise leave as default. Defaults c(2000, 20000, 100).
df	a data frame containing a water class column, which has already been computed using define_water_df. The df may include a column named for the coagulant being dosed, and a column named for the set of coefficients to use.

input_water	name of the column of water class data to be used as the input for this function. Default is "defined".
water_prefix	Append the input_water name to the start of the output columns. Default is TRUE.

Details

GAC model for TOC removal

The function will calculate bed volumes and normalized DOC breakthrough (DOCeff/DOCinf) given model type. Both models were developed using data sets from bench-scale GAC treatment studies using bituminous GAC and EBCTs of either 10 or 20 minutes. The specific mesh sizes used to develop the Zachman and Summers model were 12x40 or 8x30. The models were also developed using influent pH and DOC between specific ranges. Refer to the papers included in the references for more details. Both models were published in terms of TOC. GAC influent is normally coagulated and filtered, so particulate organic carbon is negligible and TOC = DOC; tidywater therefore applies the models to influent DOC.

Value

gacrun_toc returns a data frame with bed volumes and normalized breakthrough DOC values (x_norm, DOCeff/DOCinf).

gacrun_toc_df returns a data frame containing columns of the breakthrough curve (breakthrough and bed volume).

Source

See references list at: https://github.com/BrownandCaldwell-Public/tidywater/wiki/References

Zachman and Summers (2010)

USEPA (2001)

Examples

```
water <- define_water(ph = 8, toc = 2.5, uv254 = .05, doc = 1.5) %>%
  gacrun_toc(media_size = "8x30", ebct = 20, model = "Zachman")


example_df <- water_df %>%
  define_water_df() %>%
  gacrun_toc_df()
```

gac_toc *Calculate TOC Concentration in GAC system*

Description

Calculates TOC concentration after passing through GAC treatment according to the model developed in "Modeling TOC Breakthrough in Granular Activated Carbon Adsorbers" by Zachman and Summers (2010), or the logistics curve approach in EPA WTP Model v. 2.0 Manual (2001). For a single water use `gac_toc`; for a dataframe use `gac_toc_df`. Use pluck_water to get values from the output water as new dataframe columns. For most arguments in the `_df` helper "use_col" default looks for a column of the same name in the dataframe. The argument can be specified directly in the function instead or an unquoted column name can be provided.

Water must contain a DOC value (or TOC, from which define_water estimates DOC).

Usage

```
gac_toc(
  water,
  ebct = 10,
  model = "Zachman",
  media_size = "12x40",
  bed_vol,
  pretreat = "coag"
)

gac_toc_df(
  df,
  input_water = "defined",
  output_water = "gaced",
  model = "use_col",
  pluck_cols = FALSE,
  water_prefix = TRUE,
  media_size = "use_col",
  ebct = "use_col",
  bed_vol = "use_col",
  pretreat = "use_col"
)
```

Arguments

`water`	Source water object of class "water" created by define_water
`ebct`	Empty bed contact time (minutes). Model results are valid for 10 or 20 minutes. Defaults to 10 minutes.
`model`	Specifies which GAC TOC removal model to apply. Options are Zachman and WTP. Defaults to Zachman.

media_size	Size of GAC filter mesh. Model includes 12x40 and 8x30 mesh sizes. Defaults to 12x40.
bed_vol	Bed volume of GAC filter to predict effluent TOC for.
pretreat	Specifies the level of pretreatment prior to GAC treatment. Defaults to "coag". Other option is coagulant, ozonation, and biotreatment, called "o3biof".
df	a data frame containing a water class column, which has already been computed using define_water_df. The df may include columns named for the media_size, ebct, and bed volume.
input_water	name of the column of water class data to be used as the input for this function. Default is "defined".
output_water	name of the output column storing updated parameters with the class, water. Default is "gaced".
pluck_cols	Extract water slots modified by the function (doc, toc, uv254) into new numeric columns for easy access. Default to FALSE.
water_prefix	Append the output_water name to the start of the plucked columns. Default is TRUE.

Details

GAC model for TOC removal

The function will calculate DOC concentration by GAC adsorption in drinking water treatment. Both models were published in terms of TOC. GAC influent is normally coagulated and filtered, so particulate organic carbon is negligible and TOC = DOC; tidywater therefore applies the models to influent DOC. Effluent TOC is set equal to effluent DOC. UV254 concentrations are predicted based on a linear relationship with DOC from WTP Model Equation 5-93 and 5-94.

Value

gac_toc returns a water class object with updated DOC, TOC, and UV254 slots.

gac_toc_df returns a data frame containing a water class column with updated DOC, TOC, and UV254 slots

Source

See references list at: https://github.com/BrownandCaldwell-Public/tidywater/wiki/References

Zachman and Summers (2010)

U.S. EPA (2001)

Examples

```
water <- define_water(ph = 8, toc = 2.5, uv254 = .05, doc = 1.5) %>%
  gac_toc(media_size = "8x30", ebct = 20, model = "Zachman", bed_vol = 15000)


example_df <- water_df %>%
  define_water_df() %>%
```

```
  dplyr::mutate(
    model = "WTP",
    media_size = "8x30",
    ebct = 10,
    bed_vol = rep(c(12000, 15000, 18000), 4)
  ) %>%
  gac_toc_df()

example_df <- water_df %>%
  define_water_df("raw") %>%
  dplyr::mutate(
    model = "WTP",
    bed_vol = 15000
  ) %>%
  gac_toc_df(input_water = "raw")
```

`leadsol_constants` *Data frame of equilibrium constants for lead and copper solubility*

Description

A dataset containing equilibrium constants for lead solubility

Usage

```
leadsol_constants
```

Format

A dataframe with 38 rows and 3 columns

Solids:

species_name Name of lead solid or complex with possible _letter to cite different references

constant_name Reference ID for constants

log_value Equilibrium constant log value

source Source for equilibrium constant value

Source

Benjamin (2010)

Lothenbach et al. (1999)

Nasanen & Lindell (1976)

Powell et al. (2009)

Powell et al. (2005)

Schock et al. (1996)

Topolska et al. (2016)

Xie & Giammar (2007)

Zhu et al. (2015)

Wahman et al. (2021)

See references list at: https://github.com/BrownandCaldwell-Public/tidywater/wiki/References

---

`modify_water` *Modify slots in a* `water` *class object*

Description

This function modifies selected slots of a `water` class object without impacting the other parameters. For example, you can manually update "tthm" and the new speciation will not be calculated. This function is designed to make sure all parameters are stored in the correct units when manually updating a water. Some slots cannot be modified with this function because they are interconnected with too many others (usually pH dependent, eg, hco3). For those parameters, update define_water.

