NEON ALGORITHM THEORETICAL BASIS DOCUMENT (ATBD) – QA/QC PLAUSIBILITY TESTING

PREPARED BY	ORGANIZATION	DATE
Sarah Streett	TIS	12/19/2013
Jeff Taylor	TIS	01/22/2013
Cove Sturtevant	TIS	01/19/2017
Cove Sturtevant	TIS	03/31/2020

APPROVALS	ORGANIZATION	APPROVAL DATE
Kate Thibault	SCI	05/16/2022

RELEASED BY	ORGANIZATION	RELEASE DATE
Tanisha Waters	CM	05/16/2022

See configuration management system for approval history.

The National Ecological Observatory Network is a project solely funded by the National Science Foundation and managed under cooperative agreement by Battelle. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

Change Record

REVISION	DATE	ECO #	DESCRIPTION OF CHANGE
A	01/22/2013	ECO-00246	Initial release
B	01/19/2017	ECO-04423	Update of plausibility test language and algorithms to reflect current practice. Addition of flags for invalid calibration and science review.
C	05/06/2020	ECO-06422	Add suspect calibration test and remove science review flag (added to NEON.DOC.001113)
D	05/16/2022	ECO-06813	• Added NEON to document title • Minor formatting updates

TABLE OF CONTENTS

LIST OF TABLES AND FIGURES

1 DESCRIPTION

1.1 Purpose

This document specifies the plausibility algorithms as part of the automated Quality Control/Quality Assurance plan for TIS data [RD 03]. Specifically, this document describes the data flow, and the automated test routines for checking the plausibility of instrument observations. These plausibility tests will require site-specific, and often seasonally dependent, parameters for many of the Level 0 Data Products, i.e., realistic thresholds, some of which may initially rely on historical data available from public sources. This plausibility document includes several tests: range test, persistence test, step test, missing data tests, and checks for invalid and suspect calibrations, all of which will be used for most Data Products, while other Data Products may only use one or two. Finally, some basic statistical properties and example code are presented to demonstrate implementation of these tests.

1.2 Scope

These algorithms are intended to be applied automatically to the calibrated L0 data (raw) to determine quality control in producing Level 1 data products. These tests will be used to automatically examine data over a short timescale (*e.g.*, quasi-daily) and to determine the plausibility of each and every observation. The test quantities calculated in this document may be referenced by algorithms in other documents.

The data product algorithms can broadly be categorized as tests to check plausibility. Plausibility testing algorithms will focus on traditional range tests, stochastic tests, checking for non-responsive sensors, and missing data tests. Additionally, two tests verifying the veracity of the calibration information applied to the sensor data are described. Where possible, efficiency will be maximized through the use of combined tests. This set of automated algorithms and manual checks is not intended to be the sole mechanism of quality control for NEON data products.

2 RELATED DOCUMENTS, ACRONYMS AND VARIABLE NOMENCLATURE

2.1 Applicable Documents

AD[01]	NEON.DOC.000783	ATBD – De-Spiking and Time Series Analyses
AD[02]	NEON.DOC.000129	NEON Data Management Plan
AD[03]	NEON.DOC.000257	NEON Tier 3 CVAL requirements
AD[04]	NEON.DOC.001113	Quality Flags and Quality Metrics for TIS Data Products

2.2 Reference Documents

RD[01]	NEON.DOC.000243	NEON Glossary of Terms
RD[02]	NEON.DOC.000291	NEON Configured Sensor List
RD[03]	NEON.DOC.XXXXXX	TIS Data Quality Plan (modified NEON.FIU.011009.PLA.B)
RD[04]	NEON.DOC.000008	NEON Acronym List

2.3 Acronyms

Acronym	Explanation
ATBD	Algorithm Theoretical Basis Document
CI	NEON Cyberinfrastructure
CVAL	NEON Calibration and Validation Laboratory
DAS	Data Acquisition System
L0	Level 0: Raw sensor data at the native frequency
L0’	Level 0’: Calibrated sensor data at the native frequency
L1	Level 1: Sensor data with calibration and theoretical equations applied, aggregated over a specified interval
NOAA	National Oceanic Atmospheric Administration
QA/QC	Quality Assurance/Quality Control
STCDD	Sensor Type Configuration Definition (or Data) Document
USCRN	U.S. Climate Reference Network
WMO	World Meteorological Organization

2.4 Variable Nomenclature

The symbols used to display the various inputs in the ATBD, e.g., calibration coefficients and uncertainty estimates, were chosen so that the equations can be easily interpreted by the reader. However, the symbols provided will not always reflect NEON’s internal notation, which is relevant for NEON Cyber Infrastructure’s (CI) use, and or the notation that is used to present variables on NEON’s data portal. Therefore a lookup table is provided in order to distinguish what symbols specific variables can be tied to in the following document.

Symbol	Internal Notation	Description
Δ	diff	Difference
$abs(\ \ )$		Absolute value
QF		Quality flag
$QF_{CalVali}$		Quality flag for the Valid Calibration check
$QF_{CalSusp}$		Quality flag for the Suspect Calibration check
QF_{Final}		Final quality flag
QF_{Gap}		Quality flag for the Gap test
QF_{Null}		Quality flag for the Null test
QF_{Pers}	$QF_{Persistence}$	Quality flag for the Persistence test
QF_{Rng}	QF_{Range}	Quality flag for the Range test
QF_{Step}		Quality flag for the Step test
t		Time
Thsh		Threshold
Thsh_{Gap}		Threshold for the Gap test
Thsh_{Pers}		Threshold for the Persistence test
Time_{Pers}		Time parameter for the Persistence test
$\text{Thsh}_{Rng,min}$		Minimum threshold for the Range test
$\text{Thsh}_{Rng,max}$		Maximum threshold for the Range test
Thsh_{Step}		Threshold for the Step test
X		Time series of observations

3 DESCRIPTION

The plausibility tests and outputs are described in detail below.

3.1 QA/QC Test Definitions

3.1.1 Range Test

A Range test checks that every recorded observation falls within reasonable minimum and maximum values. Ideally, these min/max range limits are determined from historical climate data and previously observed instrument data. NEON will employ two sets of range limits. One set of limits will identify inconceivable events, the other will identify highly unlikely events. For example, if the temperature in Hawaii was observed to be -30 °C, the range test would flag this as implausible because this is significantly lower than the lowest value ever recorded in Hawaii (*i.e.* out of range). A value of -10 °C would be flagged as highly unlikely. Inconceivable events will be removed before the calculation of higher level data products, highly unlikely data will be flagged for further investigation. The determination of these thresholds is outlined in Section 4.1.

3.1.2 Persistence Test

Persistence tests check that there is a realistic fluctuation of values over a designated period of time. This test is designed to detect instruments that are "stuck" at a constant value. NEON's persistence test will consider runs of data that do not vary beyond the noise threshold of the measurement instrumentation. Runs that are longer than a specified threshold will be flagged. The development of this test is outlined in more detail in Section 4.1.

