

# Package ‘STICr’

December 2, 2024

**Type** Package

**Title** Process Stream Temperature, Intermittency, and Conductivity (STIC) Sensor Data

**Description** A collection of functions for processing raw data from Stream Temperature, Intermittency, and Conductivity (STIC) loggers. ‘STICr’ (pronounced “sticker”) includes functions for tidying, calibrating, classifying, and doing quality checks on data from STIC sensors. Some package functionality is described in Wheeler/Zipper et al. (2023) <[doi:10.31223/X5636K](https://doi.org/10.31223/X5636K)>.

**License** AGPL (>= 3)

**URL** <https://github.com/HEAL-KGS/STICr>

**BugReports** <https://github.com/HEAL-KGS/STICr/issues>

**Encoding** UTF-8

**LazyData** true

**RoxygenNote** 7.3.2

**Version** 1.1

**Depends** R (>= 4.2)

**Imports** dplyr, lubridate, stringr, tidyr, methods

**Suggests** knitr, rmarkdown

**VignetteBuilder** knitr

**NeedsCompilation** no

**Author** Sam Zipper [aut, cre, cph] (<<https://orcid.org/0000-0002-8735-5757>>),  
Christopher T. Wheeler [aut] (<<https://orcid.org/0000-0001-9368-383X>>),  
Stephen Cook [ctb] (<<https://orcid.org/0000-0003-3642-1790>>),  
Delaney Peterson [ctb] (<<https://orcid.org/0000-0002-3444-4772>>),  
Sarah Godsey [ctb] (<<https://orcid.org/0000-0001-6529-7886>>)

**Maintainer** Sam Zipper <[samzipper@ku.edu](mailto:samzipper@ku.edu)>

**Repository** CRAN

**Date/Publication** 2024-12-02 15:40:24 UTC

## Contents

apply_calibration . . . . .	2
calibrated_stic_data . . . . .	3
calibration_standard_data . . . . .	3
classified_df . . . . .	4
classify_wetdry . . . . .	5
field_obs . . . . .	6
get_calibration . . . . .	6
qaqc_stic_data . . . . .	7
test_threshold . . . . .	8
tidy_hobo_data . . . . .	9
tidy_stic_data . . . . .	9
trim_hobo_data . . . . .	10
validate_stic_data . . . . .	11

<b>Index</b>	<b>13</b>
--------------	-----------

---

apply_calibration	<i>apply_calibration</i>
-------------------	--------------------------

---

## Description

This function takes the cleaned data frame generated by `tidy_hobo_data` and the fitted model object generated by `get_calibration`. It outputs a data frame with the same columns as the input, plus a calibrated specific conductivity column called SpC.

## Usage

```
apply_calibration(stic_data, calibration, outside_std_range_flag = TRUE)
```

## Arguments

<code>stic_data</code>	A data frame with a column named <code>condUncal</code> , for example as produced by the function <code>tidy_hobo_data</code> .
<code>calibration</code>	a model object relating <code>condUncal</code> to a standard of some sort, for example as produced by the function <code>get_calibration</code> .
<code>outside_std_range_flag</code>	a logical argument indicating whether the user would like to include an additional column flagging (with the letter "O") instances where the calibrated SpC value is outside the range of standards used to calibrate it.

## Value

The same data frame as input, except with a new column called SpC. This will be in the same units as the data used to develop the model calibration.

**Examples**

```
calibration <- get_calibration(calibration_standard_data)
calibrated_df <- apply_calibration(tidy_stic_data, calibration, outside_std_range_flag = TRUE)
head(calibrated_df)
```

---

calibrated\_stic\_data *Example calibrated STIC output data.*

---

**Description**

Calibrated STIC data used for function examples.

**Usage**

```
calibrated_stic_data
```

**Format**

## 'calibrated\_stic\_data' A data frame with 1000 rows and 4 columns:

**datetime** Date and time of measurement.

**condUncal** Raw uncalibrated conductivity recorded by STIC logger.

**tempC** Temperature recorded by STIC logger.

**SpC** Specific conductance calculated using 'apply\_calibration' function.

**Source**

AIMS project data.

---

calibration\_standard\_data

*Example calibration STIC lab data.*

---

**Description**

Example calibration data for STIC sensor for conversion from uncalibrated conductivity to specific conductivity ('SpC').

