Package 'kgen'

May 9, 2026

Type Package

Title A Tool for Calculating Stoichiometric Equilibrium Constants (Ks) for Seawater

Version 1.1.1

License MIT + file LICENSE

Description A unified software package simultaneously implemented in 'Python', 'R', and 'Matlab' providing a uniform and internally-consistent way of calculating stoichiometric equilibrium constants in modern and palaeo seawater as a function of temperature, salinity, pressure and the concentration of magnesium, calcium, sulphate, and fluorine.

Encoding UTF-8

Depends R (>= 4.0)

Imports rjson (>= 0.2.21), reticulate (>= 1.42.0), checkmate (>= 2.1.0), data.table (>= 1.14.6)

Suggests future (>= 1.58.0), progressr (>= 0.15.1), future.apply (>= 1.20.0), testthat (>= 3.0.0)

RoxygenNote 7.3.3

Config/testthat/edition 3

NeedsCompilation no

Author Dennis Mayk [aut, cre] (ORCID: <https://orcid.org/0000-0002-5017-1495>)

Maintainer Dennis Mayk <dm807@cantab.ac.uk>

Repository CRAN

Date/Publication 2026-03-28 06:50:02 UTC

Contents

`calc_fluorine` *Calculate fluorine*

Description

Calculate fluorine

Usage

```
calc_fluorine(sal)
```

Arguments

`sal`	Salinity

Value

fluorine

References

From Dickson et al., 2007, Table 2, Note: Sal / 1.80655 = Chlorinity

calc_ionic_strength *Ionic strength after Dickson (1990a); see Dickson et al. (2007)*

Description

Ionic strength after Dickson (1990a); see Dickson et al. (2007)

Usage

```
calc_ionic_strength(sal)
```

Arguments

sal	Salinity

Value

Ionic strength

calc_K *Calculate a single equilibrium constant*

Description

Calculate **a single** specified stoichiometric equilibrium constant at given temperature, salinity, pressure and the concentration of magnesium, calcium, sulphate, and fluorine.

Usage

```
calc_K(
  k,
  temp_c = 25,
  sal = 35,
  p_bar = NULL,
  magnesium = 0.0528171,
  calcium = 0.0102821,
  sulphate = NULL,
  fluorine = NULL,
  method = "r_polynomial"
)

calc_Ks(
  ks = NULL,
  temp_c = 25,
  sal = 35,
  p_bar = NULL,
```

```
  magnesium = 0.0528171,
  calcium = 0.0102821,
  sulphate = calc_sulphate(sal = sal),
  fluorine = calc_fluorine(sal = sal),
  method = "r_polynomial"
)
```

Arguments

k	K to be calculated
temp_c	Temperature (Celcius)
sal	Salinity
p_bar	Pressure (Bar) (optional)
magnesium	Magnesium concentration in mol/kgsw. Default is modern seawater (0.0528171). Should be the average magnesium concentration in seawater - a salinity correction is then applied to calculate the magnesium concentration in the sample.
calcium	Calcium concentration in mol/kgsw. Default is modern seawater (0.0102821). Should be the average calcium concentration in seawater - a salinity correction is then applied to calculate the calcium concentration in the sample.
sulphate	Sulphate concentration in mol/kgsw. Calculated from salinity if not given.
fluorine	Fluorine concentration in mol/kgsw. Calculated from salinity if not given.
method	string describing method which should be either 'myami', 'myami_polynomial', or 'r_polynomial' (Default: 'r_polynomial').
ks	character vectors of Ks to be calculated e.g., c("K0", "K1") (Default: NULL, calculate all Ks)

Value

A single K at given conditions

Data.table of **multiple** Ks at given conditions

Functions

- `calc_Ks()`: Wrapper to calculate **multiple** stoichiometric equilibrium constants at given temperature, salinity, pressure and the concentration of magnesium, calcium, sulphate, and fluorine.

Author(s)

Dennis Mayk

Examples

```
# Calculate K1 at default conditions
calc_K("K1", temp_c = 25, sal = 35)

# Calculate K1 with pressure correction
```

```
calc_K("K1", temp_c = 25, sal = 35, p_bar = 100)
# Calculate all Ks at default conditions
calc_Ks(temp_c = 25, sal = 35)

# Calculate specific Ks
calc_Ks(ks = c("K1", "K2"), temp_c = 25, sal = 35)


# Parallel execution (requires future + future.apply packages)
if (requireNamespace("future", quietly = TRUE)) {
  future::plan(future::multisession,
    workers = future::availableCores() - 1
  )

  dt_list <- as.list(data.table::CJ(
    temp_c = seq_len(40),
    sal = 30:40,
    p_bar = 0:100,
    magnesium = seq(0, 0.06, by = 0.01),
    calcium = seq(0, 0.06, by = 0.01)
  ))

  res <- do.call(what = calc_Ks, args = dt_list)
  future::plan(future::sequential)
}
```

`calc_K0` *Calculate K0*

Description

Calculate K0

Usage

```
calc_K0(coefficients, temp_c, sal)
```

Arguments

`coefficients`	Coefficients for K calculation
`temp_c`	Temperature (Celcius)
`sal`	Salinity

Value

K0

calc_K1 *Calculate K1*

Description

Calculate K1

Usage

```
calc_K1(coefficients, temp_c, sal)
```

Arguments

coefficients	Coefficients for K calculation
temp_c	Temperature (Celcius)
sal	Salinity

Value

K1

calc_K2 *Calculate K2*

Description

Calculate K2

Usage

```
calc_K2(coefficients, temp_c, sal)
```

Arguments

coefficients	Coefficients for K calculation
temp_c	Temperature (Celcius)
sal	Salinity