3.1.3 Step Test

A Step check is another test to find outliers in the data. Rather than checking for climatological outliers as with the Range test, the step test looks for unusual jumps in the data which would not necessarily be detected by the range test. A step check considers differences in subsequent data points and flags data when this difference is unrealistically large. This test requires equally spaced observations. More details are given in Section 4.1. Note that another test for unusual variation is provided by a "despiking" algorithm described in another document (AD[01]).

3.1.4 Missing Data

Missing data points are typically determined by a Null test in which the number of dropped data points over a given period of time is monitored. For example, a compromised connection between a sensor and a data logger would result in realistic data variation (*i.e.* pass the Step test) but have an increased number of dropped data points (*i.e.* fail the Null test) so the missing data tests would flag these data as implausible. A Gap test is used to check for a series of consecutive missing measurements.

3.1.5 Calibration Checks

The quality of data from instruments deployed at NEON field sites depends on accurate and current calibrations. There will be cases in which circumstances prevent a timely re-calibration interval. While the drift of many sensors may be minor for short periods outside the valid calibration range, the validity of some sensors is highly sensitive to regular calibration. Sensor data outside of the valid calibration range does not uphold NEON Tier 3 CVAL requirements (AD[03]), and therefore the data user will be notified by means of a "Valid Calibration" flag. There may also be cases in which a calibration failed to meet quality standards. Data for which such calibration information was applied will be marked with a "Suspect Calibration" flag. The implementation details of these flags are outlined in Section 4.1.

4 SCIENTIFIC CONTEXT

4.1 Theory of Algorithm

4.1.1 Determination of Plausibility Parameters

Specified parameters for plausibility test values will be constructed for all measurements based on calibrated raw measurement values (also known as L0' data). Prior knowledge of the measurement will inform plausibility parameters when available. Some of the NEON Level 0 Data Products have not been extensively observed before. In these cases, best possible estimates of appropriate test parameters will be constructed for initial plausibility tests and, after a sufficient amount of data has been collected, parameters will be updated based on NEON data.

It is recommended that plausibility parameters be determined using the sampling distributions of observations appropriate to the parameter in question. For example, the range test relies on checking extreme values, so it would be necessary to construct sampling distributions of observed minima and maxima for a given sample period of sufficiently similar data, e.g. historical data from the same or nearby site. For many variables, this will occur on seasonal, diel or semi-diel time-scales (*e.g.*, temperature, radiation, humidity). Once the appropriate sampling distribution is generated, the parameter can be determined from the tails of the distribution. Assuming a normal distribution (**Figure 1**), two or three standard deviations from the mean is commonly used as a threshold for flagging implausible data, signifying that 97.5% or 99.85% of values, respectively, are expected within the 1-tailed threshold, while the remaining 2.5% or 0.15%, respectively, will be flagged as outliers.

It is also possible that an extreme value may not be statistically and/or quantitatively defined for some quantities (*e.g.*, minimum wind speed, maximum wind direction). In these cases, acceptable plausibility parameters should be constructed based on physical limits (i.e. minimum wind speed cannot be less than zero). It should be noted that the selection of statistical thresholds is always a subjective decision – it may be that specific observations require a threshold determined in a unique way. If this is the case, it shall be explicitly detailed in the respective ATBD. The most up-to-date record of plausibility test parameters will be maintained by NEON's Cyber Infrastructure (CI). The schedule for regularly augmenting and updating this record will follow the standard protocol for managing TIS Data Quality (RD[03]).

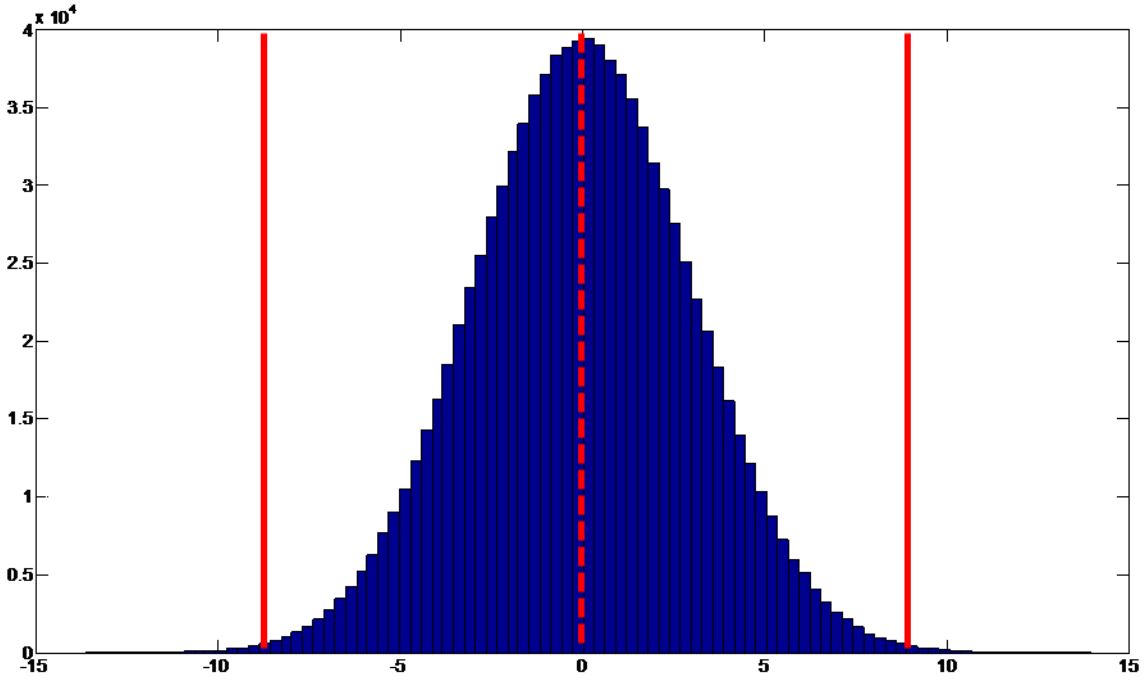

Figure 1. Histogram of simulated data following a Gaussian distribution with normalized mean 0 (dotted red line) and standard deviation of 3. The range of values that lie between 3 standard deviations of the mean (solid red lines) represent 99.7% of all the data.

