Manitoba Métis Federation Weather Data Codebook

MANITOBA METIS FEDERATION AND
CENTRE FOR EARTH OBSERVATION SCIENCE

Document Control

0.1 Version History

Version	Author(s)	Type	Date Modified	Comments
1.0	Friesen, K.	Working Copy	2022-02-24	Final drafted version.
1.1	Hepner, K.	Working Copy	2023-06-12	Updated script description and standardized variable names.

0.2 Document Location

A hard copy of the document can be found in the Lab 489 document cupboard.

A digital copy of this document can be found here:

0.3 License

Manitoba Métis Federation(2021). Weather Data Codebook, Version 1.1. Manitoba Metis Federation and
Centre for Earth Observation Science. University of Manitoba.

Contents

1 Weather Station Data

1.1 Description

This codebook describes the **MMF_weather_QA-compiled.R** script that consecutively standardizes variable header names, identifies error prone data with an added metadata field, and compiles separate weather data files. Raw weather data if first downloaded from the DataGarrison site, which may occur when a station logger box has been reset, hence the process of cleaning the data could be dictated by the event that caused the reset of the loggers at the weather station site(s). Review *Weather Keeper Cookbook* to determine if you need to download from the site.

Variables

Raw weather station data is at Level 0, defined on page 11.

Table 1.1: Raw Weather Station Variables.

Number	Variable
1	Date_Time
2	Pressure_20812842_mbar
3	Temperature_21048243_deg_C
4	PAR_21040570_uE
5	Temperature_21048243_deg_C
6	RH_21048243_
7	Rain_21050258_mm
8	Wind Speed_21055169_m/s
9	Gust Speed_21055169_m/s
10	Wind Direction_21055169_deg
11	Backup Batteries_20827695_V

1.2 Data processing

This script standardizes the names of the variables (Table 1.1) to CF (primarily) or BODC equivalents and adds a metadata field to identify measurements outside the calibrated range of each variable (or sensor) by assigning it a code. These standardized weather data file(s) are then compiled into a singe file. This process is repeated every time a dataset is archived on the DataGarrison site. Archiving occurs when:

- Automatic reboot from sensor failure.
- Manual reboot done by user on Data Garrison.
- Manual set sensor alarm limit, that produces an automatic reboot.

All of these events will send you an alert via email if the preference section is set correctly to your email.

Variables

The standardized and QA metadata variables now positions the data at Level 1, defined on page 11.

Table 1.3: QA Weather Station Variables.

Number	Variable
1	Date_and_Time
2	air_pressure
3	air_pressure_result_value_qualifier
4	Photosynthetically_Active_Radiation
5	Photosynthetically_Active_Radiation_result_value_qualifier
6	air_temperature
7	air_temperature_result_value_qualifier
8	relatively_humidity
9	relatively_humidity_result_value_qualifier
10	Precip
11	Precip_result_value_qualifier
12	wind_speed
13	wind_speed_result_value_qualifier
14	wind_speed_of_gust
15	wind_speed_of_gust_result_value_qualifier
16	wind_from_direction
17	wind_from_direction_result_value_qualifier
18	battery_output
19	battery_output_result_value_qualifier

2 Setup

You will need to install R and RStudio to run the *MMF_weather_QA-compiled.R* script. If installation instructions are needed, please reference the CanWIN Orientation Manual on GitLab for R and RStudio and other software tools.

You will also need to download the Data Processing file from CanWIN's Data Catalogue Weather Keeper Data Processing R script.

To run the script **MMF_weather_QA-compiled.R**.

1. Make sure raw weather station file(s) are all located in the same folder, preferably labelled "raw".
2. Place cursor at indicated code line and click Run on the top right-hand corner of the script window, this is also described in the script.
3. Follow dialogue prompts to direct script to the appropriate folder location for the raw data (you won't see the raw data file(s)).
4. Follow prompts to direct script to save processed file(s) in a separate folder from the raw file(s), preferably label this folder "processed".
5. Sit back and enjoy, the script should run in less than a minute. If you have any issues email portalco@umanitoba.ca.

2.1 Code specifics

Required packages for this script are coded into the script. Hence, if you do not have these packages it will install the packages automatically.

The following packages installed when running the script:

1. **svDialogs** is a package that allows you to create quick standard dialog boxes for Windows, MacOS, and Linux operating systems.
2. **tidyverse** is a collection of R packages for data manipulation, graphing, and much more.

2.1.1 Keyword Definition

There are no overarching keywords in this script to call out functions.

Functions

This script does not have any functions in it and is enclosed by curly braces with dialog boxes integrated in the code to help guide the user. The main functions in this script are three loops: two for-loops (the second one being nested in the first), to standardize and QA the data, and one while-loop to compile the datasets at the end of the script. Additionally, there are if-else statements within the script to test for certain conditions when coding error prone data.

