# Package: evrfish (via r-universe)

June 1, 2026

**Title** R Tools for EVR Fish Projects

**Version** 0.7.0

**Description** Provides functions linearly interpolate missing values and calculate growing season degree days and growing degree days from daily water temperature data. It also exports fish codes for British Columbia and Alberta from the `fishbc` package.

**License** MIT + file LICENSE

**URL** https://poissonconsulting.github.io/evrfish/, https://github.com/poissonconsulting/evrfish

**BugReports** https://github.com/poissonconsulting/evrfish/issues/

**Depends** R (>= 4.1)

**Imports** chk, dplyr, dttr2, fishbc (>= 0.2.1), gsdd (>= 0.3.0), lifecycle, rlang, tibble, utils

**Suggests** ggplot2, knitr, latex2exp, readr, rmarkdown, scales, testthat (>= 3.0.0)

**VignetteBuilder** knitr

**Remotes** poissonconsulting/gsdd

**Config/testthat/edition** 3

**Encoding** UTF-8

**LazyData** true

**Roxygen** list(markdown = TRUE)

**RoxygenNote** 7.3.3.9000

**Config/Needs/website** poissonconsulting/poissontemplate

**Config/pak/sysreqs** libicu-dev

**Repository** https://poissonconsulting.r-universe.dev

**Date/Publication** 2025-10-16 03:41:47 UTC

**RemoteUrl** https://github.com/poissonconsulting/evrfish

**RemoteRef** HEAD

**RemoteSha** 9e83712dce38831a112b583ecc798000279a9374

# Contents



---

`aggregate_water_temp_data` *Aggregate Water Temperature Data*

---

## Description

Aggregates subdaily water temperature data to mean daily water temperature. If the water temperature data spans less than the min_coverage than a missing value is return instead.

## Usage

```
aggregate_water_temp_data(
  data,
  ...,
  min_coverage = 0.875,
  date_time = "date_time",
  value = "water_temperature"
)
```

## Arguments

| | |
|---|---|
| `data` | A data frame. |
| `...` | These dots are for future extensions and must be empty. |
| `min_coverage` | A numeric value of the minimum coverage as a proportion. |
| `date_time` | A string indicating the column name of the POSIXct vector. |
| `value` | A string indicating the column name of the value vector. |

## Details

The min_coverage is converted into the minimum number of non-missing values in each day based on how many values of the shortest time interval within the time series are required to achieve the minimum coverage. For example if the date times are every 15 minutes then 24 / 0.25 = 96 values are required for 100% coverage and 84 values are required for the default 87.5% minimum coverage.

## Value

A data frame with a date column and update water temperature column.

## Examples

```
data <- data.frame(
  date_time =
    as.POSIXct(c(
      "2021-05-07 00:00:00", "2021-05-07 08:00:00",
      "2021-05-07 16:00:00"
    )),
  water_temperature = c(5, 5, 7)
)

aggregate_water_temp_data(data)
```

---

`classify_time_series_data` *Classify Time Series Data*

---

## Description

Time series data will be either classified as reasonable, questionable, or erroneous in the status_id column or NA if the value is missing.

## Usage

```
classify_time_series_data(
  data,
  ...,
  date_time = "date_time",
  value = "value",
  questionable_min = 0,
  questionable_max = 30,
  erroneous_min = -0.5,
  erroneous_max = 40,
  questionable_rate = 2,
  erroneous_rate = 5,
  questionable_buffer = 1,
```

```
  erroneous_buffer = 1,
  gap_range = 5
)
```