4.1.2 Range Test

The goal of the range test is to ensure that the first release of NEON data either contains no inconceivable observations or at the very least they are quality flagged. A measurement fails the range test when it is less than the minimum range threshold or greater than the maximum range threshold:

$$QF_{Rng,t} = \begin{cases} 1 & X_t < \text{Thsh}_{Rng,min} \\ 1 & X_t > \text{Thsh}_{Rng,max} \\ 0 & \text{otherwise} \end{cases}$$

where $QF_{Rng,t}$ is the quality flag for the range test at time t, X_t is the observation at time t, and $\text{Thsh}_{Rng,min}$ and $\text{Thsh}_{Rng,max}$ are the respective minimum and maximum plausibility thresholds. The hard range test identifies inconceivable data values, whereas a soft range test identifies highly unlikely values. Hard and soft range tests are performed in exactly the same way, although they may result in different data outcomes, as specified in the ATBD (e.g. excluded from Level 1 (L1) data, or simply flagged). The thresholds for this test will initially be based on a combination of sensor range and world

records. Over time, these thresholds will be adjusted to be location and time specific. All of the plausibility test parameter definitions are summarized in Section 4.1.8.

4.1.3 Persistence Test

The role of the persistence test is to check for stuck instrumentation. In order to determine whether an instrument is "stuck", it is important to take under consideration the overall noise in the observations attributable to the measurement assembly. The persistence check will determine whether a consecutive series of observations fails to change by at least a stated threshold amount, Thsh_{Pers}, over a stated interval length, Time_{Pers}.

The test used by NEON will step through the observations, keeping track of the test's starting location as well as the value and location of the running minimum and maximum of the time series. The location of running maximum is only updated when an observation is greater than the current running maximum value. Likewise, the location of the running minimum is updated only when an observation is less than the current running minimum value. This ensures that in the event that the minimum or maximum value is repeated within a run the location remains at its earliest occurrence. When the difference between the running minimum and maximum exceeds Thsh_{Pers}, the length of the observations under test will be examined. If it is longer than Time_{Pers}, the data over the entire test interval, with the exception of the final value, will be flagged (i.e. $QF_{Pers} = 1$) and the process will start anew with the next observation. If the length of the current series is less than Time_{Pers}, the process will begin at either 1 past the location of the minimum, or 1 past the location of the maximum, whichever occurs earliest in the time series. This will insure that when a "stuck" instrument is encountered, all of the relevant observations will be flagged. The threshold change for the persistence test, Thsh_{Pers}, will be based upon the measurement noise of all involved instruments as provided by CVAL. The interval length, Time_{Pers}, will be based on the frequency of measurement. Note that this persistence check does not rely on equally spaced observations and is unaffected by missing data. Example code to perform this test is included in Section 5.2.

Figures 2 and **3** show two sequential persistence test runs. For purposes of illustration, we assume that the noise range, Thsh_{Pers}, is the distance between the gray dashed lines and that the test fails if an interval of longer than twelve points (Time_{Pers}) within the noise range is encountered. The test begins with the first data point and marks it as the current minimum and maximum (denoted with blue and red overbars, respectively). The cumulative minimum is updated for each of the next two data points and then again with the fifth data point. The cumulative maximum does not change until the seventh observation. As the test steps through each data point, it checks to see whether the distance between the cumulative maximum and minimum is greater than Thsh_{Pers},. For this first run, the threshold is exceeded by the ninth observation. Since this time interval is less than Time_{Pers} (e.g., twelve), no observations are flagged. The next run will begin with the observation following the cumulative minimum from the previous run (**Figure 3**). Note that the number of observations examined by each run

varies, the first run considers only the first nine observations whereas the second run looks at observations 6-30.

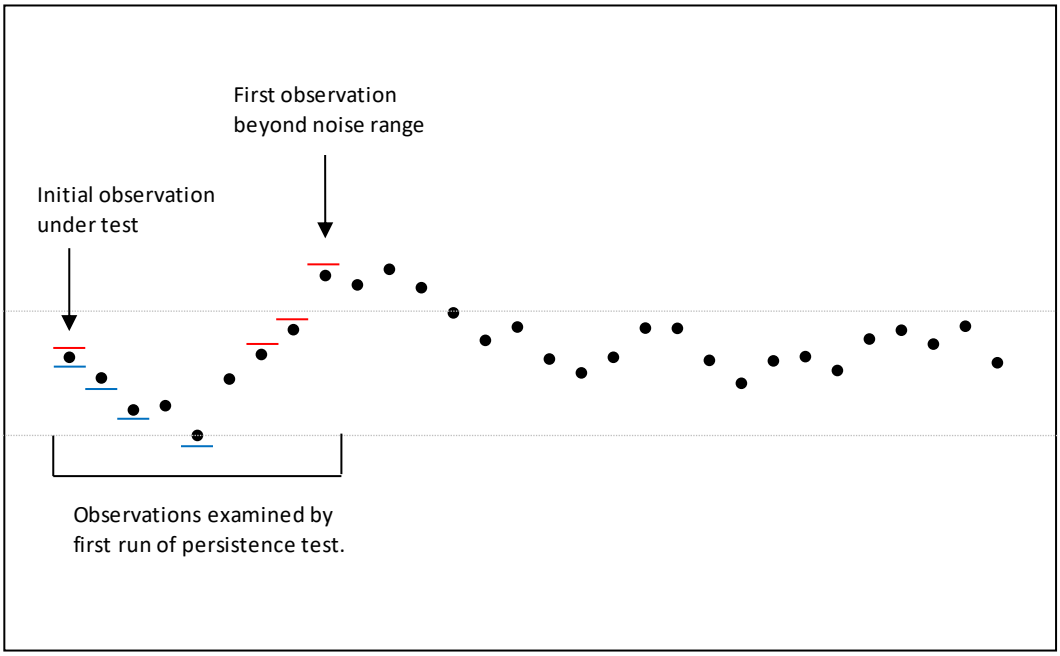

Figure 2. First run of the Persistence check. The running min/max are denoted by a blue/red bar, respectively. The gray dashed line indicates the instrument noise range and is shown for the overall minimum of the first run for convenience.

The persistence test fails during the second run when the test encounters more than twelve observations falling within the noise range. The test continues to flag data points until an observation outside of the range is encountered. Note that although the cumulative minimum and maximum are updated throughout the run, their location is not updated when an identical value is recorded (e.g., observation 22). When a noise sequence has been flagged, the next persistence run begins with the first observation following the noise sequence (e.g., observation 30 in **Figure 3**).