**Usage**

```
calibration_standard_data
```

**Format**

## 'calibration\_standard\_data' A data frame with 4 rows and 3 columns:

**sensor** Serial number for STIC sensor.

**standard** Specific conductance ('SpC') standard values used for soaking STIC.

**condUncal** Uncalibrated conductivity recorded by STIC when soaked in each standard.

**Source**

AIMS project data.

---

classified\_df

*Example classified STIC output data.*

---

**Description**

Classified STIC data used for function examples.

**Usage**

classified\_df

**Format**

## 'classified\_df' A data frame with 1000 rows and 5 columns:

**datetime** Date and time of measurement.

**condUncal** Raw uncalibrated conductivity recorded by STIC logger.

**tempC** Temperature recorded by STIC logger.

**SpC** Specific conductance calculated using 'apply\_calibration' function.

**wetdry** Classified STIC data created by 'classify\_wetdry' function.

**Source**

AIMS project data.

---

classify_wetdry	<i>classify_wetdry</i>
-----------------	------------------------

---

### Description

This is a function to classify STIC data into a binary "wet" and "dry" column. Data can be classified according to any classification variable defined by the user. User can choose one of two methods for classification: either an absolute numerical threshold or as a chosen percentage of the maximum value of the classification variable.

### Usage

```
classify_wetdry(stic_data, classify_var, threshold, method)
```

### Arguments

stic_data	A data frame with STIC data, such as that produced by <a href="#">apply_calibration</a> or <a href="#">tidy_hobo_data</a> .
classify_var	Name of the column in data frame you want to use for classification.
threshold	This is the user-defined threshold for determining wet versus dry based on the designated classification variable. If using the "absolute" method, the threshold will be a value in the same units as the designated classification variable. If using the "percent" method, the value will be a decimal percentage (range 0-1) of the max value of the classification variable in the data frame. Values above this proportion of the maximum will be designated as wet. If using the "y-intercept" method, this should be a model fit used to generate calibrated SpC values such as that produced by <a href="#">get_calibration</a> .
method	User chooses which classification method used to generate the binary data. "absolute" uses an absolute numerical threshold for classifying wet vs dry. "percent" uses a threshold based on a given percentage of the maximum value of the classification variable in the data frame. "y-intercept" uses the y-intercept from the <a href="#">get_calibration</a> function.

### Value

The same data frame as input, but with a new column called "wetdry".

### Examples

```
classified_df <-  
  classify_wetdry(calibrated_stic_data,  
    classify_var = "SpC", method = "absolute", threshold = 200  
  )  
head(classified_df)
```

---

field_obs	<i>Example field observations that could be compared to classified STIC data.</i>
-----------	---

---

### Description

Example field observations that could be compared to classified STIC data.

### Usage

```
field_obs
```

### Format

## 'field\_obs' A data frame with 5 rows and 3 columns:

**datetime** Date and time of field observation.

**wetdry** Field observation of stream water status ('wet' or 'dry').

**SpC** Field observations of specific conductance.

### Source

Made up data.

---

get_calibration	<i>get_calibration</i>
-----------------	------------------------

---

### Description

This is a function to fit specific conductivity (SpC) standards and uncalibrated conductivity measured by the STIC to a model object. This model can then be used to predict SpC values using [apply\\_calibration](#). As of right now, only linear models are supported.

### Usage

```
get_calibration(calibration_data)
```

### Arguments

calibration\_data  
STIC calibration data frame with columns "standard" and "condUncal".

### Value

A fitted lm model object relating SpC to the uncalibrated conductivity values measured by the STIC

**Examples**

```
head(calibration_standard_data)
lm_calibration <- get_calibration(calibration_standard_data)
summary(lm_calibration)
```

---

qaqc_stic_data	<i>qaqc_stic_data</i>
----------------	-----------------------

---

**Description**

This function provides multiple options for QAQC flagging of processed and classified STIC data frames, such as those generated by the [classify\\_wetdry](#) function. Users can select which operations are to be performed, and a single new QAQC column is created with all flags concatenated. QAQC options currently include: (1) correction and flagging of negative SPC values resulting from the calibration process, i.e., changing the negative values to 0 and flagging this (2) inspecting the wetdry classification time series for potential deviation anomalies based on user-defined windows