## Arguments

| | |
|---|---|
| data | A data frame. |
| ... | These dots are for future extensions and must be empty. |
| date_time | A string indicating the column name of the POSIXct vector. |
| value | A string indicating the column name of the value vector. |
| questionable_min | A numeric value indicating the lower bound of the questionable range of temperature values. |
| questionable_max | A numeric value indicating the upper bound of the questionable range of temperature values. |
| erroneous_min | A numeric value indicating the lower bound of the erroneous range of temperature values. |
| erroneous_max | A numeric value indicating the upper bound of the erroneous range of temperature values. |
| questionable_rate | A numeric value indicating the rate of change (temperature per hour) of temperature values that is considered questionable. |
| erroneous_rate | A numeric value indicating the rate of change (temperature per hour) of temperature values that is considered erroneous. |
| questionable_buffer | A numeric value indicating the buffer in hours for questionable values. |
| erroneous_buffer | A numeric value indicating the number of hours buffer for erroneous values. |
| gap_range | A numeric value indicating the number of hours between two non reasonable values that will be coded as questionable or erroneous. |

## Details

The function only works on a single time series.

The function will error if there are missing or duplicated date time.

The data is processed by:

1. Classifying the time series values based on their values (‘questionable_min, questionable_max, erroneous_min, erroneous_max).
2. The rate of change to each value is then calculated and the values are classified based on the absolute rate of change (questionable_rate, erroneous_rate).
3. Adjacent values to all questionable/erroneous are then coded as questionable/erroneous.
4. Next any value within the time buffer of a questionable/erroneous value is classified as questionable/erroneous (questionable_buffer, erroneous_buffer).
5. In addition, ignoring the buffer, reasonable values between two questionable/erroneous values are coded as questionable if the hourly duration of the gap is within the (gap_range).

## Value

The original data frame sorted by the date time with a status_id column.

## Examples

```
data <- data.frame(
  date_time =
    as.POSIXct(c(
      "2021-05-07 08:00:00", "2021-05-07 09:00:00",
      "2021-05-07 10:00:00", "2021-05-07 11:00:00", "2021-05-07 12:00:00",
      "2021-05-07 13:00:00"
    )),
  water_temperature = c(4.124, 4.078, 4.102, 4.189, 4.243, 6.578)
)

classify_time_series_data(data, value = "water_temperature")
```

---

# classify_water_temp_data

*Classify Water Temperature Data*

---

## Description

A wrapper on `classify_time_series_data()` with the arguments set for water temperature data.

## Usage

```
classify_water_temp_data(
  data,
  questionable_min = 0,
  questionable_max = 30,
  erroneous_min = -0.5,
  erroneous_max = 40,
  questionable_rate = 2,
  erroneous_rate = 5,
  questionable_buffer = 1,
  erroneous_buffer = 1,
  gap_range = 5,
  date_time = "temperature_date_time",
  value = "water_temperature"
)
```

## Arguments

| | |
|---|---|
| `data` | A data frame. |
| `questionable_min` | A numeric value indicating the lower bound of the questionable range of temperature values. |

questionable_max
: A numeric value indicating the upper bound of the questionable range of temperature values.

erroneous_min
: A numeric value indicating the lower bound of the erroneous range of temperature values.

erroneous_max
: A numeric value indicating the upper bound of the erroneous range of temperature values.

questionable_rate
: A numeric value indicating the rate of change (temperature per hour) of temperature values that is considered questionable.

erroneous_rate
: A numeric value indicating the rate of change (temperature per hour) of temperature values that is considered erroneous.

questionable_buffer
: A numeric value indicating the buffer in hours for questionable values.

erroneous_buffer
: A numeric value indicating the number of hours buffer for erroneous values.

gap_range
: A numeric value indicating the number of hours between two non reasonable values that will be coded as questionable or erroneous.

date_time
: A string indicating the column name of the POSIXct vector.

value
: A string indicating the column name of the value vector.

## Value

A data frame

## Examples

```
data <- data.frame(
  temperature_date_time =
    as.POSIXct(c(
      "2021-05-07 08:00:00", "2021-05-07 09:00:00",
      "2021-05-07 10:00:00", "2021-05-07 11:00:00", "2021-05-07 12:00:00",
      "2021-05-07 13:00:00"
    )),
  water_temperature = c(4.124, 4.078, 4.102, 4.189, 4.243, 6.578)
)

classified_data <- classify_water_temp_data(data)
```

---

# date_atus *Calculate Date of Accumulated Thermal Units (ATUs)*

---

## Description

A wrapper on `gsdd::date_atus()` to calculats the date on which a specified number of Accumulated Thermal Units (ATUs) are exceeded.