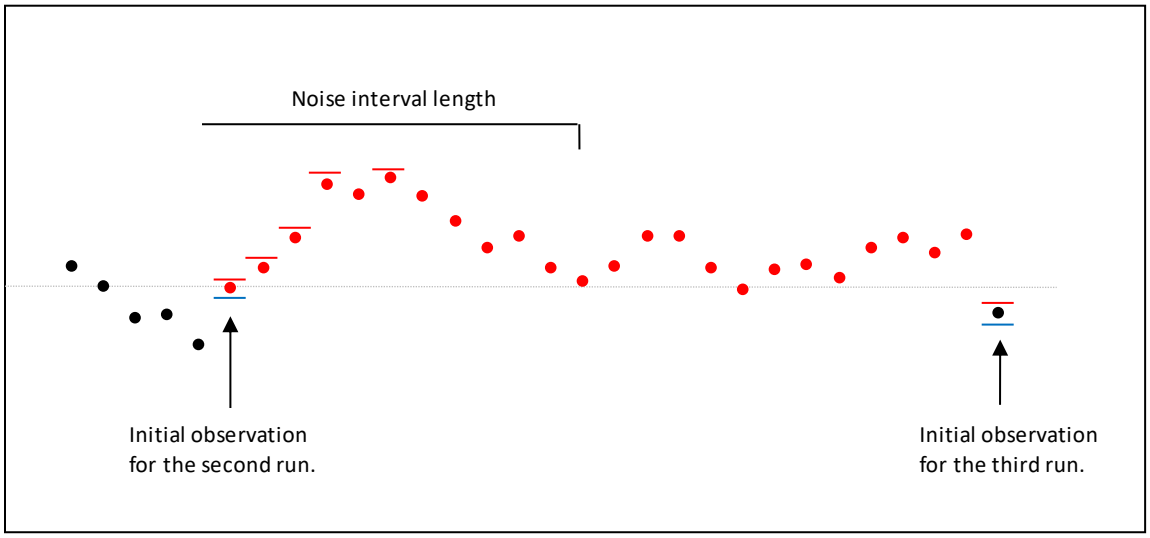

Figure 3. Second run of the persistence check. The running min/max are denoted by a blue/red bar, respectively. The gray, dashed line indicates the instrument noise range and is shown for the overall minimum of the second run for convenience. The red points are flagged (i.e. $\boldsymbol{QF_{Pers} = 1}$) by the Persistence test.

4.1.4 Step Test

The Step test utilizes the difference between subsequent observations to check for implausibly large jumps in the data. Initially the step threshold, Thsh_{Step}, will be set to a relevant value (typically 4-7 times the median absolute deviation of subsequent observations) based on nearby historical data if possible. Else, a conservatively large value will be chosen until enough NEON data is collected to set a reasonable threshold. Over time, this threshold will be adjusted to be both location and time dependent. The Step test is performed by conducting a lag-1 difference operation on the observation time series:

$$\Delta X_t = X_t - X_{t-1}$$

where X_t is the calibrated observation at time t. When the absolute value of ΔX_t exceeds the threshold, both X_t and X_{t-1} fail the Step test. Otherwise, only the value at time t is set to 0:

If $abs(\Delta X_t) > \mathrm{Thsh}_{Step}$, then

$$QF_{Step,t} = QF_{Step,t-1} = 1$$

Else,

$$QF_{Step,t} = 0$$

4.1.5 Null and Gap Tests

The Null test and Gap tests are used to monitor the loss of data. The exact threshold for acceptable data loss will vary with instrument and sampling interval and, in some cases, may simply be defined as an arbitrary number (e.g. 0 or 1 maximum missing data values per day) or by a local calibration cycle. The Null test fails when the sensor or data acquisition system (DAS) report "no data" when a measurement was expected:

$$QF_{Null,t} = \begin{cases} 1 & \text{Sensor or DAS report "no data" when a measurement at time } t \text{ is expected} \\ 0 & \text{otherwise} \end{cases}$$

A Gap test is used to explicitly check for a prolonged period of missing data. The Gap test fails when the current measurement lies within a set of consecutive expected but missing values, and the number of consecutive missing values is equal to or greater than a chosen threshold, Thsh_{Gap}:

$$QF_{Gap,t} = \begin{cases} 1 & \text{when the current measurement at time } t \text{ is missing, and is part of a consecutive set of missing measurements numbering } \mathrm{Thsh}_{Gap} \text{ or greater} \\ 0 & \text{otherwise} \end{cases}$$

For data that is sampled at higher frequencies, the statistical approach that has been used to define all plausibility thresholds should continue to be applied in the case of the Gap test, and should be defined under optimal sampling conditions.

4.1.6 Valid Calibration Check

Most sensors have a valid date range associated with the calibration of the sensor. This valid range is included in the calibration XML file that CI ingests and applies to the L0 data. All data streams requiring a calibration will be quality flagged when the date of measurement is outside the valid calibration date range. Computation of this Valid Calibration flag, $QF_{CalVali}$, is as follows:

$$QF_{CalVali,t} = \begin{cases} 1 & \text{If the timestamp of the current sensor measurement at time } t \text{ is outside the valid calibration date range for the sensor} \end{cases}$$

0 Otherwise

Data products that combine the output from multiple calibrated sensors will use a single Valid Calibration flag to indicate when data from any of the sensors used calibration information that was outside the valid date range. The ATBD for each data product will specify which data streams are to be checked for valid calibration and any other data product-specific details for calculating the Valid Calibration flag.

4.1.7 Suspect Calibration Check

A calibration may yield results that do not meet performance requirements as a function of the field calibration process or retroactively discovered for a calibration where a sensor has or continues to stream suspect data. When such a suspect calibration is identified, the CVALR1 coefficient in the calibration XML file is set to 1. Any data stream requiring a calibration will be quality flagged when its CVALR1 coefficient equals 1. Computation of this Suspect Calibration flag, $QF_{CalSusp}$, is as follows:

$$QF_{CalSusp,t} = \begin{cases} 1 & \text{If the CVALR1 coefficient in the calibration XML file that applies to the sensor measurement at time } t \text{ is present and equal to a value of 1} \\ 0 & \text{Otherwise} \end{cases}$$

Note that some calibration XML files may not include the CVALR1 coefficient. In that case, the calibration is assumed to meet performance requirements. Data products that combine the output from multiple calibrated sensors will use a single Suspect Calibration flag to indicate when data from any of the sensors failed to meet performance requirements. The ATBD for each data product will specify which data streams are to be checked for a suspect calibration and any other data product-specific details for calculating the Suspect Calibration flag.

4.1.8 Plausibility Test Parameter Definitions

Table 1 summarizes the plausibility test parameters defined in this document:

Table 1. Summary of plausibility test parameters.

Parameter	Plausibility Test	Definition	Underlying statistical quantity for parameter determination
$\mathrm{Thsh}_{Rng,min}$	Range	Minimum acceptable measurement value	Extreme low values
$\mathrm{Thsh}_{Rng,max}$	Range	Maximum acceptable measurement value	Extreme high values
Time_{Pers}	Persistence	Acceptable time interval for consecutive measurement values to vary no more than Thsh_{Pers}	Sampling frequency
Thsh_{Pers}	Persistence	Minimum acceptable range of measurement variation over the time interval Time_{Pers}	Measurement noise
Thsh_{Step}	Step	Maximum acceptable difference between measurement pairs	Differences of subsequent measurement pairs
Thsh_{Gap}	Gap	Minimum number of missing measurements that constitute a data gap	Large gaps of missing data

5 ALGORITHM IMPLEMENTAION

5.1 Algorithm Framework

Once all of the plausibility test parameters have been defined, the tests are implemented in sequence for each applicable observation at each site (as specified in the ATBD). This is an automated testing procedure in which individual calibrated data streams (i.e., after applying applicable calibration coefficients) are checked prior to any other data manipulation. **Figure 4** shows an example process for this procedure.