Usage

```
modify_water(water, slot, value, units)

modify_water_df(
  df,
  input_water = "defined",
  output_water = "modified",
  slot = "use_col",
  value = "use_col",
  units = "use_col"
)
```

Arguments

`water`	A water class object
`slot`	A vector of slots in the water to modify, eg, "tthm"
`value`	A vector of new values for the modified slots
`units`	A vector of units for each value being entered, typically one of c("mg/L", "ug/L", "M", "cm-1"). For ions any units supported by convert_units are allowed. For organic carbon, one of "mg/L", "ug/L". For uv254 one of "cm-1", "m-1". For DBPs, one of "ug/L" or "mg/L".
`df`	a data frame containing a water class column, which has already been computed using define_water_df
`input_water`	name of the column of water class data to be used as the input for this function. Default is "defined".
`output_water`	name of the output column storing updated parameters with the class, water. Default is "modified".

Details

In `modify_water_df`, `slot`, `value`, and `units` can each be a constant, an unquoted column name, or left as "use_col" to read a column of the same name ("slot", "value", "units"). Constants are applied to every row as given, so `slot = c("br", "na")`, `value = c(50, 60)`, `units = c("ug/L", "mg/L")` sets both slots on each water. Unlike other `_df` functions, vector constants are not expanded into one row per combination. Columns are read row by row.

Value

A data frame containing columns of selected parameters from a list of water class objects.

`modify_water_df` returns a data frame containing a water class column with updated slot

Examples

```
water1 <- define_water(ph = 7, alk = 100, tds = 100, toc = 5) %>%
  modify_water(slot = "toc", value = 4, units = "mg/L")

water2 <- define_water(ph = 7, alk = 100, tds = 100, toc = 5, ca = 10) %>%
  modify_water(slot = c("ca", "toc"), value = c(20, 10), units = c("mg/L", "mg/L"))


example_df <- water_df %>%
  define_water_df() %>%
  dplyr::mutate(bromide = 50) %>%
  modify_water_df(slot = "br", value = bromide, units = "ug/L")

example_df <- water_df %>%
  define_water_df() %>%
  modify_water_df(
    slot = c("br", "na"),
    value = c(50, 60),
    units = c("ug/L", "mg/L")
  )
```

---

mweights *Molar weights of relevant compounds*

Description

A dataset containing the molar weights of several compounds in g/mol. Column names are lowercase chemical formulas (with no charge), with the exception of the following coagulants: alum = Al2(SO4)3*14H2O, ferricchloride = FeCl3, ferricsulfate = Fe2(SO4)3*8.8H2O,

Usage

```
mweights
```

Format

A dataframe with one row and one column per compound

`opensys_ph` *Calculate pH for water in an open system*

Description

Calculates the new water quality (pH, alkalinity, pH dependent ions) for a water in an open system where CO2(aq) is at equilibrium with atmospheric CO2. The function takes an object of class "water" and the partial pressure of CO2, then returns a water class object with updated water slots. For a single water, use `opensys_ph`; to apply the model to a dataframe, use `opensys_ph_df`. For most arguments, the '_df helper "use_col" default looks for a column of the same name in the dataframe. The argument can be specified directly in the function instead or an unquoted column name can be provided.

Usage

```
opensys_ph(water, partialpressure = 10^-3.42)

opensys_ph_df(
  df,
  input_water = "defined",
  output_water = "opensys",
  pluck_cols = FALSE,
  water_prefix = TRUE,
  partialpressure = "use_col"
)
```

Arguments

`water`	Source water of class "water" created by define_water
`partialpressure`	Partial pressure of CO2 in the air in atm. Default is 10^-3.5 atm, which is approximately Pco2 at sea level.
`df`	a data frame containing a water class column, which has already been computed using define_water_df. The df may include a column with names for each of the chemicals being dosed.
`input_water`	name of the column of water class data to be used as the input for this function. Default is "defined".
`output_water`	name of the output column storing updated water class object. Default is "opensys".
`pluck_cols`	Extract water slots modified by the function (ph, alk) into new numeric columns for easy access. Default to FALSE.
`water_prefix`	Append the output_water name to the start of the plucked columns. Default is TRUE.

Details

opensys_ph uses the equilibrium concentration of CO2(aq) to determine the concentrations of carbonate species in the water and the pH by solving for the CO2 dose that results in a H2CO3 concentration equal to CO2(aq).

Value

A water with updated pH/alk/etc.

opensys_ph_df returns a data frame containing a water class column with updated ph and alk (and pH dependent ions). Optionally, it also adds columns for each of those slots individually.

See Also

chemdose_ph

Examples

```
water <- define_water(ph = 7, temp = 25, alk = 5) %>%
  opensys_ph()


example_df <- water_df %>%
  define_water_df() %>%
  opensys_ph_df(
    input_water = "defined", output_water = "opensys",
    partialpressure = 10^-4, pluck_cols = TRUE
  )
```

---

ozonate_bromate *Calculate bromate formation*

Description

Calculates bromate (BrO3-, ug/L) formation based on selected model. Required arguments include an object of class "water" created by define_water ozone dose, reaction time, and desired model. The function also requires additional water quality parameters defined in define_water including bromide, DOC or UV254 (depending on the model), pH, alkalinity (depending on the model), and optionally, ammonia (added when defining water using the tot_nh3 argument.) For a single water use ozonate_bromate; for a dataframe use ozonate_bromate_df. Use pluck_cols = TRUE to get values from the output water as new dataframe columns. For most arguments in the _df helper "use_col" default looks for a column of the same name in the dataframe. The argument can be specified directly in the function instead or an unquoted column name can be provided.

Usage

```
ozonate_bromate(water, dose, time, model = "Ozekin")

ozonate_bromate_df(
  df,
  input_water = "defined",
  output_water = "ozonated",
  pluck_cols = FALSE,
  water_prefix = TRUE,
  dose = "use_col",
  time = "use_col",
  model = "use_col"
)
```

Arguments

water	Source water object of class "water" created by define_water
dose	Applied ozone dose (mg/L as O3). Results typically valid for 1-10 mg/L, but varies depending on model.
time	Reaction time (minutes). Results typically valid for 1-120 minutes, but varies depending on model.
model	Model to apply. One of c("Ozekin", "Sohn", "Song", "Galey", "Siddiqui")
df	a data frame containing a water class column, which has already been computed using define_water_df. The df may include a column named for the applied chlorine dose (cl2), and a column for time in minutes.
input_water	name of the column of water class data to be used as the input for this function. Default is "defined".
output_water	name of the output column storing updated water class object. Default is "ozonated".
pluck_cols	Extract water slots modified by the function (bro3) into new numeric columns for easy access. Default to FALSE.
water_prefix	Append the output_water name to the start of the plucked columns. Default is TRUE.

Value

ozonate_bromate returns a single water class object with calculated bromate (ug/L).

ozonate_bromate_df returns a data frame containing a water class column with updated bro3 concentration. Optionally, it also adds columns for each of those slots individually.