### Usage

```
date_atus(x, atus = 600, start_date = as.Date("1972-03-01"))
```

### Arguments

| | |
|---|---|
| x | A data frame with two columns date and temperature. date, which must be of class Date provides the dates and temperature which must be a numeric vector provides the mean daily water temperature in degrees centigrade. |
| atus | A positive number of the accumulated thermal units to exceed. |
| start_date | A Date scalar of the first date within each year to consider (the year is ignored). #' If start_date occurs before the end_date (when ignoring the year) then the window is considered to span two calendar years. |

### Value

A tibble with four columns year, start_date, end_date and atus.

### Examples

```
date_atus(gsdd::temperature_data)
```

---

## freshwaterfish *BC Fish Data*

---

### Description

Curated data on the codes, classification and conservation status of freshwater fishes in British Columbia.

### Usage

```
freshwaterfish
```

### Format

An object of class tbl_df (inherits from tbl, data.frame) with 161 rows and 17 columns.

## gdd *Calculate Growing Degree Days (GDD)*

### Description

A wrapper on `gsdd::gdd()` to get the Growing Degree Days up to a date for the longest growing season.

### Usage

```
gdd(x, end_date = as.Date("1972-09-30"), min_length = 60, msgs = TRUE)
```

### Arguments

| | |
|---|---|
| `x` | A data frame with two columns `date` and `temperature`. `date`, which must be of class Date provides the dates and `temperature` which must be a numeric vector provides the mean daily water temperature in degrees centigrade. |
| `end_date` | A Date scalar of the last date within each year to consider (the year is ignored). |
| `min_length` | A whole number of the minimum number of values to consider. |
| `msgs` | A flag specifying whether to provide messages. |

### See Also

`gsdd::gdd()`, `gsdd()` and `gss()`.

### Examples

```
gdd(gsdd::temperature_data)
```

## gsdd *Calculate Growing Season Degree Days (GSDD)*

### Description

A wrapper on `gsdd::gsdd()` to get the Growing Season Degree Days for the longest growing season.

### Usage

```
gsdd(x, min_length = 120, msgs = TRUE)
```

### Arguments

| | |
|---|---|
| x | A data frame with two columns date and temperature. date, which must be of class Date provides the dates and temperature which must be a numeric vector provides the mean daily water temperature in degrees centigrade. |
| min_length | A whole number of the minimum number of values to consider. |
| msgs | A flag specifying whether to provide messages. |

### See Also

gsdd::gsdd(), gsdd() and gss().

### Examples

```
gsdd(gsdd::temperature_data)
```

---

## gsdd_cf *Calculate Growing Season Degree Days (GSDD)*

---

### Description

Soft-deprecated for gsdd::gsdd_vctr().

### Usage

```
gsdd_cf(x, ignore_truncation = FALSE, min_length = 120, msgs = TRUE)
```

### Arguments

| | |
|---|---|
| x | A numeric vector of the mean daily water temperature values for the period of interest in C. |
| ignore_truncation | A flag specifying whether to ignore truncation of the mean daily water temperature vector or a string of "start", "end", "none" (equivalent to FALSE) or "both" (equivalent to TRUE) specifying which type of truncation to ignore. |
| min_length | A whole number of the minimum number of values to consider. |
| msgs | A flag specifying whether to provide messages. |

### Value

A non-negative real number of the GSDD.

### See Also

gsdd::gsdd_vctr()

### Examples

```
gsdd_cf(c(rep(1, 10), rep(10, 20), rep(1, 200)))
gsdd_cf(gsdd::temperature_data$temperature)
```

## gss *Calculate Growing Seasons (GSS)*

### Description

A wrapper on `gsdd::gss()` to by default get all Growing Seasons ignoring truncation. For more information see `gsdd::gss()`.