The sequence shown here need not be followed in every case. Different observations and data products will require different sequences of tests that shall be detailed in the relevant ATBD. In the interest of computational efficiency, it may be sufficient to stop subsequent quality tests for data that has already been flagged. For example, if a measurement has failed the Null test, there is no need to perform further quality testing. All of these details can be found in the QA/QC section of the specific data product ATBD.

Note that the Valid Calibration Flag, if applicable, should always be included in the computation of α and β quality metrics and the Final Quality Flag (AD[04]). Although some sensors are less sensitive to a regular calibration, the sensor-specific calibration interval should be adjusted rather than making a case-by-case decision to incorporate the Valid Calibration Flag into the Final Quality Flag.

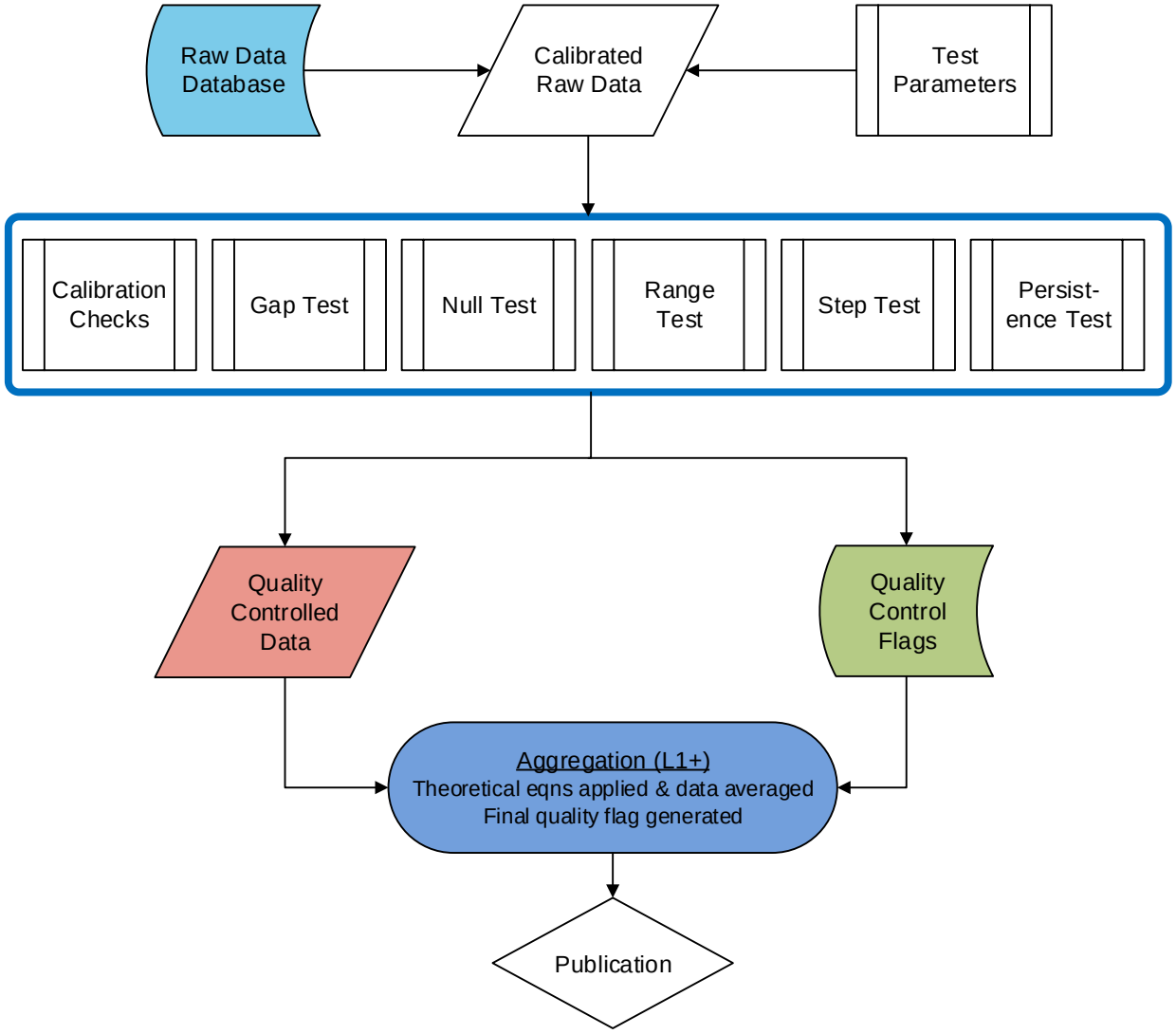

Figure 4. Data flow diagram for automated plausibility testing. The sequence of these tests need not be in the linear configuration shown here. Relevant ATBDs will define the actual process. See AD[04] for generation of the final quality flag.

5.2 Sample code

The following R programming code demonstrates the implementation of the Range, Step, Persistence, Null, and Gap tests. All user comments are preceded by the "#" symbol.

```
##############################################################################

#' @title Plausibility tests (Range, Step, Persistence, Null, Gap)
```

#' @author

#' Cove Sturtevant \email{csturtevant@neoninc.org}

#' @description

#' Function definition. Determines inplausible data indices based on user-specified limits for the data range, step between adjacent values, persistence (similarity of adjacent values), nulls, and gaps.

#' @param \code{data} Required input. A data frame containing the data to be evaluated (do not include the time stamp vector here).

#' @param \code{ts} Optional. A time vector of class POSIXlt of times corresponding with each row in data. Defaults to an evenly spaced time vector starting from system time of execution by seconds.

#' @param \code{RngMin} Optional. A numeric vector of length equal to number of variables in data containing the minimum acceptable value for each variable. Defaults to observed minimums (no flags will result)

#' @param \code{RngMax} Optional. A numeric vector of length equal to number of variables in data containing the maximum acceptable value for each variable. Defaults to observed maximums (no flags will result)

#' @param \code{DiffStepMax} Optional. A numeric vector of length equal to number of variables in data containing the maximum acceptable absolute difference between sequential data points for each variable. Defaults to observed maximum (no flags will result)

#' @param \code{DiffPersMin} Optional. A numeric vector of length equal to number of variables in data containing the minimum absolute change in value over the interval specified in TintPers to indicate the sensor is not "stuck". Defaults to a vector of zeros (no flags will result).

#' @param \code{TintPers} Optional. A difftime object of length equal to number of variables in data specifying the time interval for each variable over which to test for the minimum absolute change in value specified in DiffPersMin. Defaults to 60 x median observed time difference. Class difftime can be generated using as.difftime.