Source

Ozekin (1994), Sohn et al (2004), Song et al (1996), Galey et al (1997), Siddiqui et al (1994)

See references list at: https://github.com/BrownandCaldwell-Public/tidywater/wiki/References

Examples

```
example_dbp <- define_water(8, 20, 66, toc = 4, uv254 = .2, br = 50) %>%
  ozonate_bromate(dose = 1.5, time = 5, model = "Ozekin")
example_dbp <- define_water(7.5, 20, 66, toc = 4, uv254 = .2, br = 50) %>%
  ozonate_bromate(dose = 3, time = 15, model = "Sohn")


example_df <- water_df %>%
  dplyr::slice_head(n = 6) %>%
  dplyr::mutate(br = 50) %>%
  define_water_df() %>%
  dplyr::mutate(
    dose = c(seq(.5, 3, .5)),
    OzoneTime = 30
  ) %>%
  ozonate_bromate_df(time = OzoneTime, model = "Sohn", pluck_cols = TRUE)
```

---

pactoccoeffs *Data frame of PAC TOC model coefficients*

Description

A dataset containing coefficients for calculating PAC TOC removal

Usage

```
pactoccoeffs
```

Format

A dataframe with 4 rows and 3 columns

pactype Specifies PAC type

A Constant in the PAC model

a Coefficient in PAC model, associated with DOC0

b Coefficient in PAC model, associated with dose

c Coefficient in PAC model, associated with time

Source

Cho (2007)

pac_toc *Calculate DOC Concentration in PAC system*

Description

Calculates DOC concentration multiple linear regression model found in 2-METHYLISOBORNEOL AND NATURAL ORGANIC MATTER ADSORPTION BY POWDERED ACTIVATED CARBON by HYUKJIN CHO (2007). Assumes all particulate TOC is removed when PAC is removed; therefore TOC = DOC in output. For a single water use `pac_toc`; for a dataframe use `pac_toc_df`. Use `pluck_cols = TRUE` to get values from the output water as new dataframe columns. For most arguments in the `_df` helper "use_col" default looks for a column of the same name in the dataframe. The argument can be specified directly in the function instead or an unquoted column name can be provided.

water must contain DOC or TOC value.

Usage

```
pac_toc(water, dose, time, type = "bituminous")

pac_toc_df(
  df,
  input_water = "defined",
  output_water = "paced",
  pluck_cols = FALSE,
  water_prefix = TRUE,
  dose = "use_col",
  time = "use_col",
  type = "use_col"
)
```

Arguments

`water`	Source water object of class "water" created by define_water
`dose`	Applied PAC dose (mg/L). Model results are valid for doses concentrations between 5 and 30 mg/L.
`time`	Contact time (minutes). Model results are valid for reaction times between 10 and 1440 minutes
`type`	Type of PAC applied, either "bituminous", "lignite", "wood".
`df`	a data frame containing a water class column, which has already been computed using define_water_df. The df may include columns named for the dose, time, and type
`input_water`	name of the column of water class data to be used as the input for this function. Default is "defined".
`output_water`	name of the output column storing updated water class object. Default is "paced". Pronouced P.A.ceed (not ideal we know).

pluck_cols	Extract water slots modified by the function (doc, toc, uv254) into new numeric columns for easy access. Default to FALSE.
water_prefix	Append the output_water name to the start of the plucked columns. Default is TRUE.

Details

The function will calculate DOC concentration by PAC adsorption in drinking water treatment. UV254 concentrations are predicted based on a linear relationship with DOC, and are set to 0 when that relationship gives a negative value. For waters with DOC below 1.3 mg/L, the model is applied with the same percent removal as a 1.3 mg/L water. With a dose or time of 0, the input water is returned unchanged.

Value

pac_toc returns a water class object with updated DOC, TOC, and UV254 slots.

pac_toc_df returns a data frame containing a water class column with updated DOC, TOC, and UV254 concentrations. Optionally, it also adds columns for each of those slots individually.

Source

See references list at: https://github.com/BrownandCaldwell-Public/tidywater/wiki/References

CHO(2007)

Examples

```
water <- define_water(toc = 2.5, uv254 = .05, doc = 1.5) %>%
  pac_toc(dose = 15, time = 50, type = "wood")


example_df <- water_df %>%
  define_water_df("raw") %>%
  dplyr::mutate(dose = seq(11, 22, 1), PACTime = 30) %>%
  pac_toc_df(input_water = "raw", time = PACTime, type = "wood", pluck_cols = TRUE)
```

---

plot_ions *Create summary plot of ions from water class*

Description

This function takes a water data frame defined by define_water and outputs an ion balance plot.

Usage

```
plot_ions(water)
```

Arguments

water	Source water vector created by link function here

Value

A ggplot object displaying the water's ion balance.

Examples

```
water <- define_water(7, 20, 50, 100, 20, 10, 10, 10, 10, tot_po4 = 1)
plot_ions(water)
```

---

plot_lead *Create dissolved lead and DIC contour plot given input data frame*

Description

This function takes a data frame and outputs a contour plot of dissolved lead and DIC plot. Assumes that the range of pH and dissolved inorganic carbon (DIC) occurs at a single temperature and TDS.

Usage

```
plot_lead(df, temp, tds, ph_range, dic_range)
```

Arguments

df	Source data as a data frame. Must have pH and DIC columns. Columns containing a single temperature and TDS can also be included.
temp	Temperature used to calculate dissolved lead concentrations. Defaults to a column in df.
tds	Total dissolved solids used to calculate dissolved lead concentrations. Defaults to a column in df.
ph_range	Optional argument to modify the plotted pH range. Input as c(minimum pH, maximum pH).
dic_range	Optional argument to modify the plotted DIC range. Input as c(minimum DIC, maximum DIC).

Value

A ggplot object displaying a contour plot of dissolved lead, pH, and DIC

Examples

```
historical <- data.frame(
  ph = c(7.7, 7.86, 8.31, 7.58, 7.9, 8.06, 7.95, 8.02, 7.93, 7.61),
  dic = c(
    14.86, 16.41, 16.48, 16.63, 16.86, 16.94, 17.05, 17.23,
    17.33, 17.34
  ),
  temp = 25,
  tds = 200
)
plot_lead(historical)
```

`pluck_water` *Pluck out a single parameter from a* `water` *class object*

Description

This function plucks one or more selected parameters from selected columns of `water` class objects. The names of the output columns will follow the form `water_parameter`

Usage

```
pluck_water(df, input_waters = c("defined"), parameter)
```

Arguments

`df`	a data frame containing a water class column, which has already been computed using define_water
`input_waters`	vector of names of the columns of water class data to be used as the input for this function.
`parameter`	vector of water class parameters to view outside the water column. Can also specify "all" to get all non-NA water slots except `ks` (see convert_water).

Value

A data frame containing columns of selected parameters from a list of water class objects.

See Also

convert_water

Examples

```
pluck_example <- water_df %>%
  define_water_df("raw") %>%
  pluck_water(input_waters = c("raw"), parameter = c("hco3", "doc"))
```

regulate_toc *Determine if TOC removal meets Stage 1 DBP Rule requirements*

Description

This function takes raw water alkalinity, raw water TOC, and finished water TOC. It then calculates the TOC removal percentage and checks compliance with the Stage 1 DBP Rule.