**Usage**

```
qaqc_stic_data(
  stic_data,
  spc_neg_correction = TRUE,
  inspect_deviation = TRUE,
  deviation_size = NULL,
  window_size = NULL
)
```

**Arguments**

<code>stic_data</code>	A data frame with classified STIC data, such as that produced by <code>classify_wetdry</code> .
<code>spc_neg_correction</code>	a logical argument indicating whether the user would like to correct negative SPC values resulting from the calibration process to 0. The character code associated with this correction is "C".
<code>inspect_deviation</code>	a logical argument indicating whether the user would like to identify deviation anomalies, in which a series of wet or dry readings less than or equal to 'deviation_size' in length is surrounded on both sides by 'window_size' or more observations of its opposite. This operation is meant to identify potentially suspect binary wet/dry data points for further examination. The character code associated with this operation is "D".
<code>deviation_size</code>	a numeric argument specifying the maximum size (i.e., number of observations) of a clustered group of points that can be flagged as an deviation
<code>window_size</code>	a numeric argument specifying the minimum size (i.e., number of observations) that the deviation must be surrounded by in order to be flagged

**Value**

The same data frame as input, but with new QAQC columns or a single, concatenated QAQC column. The QAQC output can include: "C", meaning the calibrated SpC value was negative from 'spc\_neg\_correction'; "D", meaning the point was identified as a deviation or deviation based on a moving window from 'inspect\_deviation'; or "O", meaning the calibrated SpC was outside the standard range based on the function `apply_calibration`.

**Examples**

```
qaqc_df <-
  qaqc_stic_data(classified_df,
    spc_neg_correction = TRUE,
    inspect_deviation = TRUE,
    deviation_size = 4, window_size = 96
  )
head(qaac_df)
```

---

<code>test_threshold</code>	<i>test_threshold.R</i>
-----------------------------	-------------------------

---

**Description**

This function is intended to allow the user to visually assess the effects of classification threshold uncertainty on STIC classification. It takes the the model object used to calibrate SpC, as well as a classified STIC data frame with column names matching those produced by [classify\\_wetdry](#).

**Usage**

```
test_threshold(stic_data, calibration)
```

**Arguments**

<code>stic_data</code>	classified STIC data frame with the variable names of that produced by <a href="#">classify_wetdry</a>
<code>calibration</code>	the model object used to calibrate SpC, generated by the <a href="#">get_calibration</a> function and used in <a href="#">apply_calibration</a>

**Value**

A time series plot of classified wet/dry observations through time using three different absolute classification thresholds: the y-intercept of the fitted model developed in `get_calibration`, the y-intercept plus one standard error, and the y-intercept minus one standard error

**Examples**

```
lm_calibration <- get_calibration(calibration_standard_data)
threshold_testing_plot <- test_threshold(stic_data = classified_df, calibration = lm_calibration)
```

---

tidy_hobo_data	<i>tidy_hobo_data</i>
----------------	-----------------------

---

**Description**

This function loads raw HOBO STIC CSV files and cleans up columns and headers

**Usage**

```
tidy_hobo_data(infile, outfile = FALSE, convert_utc = TRUE)
```

**Arguments**

infile	filename (including path or URL if needed) for a raw CSV file exported from HOBOWare.
outfile	filename (including path if needed) to save the tidied data frame. Defaults to FALSE, in which case tidied data will not be saved.
convert_utc	a logical argument indicating whether the user would like to convert from the time zone associated with their CSV to UTC

**Value**

a tidied data frame with the following column names: datetime, condUncal, tempC.

**Examples**

```
clean_data <-  
  tidy_hobo_data(  
    infile = "https://samzipper.com/data/raw_hobo_data.csv",  
    outfile = FALSE, convert_utc = TRUE  
  )  
head(clean_data)
```

---

tidy_stic_data	<i>Example tidied STIC output data.</i>
----------------	---

---

**Description**

Example tidied STIC data for input to calibration and classification process.

**Usage**

```
tidy_stic_data
```

**Format**

## 'tidy\_stic\_data' A data frame with 1000 rows and 3 columns:

**datetime** Date and time of measurement.

**condUncal** Raw uncalibrated conductivity recorded by STIC logger.

**tempC** Temperature recorded by STIC logger.

**Source**

AIMS project data.