#' @param \code{TestNull} Optional. Apply the null test? A logical vector of [TRUE or FALSE] of length equal to number of variables in data. Defaults to FALSE (no null values are flagged)

#' @param \code{NumGap} Optional. A numeric value >= 1, interpretable as an integer, specifying the numer of consecutive NA values constituting a gap. Default is the one more than the length of the data series (no gaps will be flagged)

#' @return A list of flags giving the failed and NA positions for each the Range, Step, Persistence, Null, and Gap tests. Each flag is itself a nested list of failed and na (unable to eval) flagged indices for each variable in data.

#' @references

#' NEON Algorithm Theoretical Basis Document QA/QC Plausibility Testing (NEON.DOC.011081)

#' license: Terms of use of the NEON FIU algorithm repository dated 2015-01-16

#'

#' @keywords NEON QAQC, plausibility, range, step, persistence, null, gap

#' @examples Currently none

#' @seealso Currently none

#' @export

changelog and author contributions / copyrights

Cove Sturtevant (2015-12-30)

original creation

Cove Sturtevant (2015-01-06)

update to include indices where tests unable to be evaluated, and corrected some

variable names to conform to EC-TES naming convention

Cove Sturtevant (2015-02-09)

fixed bug in persistence test computation causing large blocks of NA to fail

Cove Sturtevant (2015-02-26)

adjusted gap test to reflect current NEON practice - not based on time difference