Usage

```
regulate_toc(
  alk_raw,
  toc_raw,
  toc_finished,
  raw_water,
  treated_water,
  skip_alternate = FALSE
)

regulate_toc_df(
  df,
  alk_raw = "use_col",
  toc_raw = "use_col",
  toc_finished = "use_col",
  input_raw_water = "defined",
  input_treated_water = "treated",
  skip_alternate = FALSE
)
```

Arguments

alk_raw	Raw water alkalinity (mg/L as calcium carbonate). Not needed if waters are provided.
toc_raw	Raw water total organic carbon (mg/L). Not needed if waters are provided.
toc_finished	Finished water total organic carbon (mg/L). Not needed if waters are provided.
raw_water	A water object of class "water" created by define_water() containing raw water parameters. If provided, the function will use the alk and toc slots from this object as inputs. If not provided, alk_raw and toc_raw must be specified.
treated_water	A water object of class "water" created by define_water() containing treated water parameters. If provided, the function will use the toc slot from this object as input for finished water TOC. If not provided, toc_finished must be specified.
skip_alternate	Logical value indicating whether to skip alternative compliance checks. By default, alternative compliance will be checked.
df	a data frame optionally containing columns for raw water alkalinity, raw water TOC, and finished water TOC

input_raw_water
: a name of the column of water class data to be used as the input raw water for this function. Default is "defined".

input_treated_water
: a name of the column of water class data to be used as the input treated water for this function. Default is "treated".

Details

The function prints the input parameters and the calculated removal percentage for TOC. It checks compliance with regulations considering the raw TOC, alkalinity, and removal percentage. It outputs a dataframe that contains the compliance status as a boolean, calculated removal precent as a numeric, and a comment that indicates internal reasoning for compliance status.

Value

A data frame containing the TOC removal compliance status, and a comment with the reasoning for the compliance status.

A data frame with compliance status, removal percent, and optional note columns.

Examples

```
regulate_toc(alk_raw=50, toc_raw=5, toc_finished=2)

water1 <- define_water(alk = 70, toc = 4)
water2 <- define_water(toc = 2.5)
regulate_toc(raw_water = water1, treated_water = water2)

regulate_toc(raw_water = define_water(alk = 50, toc = 5),
treated_water = define_water(toc = 2), skip_alternate = TRUE)


regulated <- water_df %>%
  dplyr::select(toc_raw = toc, alk_raw = alk) %>%
  regulate_toc_df(toc_finished = seq(2, 4, 0.1))

regulated <- water_df %>%
  define_water_df() %>%
  chemdose_ph_df(alum = 30, output_water = "dosed") %>%
  chemdose_toc_df("dosed") %>%
  pluck_water(c("coagulated", "defined"), c("toc", "alk")) %>%
 dplyr::select(toc_finished = coagulated_toc, toc_raw = defined_toc, alk_raw = defined_alk) %>%
  regulate_toc_df()

water1<-water_df |>
 define_water_df() |>
 chemdose_toc_df(output_water= "dosed", alum=20) |>
 regulate_toc_df(input_raw_water = "defined", input_treated_water= "dosed")
```

scavenge_oh *OH radical scavenging function*

Description

The background hydroxyl radical scavenging term is a measure of the degradation rate of non-target constituents in a water. The term is defined as the sum of the reaction rate of the constituent multiplied by the concentration of the constituent: `sum(k_i * [C]i)` where `k_i` is the reaction rate of a contaminant and `[C]i` its concentration. Supporting Information in Wang et al. 2019 provides the definition and additional methods for calculating background scavenging. For a data frame of waters, use `scavenge_oh_df`, which adds the scavenging term as a new column.

Usage

```
scavenge_oh(water, dose = 0, kdoc_adjust = 25000)

scavenge_oh_df(
  df,
  input_water = "defined",
  output_column = "oh_scavenging",
  dose = "use_col",
  kdoc_adjust = "use_col"
)
```

Arguments

`water`	Source water of tidywater 'water' object type
`dose`	Hydrogen peroxide (h2o2) dose applied in mg/L, defaults to 0
`kdoc_adjust`	Optional input for a different k_DOC rate constant for fitting data, defaults to 2.5e4 L/mg C/s. See Details for other reported values.
`df`	a data frame containing a water class column, which has already been computed using define_water_df. The df may include columns named for the H2O2 dose in mg/L and the DOC rate constant.
`input_water`	name of the column of water class data to be used as the input. Default is "defined".
`output_column`	name of the output column storing the OH radical scavenging term in s^-1. Default is "oh_scavenging".

Details

Rate constants used in the code:

- DOC: 2.5e4 L/mg C/s (Larson and Zepp 1988; 3.0e8 M_C^-1 s^-1 divided by 12,011 mg C/mol)
- HCO3-: 8.5e6 M^-1 s^-1 (Buxton et al. 1988)

- CO3 2-: 3.9e8 M^-1 s^-1 (Buxton et al. 1988)
- NO2-: 1.0e10 M^-1 s^-1 (Coddington et al. 1999; Vinge et al. 2020)
- NO3-: 1.0e5 M^-1 s^-1 (Buxton et al. 1988)
- H2O2: 2.7e7 M^-1 s^-1 (Buxton et al. 1988)

Other reported DOC rate constants, for use as `kdoc_adjust`. Sources report these per mole of carbon (M_C^-1 s^-1); they are converted to L/mg C/s by dividing by 12,011 mg C/mol. Brezonik and Katsoyiannis values are as compiled in Wols and Hofman-Caris (2012), Table 2.

- 2.41e4 (2.9e8 M_C^-1 s^-1): lake and river water, Brezonik and Fulkerson-Brekken (1998)
- 2.25e4 (2.7e8 M_C^-1 s^-1): Lake Zurich, Katsoyiannis et al. (2011)
- 1.67e4 (2.0e8 M_C^-1 s^-1): Lake Jonsvatnet, Katsoyiannis et al. (2011)
- 1.75e4 (2.1e8 M_C^-1 s^-1): Lake Greifensee, Katsoyiannis et al. (2011)
- 2.91e4 (3.5e8 M_C^-1 s^-1): effluent organic matter, Katsoyiannis et al. (2011)
- 1.67e4 (2.0e8 M_C^-1 s^-1): model value, Wols et al. (2014), Table 3
- 5.29e4 (6.35e8 M_C^-1 s^-1): midpoint of the 3.5e8 to 9.2e8 range cited by Vinge et al. (2021)

Value

`scavenge_oh` returns the total background hydroxyl radical scavenging term in units of s^-1.

`scavenge_oh_df` returns a data frame containing the original data frame and a column with the background hydroxyl radical scavenging term in s^-1.