Value

K2

calc_KB *Calculate KB*

Description

Calculate KB

Usage

```
calc_KB(coefficients, temp_c, sal)
```

Arguments

coefficients	Coefficients for K calculation
temp_c	Temperature (Celcius)
sal	Salinity

Value

KB

calc_KF *Calculate KF*

Description

Calculate KF

Usage

```
calc_KF(coefficients, temp_c, sal)
```

Arguments

coefficients	Coefficients for K calculation
temp_c	Temperature (Celcius)
sal	Salinity

Value

KF

calc_KP1 *Calculate KP1*

Description

Calculate KP1

Usage

```
calc_KP1(coefficients, temp_c, sal)
```

Arguments

coefficients	Coefficients for K calculation
temp_c	Temperature (Celcius)
sal	Salinity

Value

KP1

calc_KP2 *Calculate KP2*

Description

Calculate KP2

Usage

```
calc_KP2(coefficients, temp_c, sal)
```

Arguments

coefficients	Coefficients for K calculation
temp_c	Temperature (Celcius)
sal	Salinity

Value

KP2

calc_KP3 *Calculate KP3*

Description

Calculate KP3

Usage

```
calc_KP3(coefficients, temp_c, sal)
```

Arguments

coefficients	Coefficients for K calculation
temp_c	Temperature (Celcius)
sal	Salinity

Value

KP3

calc_KS *Calculate KS*

Description

Calculate KS

Usage

```
calc_KS(coefficients, temp_c, sal)
```

Arguments

coefficients	Coefficients for K calculation
temp_c	Temperature (Celcius)
sal	Salinity

Value

KS

calc_KSi *Calculate KSi*

Description

Calculate KSi

Usage

```
calc_KSi(coefficients, temp_c, sal)
```

Arguments

coefficients	Coefficients for K calculation
temp_c	Temperature (Celcius)
sal	Salinity

Value

KSi

calc_Ksp *Calculate Ksp*

Description

Calculate Ksp

Usage

```
calc_Ksp(coefficients, temp_c, sal)
```

Arguments

coefficients	Coefficients for K calculation
temp_c	Temperature (Celcius)
sal	Salinity

Value

Ksp

calc_KW *Calculate KW*

Description

Calculate KW

Usage

```
calc_KW(coefficients, temp_c, sal)
```

Arguments

coefficients	Coefficients for K calculation
temp_c	Temperature (Celcius)
sal	Salinity

Value

KW

calc_pc *Calculate pressure correction factor for Ks*

Description

Calculate pressure correction factor for Ks

Usage

```
calc_pc(coefficients, temp_c, p_bar)
```

Arguments

coefficients	Coefficients for K calculation
temp_c	Temperature (Celcius)
p_bar	Pressure (Bar)

Value

Pressure correction factor

References

From Millero et al. (2007, doi:10.1021/cr0503557), Eqns 38-40

calc_pressure_correction

Calculate pressure correction factor

Description

Calculate pressure correction factor for a specified equilibrium constant.

Usage

```
calc_pressure_correction(k, temp_c, p_bar)
```

Arguments

k	K to be calculated
temp_c	Temperature (Celcius)
p_bar	Pressure (Bar)

Value

pressure correction factor

Author(s)

Dennis Mayk

Examples

```
calc_pressure_correction("K1", temp_c = 25, p_bar = 100)
```

calc_seawater_correction

Kgen seawater composition correction function

Description

Kgen seawater composition correction function

Usage

```
calc_seawater_correction(
  k,
  sal,
  temp_c,
  magnesium = 0.0528171,
  calcium = 0.0102821,
  method = "r_polynomial"
)
```

Arguments

k	K to be calculated
sal	Salinity
temp_c	Temperature (Celcius)
magnesium	Magnesium concentration in mol/kgsw. Default is modern seawater (0.0528171). Should be the average magnesium concentration in seawater - a salinity correction is then applied to calculate the magnesium concentration in the sample.
calcium	Calcium concentration in mol/kgsw. Default is modern seawater (0.0102821). Should be the average calcium concentration in seawater - a salinity correction is then applied to calculate the calcium concentration in the sample.
method	string describing method which should be either 'myami', 'myami_polynomial', or 'r_polynomial' (Default: 'r_polynomial').

Value

list of seawater correction factors

Author(s)

Dennis Mayk

Examples

```
calc_seawater_correction("K1", sal = 35, temp_c = 25)
```

---

calc_sulphate *Calculate sulphate*

Description

Calculate sulphate

Usage

```
calc_sulphate(sal)
```

Arguments

sal	Salinity

Value

sulphate

References

From Dickson et al., 2007, Table 2, Note: Sal / 1.80655 = Chlorinity

kgen_poly *Kgen R polynomial function*

Description

Kgen R polynomial function

Usage

```
kgen_poly(sal, temp_c, magnesium = 0.0528171, calcium = 0.0102821)
```

Arguments

sal	Salinity
temp_c	Temperature (Celcius)
magnesium	Magnesium concentration in mol/kgsw. Default is modern seawater (0.0528171). Should be the average magnesium concentration in seawater - a salinity correction is then applied to calculate the magnesium concentration in the sample.
calcium	Calcium concentration in mol/kgsw. Default is modern seawater (0.0102821). Should be the average calcium concentration in seawater - a salinity correction is then applied to calculate the calcium concentration in the sample.

Author(s)

Dennis Mayk

K_fns *List of all functions*

Description

List of all functions

Usage

```
K_fns
```

Format

An object of class `list` of length 13.

Index