between measurements, but rather consecutive number of NA values

```
#    adjusted header to conform with eddy4R coding convention

################################################################################


def.plau <- function (

 data,                    # a data frame containing the data to be evaluated (do not include the time
     stamp vector here). Required input.

 ts = as.POSIXlt(seq.POSIXt(from=Sys.time(),by="sec",length.out=length(data[,1]))),  # time vector
     corresponding with the rows in data, in  Class "POSIXlt", which is a named list of vectors
     representing sec, min, hour,day,mon,year. Defaults to an evenly spaced time vector starting from
     execution by seconds.

 RngMin = apply(data,2,min,na.rm=TRUE), # a numeric vector containing the minimum acceptable value
     for each variable in data, defaults to observed minimums

 RngMax = apply(data,2,max,na.rm=TRUE), # a numeric vector containing the maximum acceptable
     value for each variable in data, defaults to observed maximums

 DiffStepMax = apply(abs(apply(data,2,diff)),2,max,na.rm=TRUE), # a vector containing the maximum
     acceptable absolute difference between sequential data points for each variable in data

 DiffPersMin = rep.int(0,length(data)), # a vector containing the minimum absolute change in value for
     each variable in data over the interval specified in TintPers. Defaults to a vector of zeros.

 TintPers = 60*median(abs(diff(ts)),na.rm=TRUE)*rep.int(1,length(data)), # a vector of class difftime
     specifying the time interval for each variable in data over which to test for the minimum absolute
     change in value specified in DiffPersMin. Defaults to 60 x median observed time difference. Class
     difftime can be generated using as.difftime.

 TestNull = rep(FALSE,length(data)), # apply the null test? A logical vector of [TRUE or FALSE] of length
     equal to number of variables in data. Defaults to FALSE (no null values are flagged)

 NumGap = length(data[,1])+1 # an integer greater than 0 specifying the number of consecutive NA
     values that constitute a gap

) {



# Error Checking ----------------------------------------------------------


 # Check data
```

```
if(missing("data") | !is.data.frame(data)){
  stop("Required input 'data' must be a data frame")
}

# Check ts
ts <- try(as.POSIXlt(ts),silent=TRUE)
numData <- length(data[,1])
if(class(ts)[1] == "try-error"){
  stop("Input variable ts must be of class POSIXlt")
} else if (length(ts) != numData){
  stop("Length of input variable ts must be equal to length of data.")
}

# Check RngMin & RngMax
if((!is.numeric(RngMin)) | (!is.numeric(RngMax))){
  stop("Input parameters RngMin and RngMax must be numeric vectors.")
} else if ((length(RngMin) != length(data)) | (length(RngMax) != length(data))){
  warning("Length of input parameters RngMin or RngMax not equal to number of data variables. Using
      first element of each for all variables.")
  RngMin <- rep(RngMin[1],length(data))
  RngMax <- rep(RngMax[1],length(data))
}

# Check DiffStepMax
if(!is.numeric(DiffStepMax)){
  stop("Input parameter DiffStepMax must be a numeric vector.")
} else if (length(DiffStepMax) != length(data)){
```

```
  warning("Length of input parameter DiffStepMax not equal to number of data variables. Using first
     element of DiffStepMax for all variables.")

  DiffStepMax <- rep(DiffStepMax[1],length(data))

}


# Check DiffPersMin

if((!is.numeric(DiffPersMin))){

  stop("Input parameter DiffPersMin must be a numeric vector.")

} else if (length(DiffPersMin) != length(data)){

  warning("Length of input parameter DiffPersMin not equal to number of data variables. Using first
     element of DiffPersMin for all variables.")

  DiffPersMin <- rep(DiffPersMin[1],length(data))

}


# Check TintPers

TintPers <- try(as.difftime(TintPers),silent=TRUE)

if(class(TintPers) == "try-error"){

  stop("Input parameter TintPers must be a difftime object")

} else if (length(TintPers) != length(data)){

  warning("Length of input parameter TintPers not equal to number of data variables. Using first
     element of TintPers for all variables.")

  TintPers <- TintPers[1]*rep.int(1,length(data))

}


# Check TestNull

if(!is.logical(TestNull)){

  stop("Input parameter TestNull must be a logical vector.")

} else if (length(TestNull) != length(data)){
```

```
  warning("Length of input parameter TestNull not equal to number of data variables. Using first
     element of TestNull for all variables.")

  TestNull <- rep(TestNull[1],length(data))

 }


 # Check NumGap

 if (length(NumGap) != length(data)) {

  warning("Length of input parameter NumGap not equal to number of data variables. Using first
     element of NumGap for all variables.")

  NumGap <- rep(NumGap[1],length(data))

 }

 if (!is.numeric(NumGap)) {

    stop("Input parameter NumGap must be a numeric vector.")

  } else if(length(which(NumGap < 1)) > 0) {

    warning("Elements of input parameter NumGap must be integers >= 1, setting values < 1 to 1.")

    NumGap[which(NumGap < 1)] <- 1

  } else if(length(which(NumGap-floor(NumGap) > 0)) > 0) {

    warning("Some or all elements of input parameter NumGap are not integers, these will be rounded
     toward zero.")

    NumGap <- floor(NumGap)

 }


# Perform QAQC tests ------------------------------------------------

  nameData <- names(data) # Get variable names


  # Do range test
```

```
posFlagRng <- list(fail=as.list(data),na=as.list(data)) # initialize null test output

for(idxVar in 1:length(data)) {

 posFlagRng$fail[[idxVar]] <- which((data[,idxVar] < RngMin[idxVar]) | (data[,idxVar] >
    RngMax[idxVar]))

 posFlagRng$na[[idxVar]] <- which(is.na(data[,idxVar]))

}


# Do step test

posFlagStep <- list(fail=as.list(data),na=as.list(data)) # initialize step test output

for(idxVar in 1:length(data)) {

 posFlagStep$fail[[idxVar]] <- which((abs(diff(data[,idxVar])) > DiffStepMax[idxVar]))

 posFlagStep$fail[[idxVar]] <- unique(c(posFlagStep$fail[[idxVar]],posFlagStep$fail[[idxVar]]+1))

 posFlagStep$na[[idxVar]] <- which(is.na(diff(data[,idxVar])))+1

}


# Do persistence test

posFlagPers <- list(fail=as.list(data),na=as.list(data)) # initialize persistence test output

for(idxVar in 1:length(data)) {


 # Let users know persistence test may take some time

 if (DiffPersMin[idxVar] > 0) {

  print(paste0("Running persistence test for variable ",nameData[idxVar],". This may take some
    time..."))

 }


 posFlagPers$fail[[idxVar]] <- numeric(length=0) # Initialize output

 posFlagPers$na[[idxVar]] <- which(is.na(data[,idxVar])) # Initialize output
```

```
idxDataSt <- 1 # initialize starting index


# Make sure we aren't on a null value
while(is.na(data[idxDataSt,idxVar])){
  idxDataSt <- idxDataSt+1
}
idxDataMin <- idxDataSt # initialize index of running min
idxDataMax <- idxDataSt # intialize index of running max
idxData <- 2 # intialize index position


while((idxData <= numData) & (DiffPersMin[idxVar] > 0)) {


  # Is this a null value?
  if(is.na(data[idxData,idxVar])){
    idxData <- idxData+1
    next
  }


  # Is the value at this index the running max or min?
  if(data[idxData,idxVar] < data[idxDataMin,idxVar]){
    idxDataMin <- idxData
  } else if(data[idxData,idxVar] > data[idxDataMax,idxVar]){
    idxDataMax <- idxData
  }


  # Is diff between max and min at or larger than the persistence threshold
```

```
if(data[idxDataMax,idxVar]-data[idxDataMin,idxVar] >= DiffPersMin[idxVar]) {

 # We've hit the threshold, now check wether we are beyond the allowable time interval

 if(ts[idxData]-ts[idxDataSt] <= TintPers[idxVar]) {

  # Hooray! The data is not "stuck"

  idxDataSt <- min(c(idxDataMin,idxDataMax))+1 # set start of next window to the next point after
 the lower of the running min and max


  # Make sure we aren't on a null value

  while(is.na(data[idxDataSt,idxVar])) {

   idxDataSt <- idxDataSt+1

  }


  idxDataMin <- idxDataSt # reset running minimum

  idxDataMax <- idxDataSt # reset running maximum

  idxData <- idxDataSt+1 # reset the next point to be evaluated


 } else {


  # We might have a stuck sensor, but first let's check whether we blew the time threshold b/c

  # all the data were NA prior to this point

  if (sum(!is.na(data[idxDataSt:(idxData-1),idxVar])) <= 1) {


   # Data were all NA after starting index, mark as cannot evaluate

   posFlagPers$na[[idxVar]] <- union(posFlagPers$na[[idxVar]],idxDataSt:(idxData-1))


  } else {
```

```
      # Awe bummer, the sensor was stuck before this point.
      posFlagPers$fail[[idxVar]] <- unique(c(posFlagPers$fail[[idxVar]],idxDataSt:(idxData-1)))
    }

    idxDataSt <- idxData # restart the test from here
    idxData <- idxDataSt+1 # reset the next point to be evaluated
  }

} else if ((idxData == numData) & (ts[idxData]-ts[idxDataSt] > TintPers[idxVar])) {

  # We didn't hit the threshold and we've reached the end of the data. We are also beyond the
    allowable
  # time interval for the persistence test, so let's flag the data
  posFlagPers$fail[[idxVar]] <- unique(c(posFlagPers$fail[[idxVar]],idxDataSt:idxData))

  idxData <- idxData+1 # We're done

} else {

  # We didn't pass the minimum acceptable change on this point, move to the next
  idxData <- idxData+1
 }
}
# If we reached the end of the data but the last value was NA, we need to go back and evaluate the
    last
# non-NA value
```

```
  idxData <- numData
  if (is.na(data[idxData,idxVar])) {
    # Get to last non-NA point
    while(is.na(data[idxData,idxVar])) {
      idxData <- idxData-1
    }

    if (ts[idxData]-ts[idxDataSt] > TintPers[idxVar]) {
      # We didn't hit the threshold for the final non-NA points and we were beyond the allowable
      # time interval for the persistence test, so let's flag the end of the data
      posFlagPers$fail[[idxVar]] <- unique(c(posFlagPers$fail[[idxVar]],idxDataSt:idxData))
    } else
      # We didn't hit the threshold for the final non-NA points, but we are not yet beyond the
      # allowable time interval, so let's flag as unable to evaluate
      posFlagPers$na[[idxVar]] <- unique(c(posFlagPers$na[[idxVar]],idxDataSt:idxData))

  }
}

# Do Null test
posFlagNull <- list(fail=as.list(data),na=as.list(data)) # initialize null test output
for(idxVar in 1:length(data)) {
  posFlagNull$fail[[idxVar]] <- numeric(length=0)
  posFlagNull$na[[idxVar]] <- numeric(length=0) # there is never an instance where we cannot evaluate
      the null test
  if(TestNull[idxVar]) {
    posFlagNull$fail[[idxVar]] <- which(is.na(data[,idxVar]))
```

```
 }
}

# Do Gap test
posFlagGap <- list(fail=as.list(data),na=as.list(data)) # initialize gap test output
for(idxVar in 1:length(data)) {
 posFlagGap$na[[idxVar]] <- numeric(length=0) # there is never an instance where we cannot evaluate
    the gap test

 posNull <- which(is.na(data[,idxVar])) # find NA values
 posGap <- posNull # Start out thinking every Null is a gap, we'll whittle it down below
 diffPosNull <- diff(posNull) # difference between null position

 # Only evaluate this if we need to
 if (length(posNull) >= NumGap[idxVar]) {

  # Go thru each NA value to determine if it is part of a set >= NumGap[idxVar]
  for (idxNull in 1:length(posNull)) {

   # Isolate positions within the NumGap[idxVar] range that are consecutive
   diffPosNullSelf <- c(diffPosNull[1:idxNull-1],1,diffPosNull[idxNull:length(diffPosNull)])# Fill in the
    position difference vector with a value of 1 for idxNull itself
   posNullPre <- seq(from=idxNull-NumGap[idxVar],to=idxNull-1,by=1)
   posNullPre <- rev(posNullPre[(posNullPre > 0) & (posNullPre <= numData)])
   numNullPre <- which(diffPosNullSelf[posNullPre]!=1)[1]-1 # number of consecutive nulls prior to
    this Null
   if (is.na(numNullPre)) {
    numNullPre <- length(posNullPre)
```

```
    }
    posNullPost <- seq(from=idxNull,to=idxNull+NumGap[idxVar]-1,by=1)
    posNullPost <- posNullPost[(posNullPost > 0) & (posNullPost <= numData)]
    numNullPost <- which(diffPosNullSelf[posNullPost]!=1)[1]-1  # number of consecutive nulls including
     and after this Null
    if (is.na(numNullPost)) {
     numNullPost <- length(posNullPost)
    }

    if (numNullPre+numNullPost < NumGap[idxVar]) {
     # This position is not within a gap, so remove it from out list
     posGap <- setdiff(posGap,posNull[idxNull])
    }
   }

   posFlagGap$fail[[idxVar]] <- posGap

  } else {
   posFlagGap$fail[[idxVar]] <- numeric(length=0) # No gaps
  }

}

# Return results
result <- list(
 posFlagRng = posFlagRng,
 posFlagStep = posFlagStep,
```

```
    posFlagPers = posFlagPers,
    posFlagNull = posFlagNull,
    posFlagGap = posFlagGap)

  return(result)

}
```