Source

See references list at: https://github.com/BrownandCaldwell-Public/tidywater/wiki/References

Brezonik and Fulkerson-Brekken (1998)

Buxton et al. (1988)

Coddington et al. (1999)

Katsoyiannis et al. (2011)

Larson and Zepp (1988)

Vinge et al. (2020)

Vinge et al. (2021)

Wang et al. (2019)

Wols and Hofman-Caris (2012)

Wols et al. (2014)

Examples

```
water1 <- define_water(ph=7, doc=3, alk = 70, no3 = 4)
h2o2_dose<-20

background_scavenging <- scavenge_oh(water1, h2o2_dose)

background_SC_adjusted <- scavenge_oh(water1, h2o2_dose, 1.67e4) # Katsoyiannis et al. (2011)


example_df <- water_df %>%
  define_water_df() %>%
  scavenge_oh_df(dose = 10)

example_df <- water_df %>%
  define_water_df() %>%
  dplyr::mutate(h2o2 = seq(2, 24, 2)) %>%
  scavenge_oh_df(dose = h2o2, kdoc_adjust = 1.67e4, output_column = "sc")
```

`solvecost_chem` *Determine chemical cost*

Description

This function takes a chemical dose in mg/L, plant flow, chemical strength, and $/lb and calculates cost.

Usage

```
solvecost_chem(dose, flow, strength = 100, cost, time = "day")
```

Arguments

`dose`	Chemical dose in mg/L as chemical
`flow`	Plant flow in MGD
`strength`	Chemical product strength in percent. Defaults to 100 percent.
`cost`	Chemical product cost in $/lb
`time`	Desired output units, one of c("day", "month", "year"). Defaults to "day".

Value

A numeric value for chemical cost, $/time.

Examples

```
alum_cost <- solvecost_chem(dose = 20, flow = 10, strength = 49, cost = .22)

cost_data <- data.frame(
  dose = seq(10, 50, 10),
  flow = 10
) %>%
 dplyr::mutate(costs = solvecost_chem(dose = dose, flow = flow, strength = 49, cost = .22))
```

---

solvecost_labor *Determine labor cost*

Description

This function takes number of FTE and annual $/FTE and determines labor cost

Usage

```
solvecost_labor(fte, cost, time = "day")
```

Arguments

fte	Number of FTEs. Can be decimal.
cost	$/year per FTE
time	Desired output units, one of c("day", "month", "year"). Defaults to "day".

Value

A numeric value for labor $/time.

Examples

```
laborcost <- solvecost_labor(1.5, 50000)

cost_data <- data.frame(
  fte = seq(1, 10, 1)
) %>%
  dplyr::mutate(costs = solvecost_labor(fte = fte, cost = .08))
```

solvecost_power *Determine power cost*

Description

This function takes kW, % utilization, $/kWhr and determines power cost.

Usage

```
solvecost_power(power, utilization = 100, cost, time = "day")
```

Arguments

power	Power consumed in kW
utilization	Amount of time equipment is running in percent. Defaults to continuous.
cost	Power cost in $/kWhr
time	Desired output units, one of c("day", "month", "year"). Defaults to "day".

Value

A numeric value for power, $/time.

Examples

```
powercost <- solvecost_power(50, 100, .08)

cost_data <- data.frame(
  power = seq(10, 50, 10),
  utilization = 80
) %>%
 dplyr::mutate(costs = solvecost_power(power = power, utilization = utilization, cost = .08))
```

solvecost_solids *Determine solids disposal cost*

Description

This function takes coagulant doses in mg/L as chemical, removed turbidity, and cost ($/lb) to determine disposal cost.

Usage

```
solvecost_solids(
  alum = 0,
  ferricchloride = 0,
  ferricsulfate = 0,
  flow,
  toc_removed = 0,
  ca_removed = 0,
  mg_removed = 0,
  polymer = 0,
  turb,
  b = 1.5,
  cost,
  time = "day"
)
```

Arguments

alum	Hydrated aluminum sulfate Al2(SO4)3*14H2O + 6HCO3 -> 2Al(OH)3(am) +3SO4 + 14H2O + 6CO2
ferricchloride	Ferric Chloride FeCl3 + 3HCO3 -> Fe(OH)3(am) + 3Cl + 3CO2
ferricsulfate	Amount of ferric sulfate added in mg/L: Fe2(SO4)3*8.8H2O + 6HCO3 -> 2Fe(OH)3(am) + 3SO4 + 8.8H2O + 6CO2
flow	Plant flow in MGD
toc_removed	Amount of total organic carbon removed by the treatment process in mg/L
ca_removed	Amount of calcium removed by softening as mg/L Ca
mg_removed	Amount of magnesium removed by softening as mg/L Mg
polymer	Amount of polymer added in mg/L as chemical
turb	Turbidity removed in NTU
b	Correlation factor from turbidity to suspended solids. Defaults to 1.5.
cost	Disposal cost in $/lb
time	Desired output units, one of c("day", "month", "year"). Defaults to "day".

Value

A numeric value for disposal costs, $/time.

Source

https://water.mecc.edu/courses/ENV295Residuals/lesson3b.htm#:~:text=From%20the%20diagram%2C%20for%20example

Examples

```
alum_solidscost <- solvecost_solids(alum = 50, flow = 10, turb = 2, cost = 0.05)

cost_data <- data.frame(
  alum = seq(10, 50, 10),
  flow = 10
) %>%
  dplyr::mutate(costs = solvecost_solids(alum = alum, flow = flow, turb = 2, cost = 0.05))
```

`solvect_chlorine` *Determine disinfection credit from chlorine.*

Description

This function takes a water defined by define_water and other disinfection parameters and outputs a data frame of the required CT (`ct_required`), actual CT (`ct_actual`), and giardia log removal (`glog_removal`). For a single water, use `solvect_chlorine`; to apply the model to a dataframe, use `solvect_chlorine_df`. For most arguments, the `_df` helpers "use_col" default looks for a column of the same name in the dataframe. The argument can be specified directly in the function instead or an unquoted column name can be provided.

Usage

```
solvect_chlorine(water, time, residual, baffle, free_cl_slot = "residual_only")

solvect_chlorine_df(
  df,
  input_water = "defined",
  time = "use_col",
  residual = "use_col",
  baffle = "use_col",
  free_cl_slot = "residual_only",
  water_prefix = TRUE
)
```

Arguments

`water`	Source water object of class "water" created by `define_water`. Water must include ph and temp
`time`	Retention time of disinfection segment in minutes.
`residual`	Minimum chlorine residual in disinfection segment in mg/L as Cl2.
`baffle`	Baffle factor - unitless value between 0 and 1.
`free_cl_slot`	Defaults to "residual_only", which uses the residual argument. If "slot_only", the model will use the free_chlorine slot in the input water. "sum_with_residual", will use the sum of the residual argument and the free_chlorine slot.

df	a data frame containing a water class column, which has already been computed using define_water_df
input_water	name of the column of Water class data to be used as the input for this function. Default is "defined_water".
water_prefix	name of the input water used for the calculation will be appended to the start of output columns. Default is TRUE.

Details

CT actual is a function of time, chlorine residual, and baffle factor, whereas CT required is a function of pH, temperature, chlorine residual, and the standard 0.5 log removal of giardia requirement. CT required is an empirical regression equation developed by Smith et al. (1995) to provide conservative estimates for CT tables in USEPA Disinfection Profiling Guidance. Log removal is a rearrangement of the CT equations.