3 Script Description

Script name(s): Weather_data_QAcompiler_XX.R Where XX denotes the version of script, meaning a new script for different stations.

- 2.0 - St Laurent site with last 5 digits of IMEI 13390.
- 2.1 - Dawson Bay with last 5 digits of IMEI 54190.

The versions are not significantly different. There is only one section of the script that differs, which is dependent on the issues each station experienced during operation or data collection. This file is located on the CanWIN Data Catalogue Weather Keeper Data Processing page.

3.1 Inputing working directories

The initial lines of data use the package **svDialogs** to allow the user to guide the code to the directory where their raw data file(s) are stored. The following variables store those directories:

- `input_path` directory pathway of raw weather data file(s) from the DataGarrison.
- `output_path` directory pathway to save processed QA processed and compiled weather data.

If no file(s) are found in a directory, assessed by the length of the *input_path* as *!(isna)*, an error message will appear and stop the script.

3.2 Standardizing variable names and adding metadata

Once the directories are saved in the above variables the names' of the raw weather file(s) in the *input_path* directory are saved in a list *file.names*. An error message will appear if there are no file(s) names in the list.

1. The first for-loop iterates through each file in the *file.names* list using the index *n*, starting from the first list where $n = 1$.
2. A message is printed in the console (bottom-window) of RStudio to let you know which file in the *file.names* list is being read into the RStudio.
3. The first .txt file in the list is read into the script using base R *read.delim* function and setting *skip* to 2 to indicate the first line of date in row 3 of the raw weather data file. This dataset is saved in the variable *met*.
4. A message is printed in the console to indicate the beginning of standardizing variable names.

5. The dataset is then piped (%>%) through a tidyverse function *select()* to remove (with the negative sign) the *X* column at the end of the dataset that was added when the data was read into RStudio with *read.delim()* function.

6. To ensure the script runs properly it passes through an if statement that tests for the number of columns in the dataset.

7. If the number is less than what is expected (11) then the dataset will undergo a process of checking if a portion of a string is in the name of the existing header columns in the dataset, which is saved in the variable *colnam* using the base R function colnames() to grab the names from the dataset. If the name is not found then a column with the header name is inserted into the dataset, with the measurements "NA".

8. The variable colnam is then updated to ensure all the names are in the dataset.

9. The dataset *met* is then piped (%>%) through a tidyverse function *rename()* to change the variable names (left-hand side of equal sign) to their BODC or CF standardized equivalent (right-hand side of equal sign).

10. Metadata *result_value_qualifier* lists are created for each variable, or column, in the dataset (except for the datetime column).

11. The metadata lists are then added into the *met* dataset through the tidyverse function *add_column()* which allows the user to specifiy where the list is inserted, adding it beside the respective variable it is named after.

12. the rows of data by date in chronological order.

13. Before entering the nested for-loop the date and time column is broken down by year, month, day, and time. This is completed by using the base R function *substr()* to take a substring of the date and time column to standardize at a later time.

3.3 Assessing quality of measurements

The second for-loop is nested within the first for-loop to iterate through each row of the dataset to assess each measurement and apply a *result_value_qualifier* code when errors are noted within the deployment details of the station or sensor errors with the reported observations which is described on page 11.

1. A message is printed in the console to indicate the assessment of measurements.

2. The for-loop starts by creating an index *i* to go through (:) each row of the dataset, which is given by *nrow(met)*.

3. To QA the sensors there is a series of if and if-else statements to review measurements.

4. First the if statement checks if the sensor, or column, of data was inserted into the dataset and evaluates the observation i to check if it is an "NA" value or not.

5. If the value is "NA", hence the column is inserted, the result value qualifier is given the code "FEF" to indicate that the sensor failed to connect or stream data for downloading. Otherwise, if the observation is over the upper limit of the calibration range it assigns the result value qualifier *"ADL"* if the measurement is above the value (or true), otherwise if the observation is below the lower limit of the calibration range the measurement is given the value *"BDL"*.

6. If the measurement is between the two calibration values then no value is given in the *...result_value_qualifier* field and is left blank.

7. Problematic measurements, either due to testing during deployment of the weather station or seasonal change, are given the value *"prob_bad"* due to circumstances occurring during the measurement (e.g. sensor malfunction) that were manually assessed by a data curator and field technician.

8. Rain measurements during the turn of seasons (spring and fall) and the wintering months are removed and marked as *"NC"* to refer to as being not collected.

9. For battery output measurements that fall below 4 V, a *"prob_bad"*, result value qualifier is applied, with the remaining sensors marked as *"FEF"*.

10. Once all lines of data have been iterated through by the index *i* the code exits the for-loop.

11. Before exiting the first for-loop the rows of the dataset are renumbered and the *output_filename* is saved as the particular file name being iterated through in the *file.names* list.