---

trim_hobo_data	<i>trim_hobo_data</i>
----------------	-----------------------

---

**Description**

This function trims a tidied hobo data frame by datetime to eliminate periods where the logger was recording but not placed in the stream network

**Usage**

```
trim_hobo_data(
  stic_data,
  time_start = "2021-07-16 18:00:00",
  time_end = "2021-07-27 01:00:00"
)
```

**Arguments**

stic_data	A data frame with columns named condUncal and datetime, for example as produced by the function tidy_hobo_data.
time_start	User enters the time at which the logger was placed in the stream network
time_end	User enters the time at which the logger was removed from the stream network

**Value**

a tidied data frame with the same columns as the input, but trimmed to the user-defined time

**Examples**

```
trimmed_data <-
  trim_hobo_data(tidy_stic_data,
    time_start = "2021-07-16 18:00:00",
    time_end = "2021-07-27 01:00:00"
  )
head(trimmed_data)
```

---

validate_stic_data	<i>validate_stic_data.R</i>
--------------------	-----------------------------

---

## Description

This function takes a data frame with field observations of wet/dry status and SpC and generates both a confusion matrix for the wet/dry observations and a scatterplot comparing estimated SpC from the STICs to field-measured values.

## Usage

```
validate_stic_data(
  stic_data,
  field_observations,
  max_time_diff,
  join_cols,
  get_SpC,
  get_QAQC
)
```

## Arguments

<code>stic_data</code>	classified STIC data frame with the variable names of that produced by <a href="#">classify_wetdry</a> . At a minimum, there must be <code>datetime</code> , <code>condUncal</code> , and <code>wetdry</code> columns, and an <code>SpC</code> column if <code>get_SpC = T</code> .
<code>field_observations</code>	The input data frame of field observations must include a <code>datetime</code> column (in POSIXct format), as well as a column labeled <code>wetdry</code> consisting of the character strings “wet” or “dry” (as in the processed STIC data itself). Additionally, if field data on <code>SpC</code> was collected (e.g., with a sonde), this should be included as a third column called <code>SpC</code> , and units should be in $\mu\text{S}/\text{cm}$ .
<code>max_time_diff</code>	Maximum allowed time difference (in minutes) between field observation and STIC reading to be counted as a match.
<code>join_cols</code>	A named vector of columns that need to be matched between <code>stic_data</code> and <code>field_observations</code> in addition to <code>datetime</code> . This could include, for instance, a column specifying the site at which the observation was collected. Should be in the format of <code>c("col_name_in_stic_data" = "col_name_in_field_observations")</code> and can have as many columns as desired. If there are no additional columns to be matched, set to <code>NULL</code> .
<code>get_SpC</code>	Logical flag whether to get STIC data for <code>SpC</code> (T) or not (F). You must have an <code>SpC</code> column in <code>stic_data</code> and <code>field_observations</code> if this is used.
<code>get_QAQC</code>	Logical flag whether to get the STIC QAQC data (T) or not (F). You must have an <code>QAQC</code> column in both <code>stic_data</code> if this is used.

**Value**

The `field_observations` data frame with new columns indicating the closest-in-time STIC wetdry classification (`wetdry_STIC`), SpC measurement (`SpC_STIC`; only if `get_SpC = T`), and time difference between the field observation and STIC reading (`timediff_min`).

**Examples**

```
stic_validation <-  
  validate_stic_data(  
    stic_data = classified_df,  
    field_observations = field_obs,  
    max_time_diff = 30,  
    join_cols = NULL,  
    get_SpC = TRUE,  
    get_QAQC = FALSE  
  )
```

# Index

## \* datasets

- calibrated\_stic\_data, 3
- calibration\_standard\_data, 3
- classified\_df, 4
- field\_obs, 6
- tidy\_stic\_data, 9

apply\_calibration, 2, 5, 6, 8

- calibrated\_stic\_data, 3
- calibration\_standard\_data, 3
- classified\_df, 4
- classify\_wetdry, 5, 7, 8, 11

field\_obs, 6

get\_calibration, 2, 5, 6, 8

qaqc\_stic\_data, 7

- test\_threshold, 8
- tidy\_hobo\_data, 2, 5, 9
- tidy\_stic\_data, 9
- trim\_hobo\_data, 10

validate\_stic\_data, 11