Value

`solvect_chlorine` returns a data frame containing required CT (mg/L*min), *actual CT (mg/L*min), giardia log removal, and virus log removal.

`solvect_chlorine_df` returns a data frame containing the original data frame and columns for required CT, actual CT, and giardia log removal.

Source

Smith et al. (1995)

USEPA (2020)

USEPA (1991)

See references list at: https://github.com/BrownandCaldwell-Public/tidywater/wiki/References

Examples

```
example_ct <- define_water(ph = 7.5, temp = 25) %>%
  solvect_chlorine(time = 30, residual = 1, baffle = 0.7)
ct_calc <- water_df %>%
  define_water_df() %>%
  solvect_chlorine_df(residual = 2, time = 10, baffle = .5)

chlor_resid <- water_df %>%
  dplyr::mutate(br = 50) %>%
  define_water_df() %>%
  dplyr::mutate(
    residual = seq(1, 12, 1),
    time = seq(2, 24, 2),
    baffle = 0.7
  ) %>%
  solvect_chlorine_df()
```

`solvect_o3` *Determine disinfection credit from ozone.*

Description

This function takes a water defined by `define_water()` and the first order decay curve parameters from an ozone dose and outputs a dataframe of actual CT, and log removal for giardia, virus, and crypto. For a single water, use `solvect_o3`; to apply the model to a dataframe, use `solvect_o3_df`. For most arguments, the `_df` helper "use_col" default looks for a column of the same name in the dataframe. The argument can be specified directly in the function instead or an unquoted column name can be provided.

Usage

```
solvect_o3(water, time, dose, kd, baffle)

solvect_o3_df(
  df,
  input_water = "defined",
  time = "use_col",
  dose = "use_col",
  kd = "use_col",
  baffle = "use_col",
  water_prefix = TRUE
)
```

Arguments

`water`	Source water object of class "water" created by `define_water()`. Water must include ph and temp
`time`	Retention time of disinfection segment in minutes.
`dose`	Ozone dose in mg/L. This value can also be the y intercept of the decay curve (often slightly lower than ozone dose.)
`kd`	First order decay constant. This parameter is optional. If not specified, the default ozone decay equations will be used.
`baffle`	Baffle factor - unitless value between 0 and 1.
`df`	a data frame containing a water class column, which has already been computed using `define_water_df()`.
`input_water`	name of the column of Water class data to be used as the input for this function. Default is "defined_water".
`water_prefix`	name of the input water used for the calculation will be appended to the start of output columns. Default is TRUE.

Details

First order decay curve for ozone has the form: `residual = dose * exp(kd*time)`. kd should be a negative number. Actual CT is an integration of the first order curve. The first 30 seconds are removed from the integral to account for instantaneous demand. CT is set to 0 when the contact time is shorter than those 30 seconds.

When kd is not specified, a default decay curve is used from the Water Treatment Plant Model (2002). This model does not perform well for ozone decay, so specifying the decay curve is recommended.

Value

`solvect_o3` returns a data frame containing actual CT (mg/L*min), giardia log removal, virus log removal, and crypto log removal.

`solvect_o3_df` returns a data frame containing the original data frame and columns for required CT, actual CT, and giardia log removal.

Source

USEPA (2020) Equation 4-4 through 4-7 https://www.epa.gov/system/files/documents/2022-02/disprof_bench_3rules_final_

See references list at: https://github.com/BrownandCaldwell-Public/tidywater/wiki/References

Examples

```
# Use kd from experimental data (recommended):
define_water(ph = 7.5, temp = 25) %>%
  solvect_o3(time = 10, dose = 2, kd = -0.5, baffle = 0.9)
# Use modeled decay curve:
define_water(ph = 7.5, alk = 100, doc = 2, uv254 = .02, br = 50) %>%
  solvect_o3(time = 10, dose = 2, baffle = 0.5)


ct_calc <- water_df %>%
  dplyr::mutate(br = 50) %>%
  define_water_df() %>%
  dplyr::mutate(
    dose = 2,
    O3time = 10,
  ) %>%
  solvect_o3_df(time = O3time, baffle = .7)
```

`solvedose_alk` *Calculate a desired chemical dose for a target alkalinity*

Description

This function calculates the required amount of a chemical to dose based on a target alkalinity and existing water quality. Returns numeric value for dose in mg/L. Uses uniroot on the chemdose_ph function. For a single water, use `solvedose_alk`; to apply the model to a dataframe, use `solvedose_alk_df`. For most arguments, the `_df` helper "use_col" default looks for a column of the same name in the dataframe. The argument can be specified directly in the function instead or an unquoted column name can be provided.

Usage

```
solvedose_alk(water, target_alk, chemical)

solvedose_alk_df(
  df,
  input_water = "defined",
  output_column = "dose",
  target_alk = "use_col",
  chemical = "use_col"
)
```

Arguments

`water`	Source water of class "water" created by `define_water`
`target_alk`	The final alkalinity in mg/L as CaCO3 to be achieved after the specified chemical is added.
`chemical`	The chemical to be added. Current supported chemicals include: acids: "hcl", "h2so4", "h3po4", bases: "naoh", "na2co3", "nahco3", "caoh2", "mgoh2"
`df`	a data frame containing a water class column, which has already been computed using define_water_df. The df may include a column with names for each of the chemicals being dosed.
`input_water`	name of the column of water class data to be used as the input. Default is "defined_water".
`output_column`	name of the output column storing doses in mg/L. Default is "dose_required".

Details

`solvedose_alk` uses `stats::uniroot()` on chemdose_ph to match the required dose for the requested alkalinity target.

Value

`solvedose_alk` returns a numeric value for the required chemical dose.

`solvedose_alk_df` returns a data frame containing the original data frame and columns for target alkalinity, chemical dosed, and required chemical dose.

See Also

solvedose_ph

Examples

```
dose_required <- define_water(ph = 7.9, temp = 22, alk = 100, 80, 50) %>%
  solvedose_alk(target_alk = 150, "naoh")

example_df <- water_df %>%
  define_water_df() %>%
  dplyr::mutate(finAlk = seq(100, 210, 10)) %>%
  solvedose_alk_df(chemical = "na2co3", target_alk = finAlk)
```

`solvedose_ph` *Calculate a desired chemical dose for a target pH*

Description

Calculates the required amount of a chemical to dose based on a target pH and existing water quality. The function takes an object of class "water", and user-specified chemical and target pH and returns a numeric value for the required dose in mg/L. For a single water, use `solvedose_ph`; to apply the model to a dataframe, use `solvedose_ph_df`. For most arguments, the `_df` helper "use_col" default looks for a column of the same name in the dataframe. The argument can be specified directly in the function instead or an unquoted column name can be provided.