### Usage

```
gss(x, min_length = 120, ignore_truncation = "end", pick = "all", msgs = TRUE)
```

### Arguments

| | |
|---|---|
| `x` | A data frame with two columns `date` and `temperature`. `date`, which must be of class Date provides the dates and `temperature` which must be a numeric vector provides the mean daily water temperature in degrees centigrade. |
| `min_length` | A whole number of the minimum number of values to consider. |
| `ignore_truncation` | A flag specifying whether to ignore truncation of the mean daily water temperature vector or a string of "start", "end", "none" (equivalent to FALSE) or "both" (equivalent to TRUE) specifying which type of truncation to ignore. |
| `pick` | A string specifying whether to pick the "longest", "shortest", "first" or "last" 'season' or the season with the "biggest" or "smallest" GSDD. By default the returned value is the the GSDD value for the "longest" 'season'. |
| `msgs` | A flag specifying whether to provide messages. |

### See Also

`gsdd::gss()`, `gsdd()` and `gss()`.

### Examples

```
gss(gsdd::temperature_data)
```

## gss_plot *Plot Growing Seasons (GSS)*

### Description

A wrapper on `gsdd::gss_plot()` to by default plot all Growing Seasons ignoring truncation. For more information see `gsdd::gss_plot()`.

## Usage

```
gss_plot(
  x,
  min_length = 60,
  ignore_truncation = TRUE,
  pick = "all",
  latex = FALSE,
  nrow = NULL,
  ncol = NULL,
  msgs = TRUE
)
```

## Arguments

| | |
|---|---|
| x | A data frame with two columns `date` and `temperature`. `date`, which must be of class Date provides the dates and `temperature` which must be a numeric vector provides the mean daily water temperature in degrees centigrade. |
| min_length | A whole number of the minimum number of values to consider. |
| ignore_truncation | A flag specifying whether to ignore truncation of the mean daily water temperature vector or a string of "start", "end", "none" (equivalent to FALSE) or "both" (equivalent to TRUE) specifying which type of truncation to ignore. |
| pick | A string specifying whether to pick the "longest", "shortest", "first" or "last" 'season' or the season with the "biggest" or "smallest" GSDD. By default the returned value is the the GSDD value for the "longest" 'season'. |
| latex | A flag specifying whether to use LaTeX to include degree symbol in y-axis label. |
| nrow | A count of the number of rows to facet by. |
| ncol | A count of the number of columns to facet by. |
| msgs | A flag specifying whether to provide messages. |

## See Also

`gsdd::gss_plot()` and `gss()`.

## Examples

```
gss_plot(gsdd::temperature_data)
```

---

`interpolate_numeric_vector` *Interpolate Numeric Vector*

---

## Description

Useful for filling in short runs of missing values in a time series.

## Usage

```
interpolate_numeric_vector(x, span = 3, tails = FALSE)
```

## Arguments

| | |
|---|---|
| x | A double or integer vector of with missing values to fill in using linear interpolation. |
| span | A whole number of the maximum span of missing values to interpolate. If a gap exceeds the span none of the values are interpolate. |
| tails | A flag specifying whether to fill in missing values at the start and end by setting them to be the same value as the closest adjacent non-missing value. |

## Details

`interpolate_numeric_vector()` is essentially a wrapper on `stats::approx()`.

## Value

A double or integer vector.

## Examples

```
interpolate_numeric_vector(c(1, NA, 4))
interpolate_numeric_vector(c(1L, NA, 4L))
interpolate_numeric_vector(c(1, NA, NA, NA, NA, 3))
interpolate_numeric_vector(c(1, NA, NA, NA, NA, 3), span = 4)
interpolate_numeric_vector(c(NA, NA, 10, 1, NA))
interpolate_numeric_vector(c(NA, NA, 10, 1, NA), tails = TRUE)
```

# Index