6 UNCERTAINTY

While the metrics calculated here will be linked to the overall uncertainty estimations as described in the TIS Data Quality Plan (RD[03]), there are no associated uncertainty estimations for these metrics. These data quality metrics are intended to provide high-level, quantitative scrutiny for the performance of sensors rather than provide direct confidence estimates for the data products themselves. Individual data product ATBDs should inform how the uncertainty parameters are calculated.

7 VALIDATION AND VERIFICATION

7.1 Algorithm Validation

Verification and validation of algorithm implementation shall be managed by DPS, in consultation with FIU and CI in accordance with TIS objectives. Additional process detail may be found in the NEON Data Management Plan (AD[02]).

It is anticipated that testing will be done through construction of a "unit test harness", where trusted input data is used in tests of individual functional components of the algorithm against a set of expected test outputs (i.e., test against a golden data set). Test data sets will be generated synthetically to simulate plausibility failure cases in which each of the aforementioned plausibility tests can be verified (i.e., test against a tarnished data set). The exact use cases for which plausibility tests will pass/fail are highly data-product specific and, for this reason, care must be taken to ensure that the application of these tests is documented in each data product's respective ATBD.

8 SCIENTIFIC AND EDUCATIONAL APPLICATIONS

Upon completion of all plausibility testing for a given set of observations at a given location, all of the data and associated quality information shall be made available for the next sequence of automated quality control testing as defined in the TIS Data Quality Plan [RD 03]. Records of the flagged data should be detailed in the quality control flag report for later consideration and general statistics of the data flags should be output for regular scrutiny.

9 FUTURE PLANS AND MODIFICATIONS

9.1 Valid Calibration Check with Associated DAS

The custom Data Acquisition Systems (DAS) used by NEON use a precision resistor to measure the voltage output from analog sensors. Regular DAS calibration for analog signals is just as important as the sensor calibration, thus each DAS is also subject to a required calibration interval. CVAL already tracks the valid DAS calibration date range in an XML file. In the future, the Calibration quality flag for each analog sensor stream will also reflect whether the associated DAS (measuring the analog signal) is out of calibration. This will become possible once CI develops the capability to ingest DAS calibration XML files and associate a specific DAS with a specific data stream.

9.2 Updates of Test Parameters

It will be necessary to maintain regularly updated and version-controlled records of all test parameters associated with plausibility testing as they will vary by measurement subsystem, site location, and possibly by time. All of these parameters will ultimately be calculated and stored by CI according to the TIS Data Quality Plan [RD03]. Over time, measurement thresholds will be adjusted to be dependent on e.g. the time of year and the site location. To illustrate the necessity of time dependent parameters for certain variables and tests, hourly temperature data for the years 1983-2012 at a location near NEON's Sterling site are plotted by month in **Figure 5**. The outlier in July would not be discovered if the range limits were formulated from data encompassing all seasons.

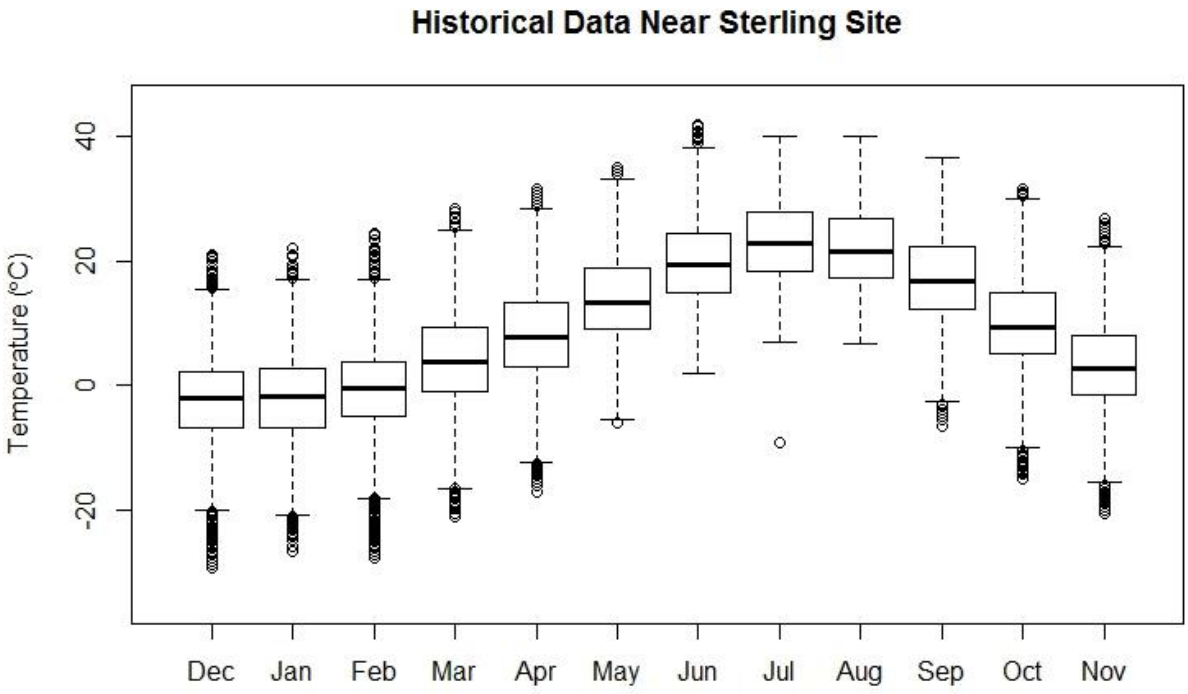

Figure 5. Boxplots of hourly temperature by month for the years 1983-2012 at a location near Sterling, CO.