Usage

```
solvedose_ph(water, target_ph, chemical)

solvedose_ph_df(
  df,
  input_water = "defined",
  output_column = "dose",
  target_ph = "use_col",
  chemical = "use_col"
)
```

Arguments

`water`	Source water of class "water" created by define_water
`target_ph`	The final pH to be achieved after the specified chemical is added.
`chemical`	The chemical to be added. Current supported chemicals include: acids: "hcl", "h2so4", "h3po4", "co2"; bases: "naoh", "na2co3", "nahco3", "caoh2", "mgoh2"
`df`	a data frame containing a water class column, which has already been computed using define_water_df. The df may include a column with names for each of the chemicals being dosed.
`input_water`	name of the column of water class data to be used as the input. Default is "defined".
`output_column`	name of the output column storing doses in mg/L. Default is "dose".

Details

solvedose_ph uses stats::uniroot() on chemdose_ph to match the required dose for the requested pH target.

Value

solvedose_ph returns a numeric value for the required chemical dose.

solvedose_ph_df returns a data frame containing the original data frame and columns for target pH, chemical dosed, and required chemical dose.

See Also

chemdose_ph, solvedose_alk

Examples

```
water <- define_water(ph = 7, temp = 25, alk = 10)

# Calculate required dose of lime to reach pH 8
solvedose_ph(water, target_ph = 8, chemical = "caoh2")


example_df <- water_df %>%
  define_water_df() %>%
  solvedose_ph_df(input_water = "defined", target_ph = 8.8, chemical = "naoh")
```

---

solvemass_chem *Convert mg/L of chemical to lb/day*

Description

This function takes a chemical dose in mg/L, plant flow in MGD, and chemical strength and calculates lb/day of product

Usage

```
solvemass_chem(dose, flow, strength = 100)
```

Arguments

dose	Chemical dose in mg/L as chemical
flow	Plant flow in MGD
strength	Chemical product strength in percent. Defaults to 100 percent.

Value

A numeric value for the chemical mass in lb/day.

Examples

```
alum_mass <- solvemass_chem(dose = 20, flow = 10, strength = 49)

library(dplyr)
mass_data <- tibble(
  dose = seq(10, 50, 10),
  flow = 10
) %>%
  mutate(mass = solvemass_chem(dose = dose, flow = flow, strength = 49))
```

---

solvemass_solids *Determine solids lb/day*

Description

This function takes coagulant doses in mg/L as chemical, removed turbidity, hardness removal in mg/L as chemical, and plant flow as MGD to determine solids production.

Usage

```
solvemass_solids(
  alum = 0,
  ferricchloride = 0,
  ferricsulfate = 0,
  flow,
  toc_removed = 0,
  ca_removed = 0,
  mg_removed = 0,
  polymer = 0,
  turb,
  b = 1.5,
  details = FALSE
)
```

Arguments

alum	Amount of hydrated aluminum sulfate added in mg/L as chemical: Al2(SO4)3*14H2O + 6HCO3 -> 2Al(OH)3(am) +3SO4 + 14H2O + 6CO2
ferricchloride	Amount of ferric chloride added in mg/L as chemical: FeCl3 + 3HCO3 -> Fe(OH)3(am) + 3Cl + 3CO2
ferricsulfate	Amount of ferric sulfate added in mg/L as chemical: Fe2(SO4)3*8.8H2O + 6HCO3 -> 2Fe(OH)3(am) + 3SO4 + 8.8H2O + 6CO2
flow	Plant flow in MGD
toc_removed	Amount of total organic carbon removed by the treatment process in mg/L
ca_removed	Amount of calcium hardness removed by softening as mg/L Ca

mg_removed	Amount of magnesium hardness removed by softening as mg/L Mg
polymer	Amount of polymer added in mg/L as chemical
turb	Turbidity removed in NTU
b	Correlation factor from turbidity to suspended solids. Defaults to 1.5.
details	Option to return a list with the solids mass and the individual contributions of each solids component. Defaults to FALSE.

Value

A numeric value for solids mass in lb/day. If details = TRUE, returns a named list with total and per-component values in lb/day.

Source

https://water.mecc.edu/courses/ENV295Residuals/lesson3b.htm#:~:text=From%20the%20diagram%2C%20for%20example

Examples

```
solids_mass <- solvemass_solids(alum = 50, flow = 10, turb = 20)

library(dplyr)
mass_data <- tibble(
  alum = seq(10, 50, 10),
  flow = 10
) %>%
  mutate(mass = solvemass_solids(alum = alum, flow = flow, turb = 20))

mass_details_true <- tibble(
  ferricsulfate = seq(10, 50, 10),
  polymer = seq(1, 5, 1),
  flow = 10
) %>%
  mutate(mass = solvemass_solids(ferricsulfate = ferricsulfate,
         polymer = polymer, flow = flow, turb = 20))


#'
```

---

solveresid_o3 *Determine ozone decay*

Description

This function applies the ozone decay model to a water from U.S. EPA (2001) equation 5-128. For a single water, use solveresid_o3; to apply the model to a dataframe, use solveresid_o3_df. For most arguments, the _df helper "use_col" default looks for a column of the same name in the dataframe. The argument can be specified directly in the function instead or an unquoted column name can be provided.

Usage

```
solveresid_o3(water, dose, time)

solveresid_o3_df(
  df,
  input_water = "defined",
  output_column = "o3resid",
  dose = "use_col",
  time = "use_col"
)
```

Arguments

water	Source water object of class water created by define_water
dose	Applied ozone dose in mg/L
time	Ozone contact time in minutes
df	a data frame containing a water class column, which has already been computed using define_water_df
input_water	name of the column of Water class data to be used as the input for this function. Default is "defined".
output_column	name of the output column storing doses in mg/L. Default is "dose_required".

Value

solveresid_o3 returns a numeric value for the residual ozone. The residual is 0 when the modeled ozone demand exceeds the dose.

solveresid_o3_df returns a data frame containing the original data frame and columns for ozone dosed, time, and ozone residual.

Source

U.S. EPA (2001)

See reference list at: https://github.com/BrownandCaldwell-Public/tidywater/wiki/References

Examples

```
ozone_resid <- define_water(7, 20, 100, doc = 2, toc = 2.2, uv254 = .02, br = 50) %>%
  solveresid_o3(dose = 2, time = 10)

ozone_resid <- water_df %>%
  dplyr::mutate(br = 50) %>%
  define_water_df() %>%
  solveresid_o3_df(dose = 2, time = 10)

ozone_resid <- water_df %>%
  dplyr::mutate(br = 50) %>%
  define_water_df() %>%
  dplyr::mutate(
```

```
  dose = seq(1, 12, 1),
  time = seq(2, 24, 2)
) %>%
solveresid_o3_df()
```

---

summarize_wq *Create summary table from water class*

Description

This function takes a water data frame defined by define_water and outputs a formatted summary table of specified water quality parameters.

summarise_wq() and summarize_wq() are synonyms.

Usage

```
summarize_wq(water, params = c("general"))

summarise_wq(water, params = c("general"))
```

Arguments

water	Source water vector created by define_water.
params	List of water quality parameters to be summarized. Options include "general", "ions", and "dbps". Defaults to "general" only.

Details

Use chemdose_dbp for modeled DBP concentrations.

Value

A knitr_kable table of specified water quality parameters.

Examples

```
# Summarize general parameters
water_defined <- define_water(7, 20, 50, 100, 80, 10, 10, 10, 10, tot_po4 = 1)
summarize_wq(water_defined)

# Summarize major cations and anions
summarize_wq(water_defined, params = list("ions"))
```

toc_compliance_table *Data frame of TOC removal requirements per Stage 1 DBP Rule*

Description

A dataset containing required TOC removal based on raw alkalinity and TOC

Usage

```
toc_compliance_table
```

Format

A dataframe with 9 rows and 5 columns

toc_min Minimum raw water TOC for compliance

toc_max Maximum raw water TOC for compliance

alk_min Minimum raw water alkalinity for compliance

alk_max Maximum raw water alkalinity for compliance

required_compliance Required percent TOC removal based on raw TOC andn alklalinity range

Source

Stage 1 Disinfection Byproduct Rule (1998)

vlog_removalcts *Data frame of virus log removal CT values*

Description

A table of virus log removal by disinfection with free chlorine, based on pH, temperature, and contact time

Usage

```
vlog_removalcts
```

Format

A dataframe with 35 rows and 4 columns

ph_range Specifies the pH of the input water

temp_value Specifies the temperature of the input water

ct_range Specifies the range of contact times to achieve virus removal

vlog_removal Level of log virus removal corresponding to ph, temp, and ct

Source

US EPA (1991) Table E-7

`water_df` *Data frame of water quality parameters*

Description

A dataset containing fabricated water quality to use as tidywater inputs. Parameters are set to reasonable water quality ranges. Parameters are as follows:

Usage

```
water_df
```

Format

A dataframe with 12 rows and 11 columns:

ph pH in standard units (SU)

temp Temperature in degree C

alk Alkalinity in mg/L as CaCO3

tot_hard Total hardness in mg/L as CaCO3

ca_hard Calcium hardness in mg/L as CaCO3

na Sodium in mg/L Na+

k Potassium in mg/L K+

cl Chloride in mg/L Cl-

so4 Sulfate in mg/L SO42-

tot_ocl Total chlorine in mg/L as Cl2

tot_po4 Total phosphate in mg/L as PO42-

Source

Fabricated for use in examples.

Index

∗ **datasets**
  bromatecoeffs, 9
  chemdosetoccoeff, 14
  chloramine_conv, 32
  cl2coeffs, 32
  dbp_correction, 37
  dbpcoeffs, 36
  discons, 46
  leadsol_constants, 57
  mweights, 59
  pactoccoeffs, 63
  toc_compliance_table, 87
  vlog_removalcts, 87
  water_df, 88

balance_ions, 3
balance_ions_df *(*balance_ions*)*, 3
biofilter_toc, 5
biofilter_toc_df *(*biofilter_toc*)*, 5
blend_waters, 7
blend_waters_df *(*blend_waters*)*, 7
bromatecoeffs, 9

calculate_activity, 10
calculate_corrosion, 10
calculate_corrosion_df *(*calculate_corrosion*)*, 10
calculate_hardness, 13
check_water, 13
chemdose_chloramine, *15*, 15
chemdose_chloramine_df *(*chemdose_chloramine*)*, 15
chemdose_chlordecay, 17
chemdose_chlordecay_df *(*chemdose_chlordecay*)*, 17
chemdose_dbp, 20, *86*
chemdose_dbp_df *(*chemdose_dbp*)*, 20
chemdose_inorgdemand, 22
chemdose_inorgdemand_df *(*chemdose_inorgdemand*)*, 22
chemdose_ph, 24, *29*, *31*, *38*, *61*, *80*, *82*
chemdose_ph_df *(*chemdose_ph*)*, 24
chemdose_toc, 29
chemdose_toc_df *(*chemdose_toc*)*, 29
chemdosetoccoeff, 14
chloramine_conv, 32
cl2coeffs, 32
convert_flow, 33
convert_units, *19*, *28*, 33, *58*
convert_water, *34*, 34, *67*
convert_watermg, *34*
convert_watermg *(*convert_water*)*, 34
correct_k, 35

dbp_correction, 37
dbpcoeffs, 36
decarbonate_ph, 37
decarbonate_ph_df *(*decarbonate_ph*)*, 37
define_chemical, 39
define_water, *3–8*, *10, 11*, *15, 16*, *18–21*, *26–30*, *34*, *38*, *40*, 40, *45*, *47–49*, *51*, *53*, *55*, *58*, *60–62*, *64, 65*, *67*, *76*, *80, 81*, *85, 86*
define_water(), *68*, *78*
define_water_df, *4*, *6*, *8*, *16*, *18*, *23*, *27*, *30*, *38*, *48, 49*, *52, 53*, *56*, *58*, *60*, *62*, *64*, *70*, *77*, *80, 81*, *85*
define_water_df *(*define_water*)*, 40
define_water_df(), *78*
discons, 46
dissolve_cu, 47
dissolve_cu_df, 48
dissolve_pb, 48
dissolve_pb_df *(*dissolve_pb*)*, 48

gac_toc, 55
gac_toc_df *(*gac_toc*)*, 55
gacbv_toc, 51
gacbv_toc_df *(*gacbv_toc*)*, 51
gacrun_toc, 53

gacrun_toc_df *(*gacrun_toc*)*, 53

leadsol_constants, 57

modify_water, 58
modify_water_df *(*modify_water*)*, 58
mweights, 59

opensys_ph, 60
opensys_ph_df *(*opensys_ph*)*, 60
ozonate_bromate, 61
ozonate_bromate_df *(*ozonate_bromate*)*, 61

pac_toc, 64
pac_toc_df *(*pac_toc*)*, 64
pactoccoeffs, 63
plot_ions, 65
plot_lead, 66
pluck_water, *7*, *55*, 67

regulate_toc, 68
regulate_toc_df *(*regulate_toc*)*, 68

scavenge_oh, 70
scavenge_oh_df *(*scavenge_oh*)*, 70
solvecost_chem, 72
solvecost_labor, 73
solvecost_power, 74
solvecost_solids, 74
solvect_chlorine, 76
solvect_chlorine_df *(*solvect_chlorine*)*, 76
solvect_o3, 78
solvect_o3_df *(*solvect_o3*)*, 78
solvedose_alk, 79, *82*
solvedose_alk_df *(*solvedose_alk*)*, 79
solvedose_ph, *80*, 81
solvedose_ph_df *(*solvedose_ph*)*, 81
solvemass_chem, 82
solvemass_solids, 83
solveresid_o3, 84
solveresid_o3_df *(*solveresid_o3*)*, 84
stats::uniroot(), *80*, *82*
summarise_wq *(*summarize_wq*)*, 86
summarize_wq, 86

toc_compliance_table, 87

uniroot, *28*

vlog_removalcts, 87

water_df, 88