12. Depending on the script version the last if-else statements process measurements during a specific time frame if there are any issues that may have occurred over a day, month, or months. To determine what occurred during these times, refer to the QA Summary found on the station dataset page or the deployment details of a specific station. This is usually done at the data curator and field technicians discretion.

13. This file is then written as a .csv in the *output_path* specified at the beginning of the code.

14. This process (described in Section 3.2 and 3.3) is then repeated for each file in the list *file.names* until the index *n* reaches the last name in the list, which is given by the length of the list or the total number of files in the list.

3.4 Compiling data files

This compiler simply creates a list of files within the data/processed folder (output_path) and inserts the lines of the data from the second, third, and etc. datasets into the first dataset listed in folder. Hence, when you are combining a compiled dataset with an archive dataset, make sure you have the compiled dataset listed first. With the naming scheme creating within this script this shouldn't be a problem; however, if you are having issues with the compiled result, this may have caused the problem.

1. The code exits the first for-loop, after iterating through all the files.

2. The *file.names* is then written over to save the file names in the data/processed or *output_path* directory.

3. The list then comprises of the datasets that have been QA'ed in section 3.2 and 3.3.

4. The working directory is then set to *output_path* where the QA files are found.

5. The first file name is read as a dataframe and named *met*.

6. An index is created *n* to keep track of the file name, in the *file.names* list, that will be read as a dataframe, starting with the second QA'ed file in the pathway.

7. The while-loop merges the second and remaining files, named *met2*, into the first file, *met*.

8. Merging of the files in the while-loop is completed by the **tidyverse** command *bind_rows*, which requires all the files to have the same variable names in order to match up the rows.

9. The second file is read into the function with a dynamic base R *read.csv* function that changes with the index *n*.

10. Once the second file is merged into the first, the index n is increased by 1 to indicate the next file in the *file.names* list.

11. The while-loop stops when the index *n* is incremented to the last file name in the *file.names* list, which is determined by the length (or number of names) in the list.

12. The scripts exits the while-loop and reorders the merged, or compiled, *met* data by date to show them in chronological order.

13. The *output_filename* is determined depending on the script version, usually referring to the station name coded with the last 5 digits of the station IMEI.

14. The file is then saved as a .csv in the *output_path* and the name is then appended with *_compiled* and the processing date at the end.

15. A dialog will appear stating the QA file(s) have been compiled and the location will be printed in the console window.

16. Another dialog box will appear on your screen indicating the end of the process, click *OK* and celebrate.

4 Execution Guidelines

The script is meant to be executed as a single R script with the varying stages described above completed in one single process.

4.1 Initializer file

There is no initilization file for you to input your directory pathways and does not require you to set you working directory to a specific location. This is taken care of via the svDialogs package, which was used in created dialog boxes to prompt users to input directory pathways to AVOS files and incubator files, as well as the final output directory. Please, have the directories structured as indicated in Chapter 2.

4.2 Running Script

To run the script all you need to do is place the cursor outside below the text description and above the first curly brace in the code. Then you simply press *Run* on the right side of the ribbon on top of the script window.

This script has no descrepencies for any operating system. It should run successfully, under a minute, on all platforms.

5 Log

Otherwise you can provide additional information here for running your script. For more information how to run an R script click here for a short resource on execution or visit Data Capentry to start understanding RStudio basic features.

A Reference Tables

A.1 Data Levels

Level 0 – Raw data: unprocessed data and data products that have not undergone quality control. Depending on the data type and data transmission system, raw data may be available within seconds or minutes after real-time. Examples include real-time precipitation, streamflow, and water quality measurements

Level 0.1 – First pass QC: A first quality control pass has been performed to remove out of range and obviously erroneous values. These values are deleted from the record. E.g: Online Environment Canada stream-flow data, laboratory data

Level 1 – Quality Controlled Data: Data that have passed quality assurance procedures such as Level 0.1 and have been further quality controlled by data provider before being submitted to CanWIN (e.g. Idronaut data with only downwelling (upwelling data removed) data included.

Level 1.5 – Advanced Quality Controlled Data: Data have undergone complete data provenance (i.e. standardized) in CanWIN. Metadata includes links to protocols and methods, sample collection details, incorporates CanWIN's or another standardized vocabulary, and has analytical units standardized. Note: Process still under development in CanWIN (as of May 13, 2020).

Level 2 – Derived Products: Derived products require scientific and technical interpretation and can include multiple data types. E.g.: watershed average stream runoff derived from stream-flow gauges using an interpolation procedure.

Level 3 – Interpreted Products: These products require researcher (PI) driven analysis and interpretation and/or model-based interpretation using other data and/or strong prior assumptions. E.g.: watershed average stream runoff and flow using streamflow gauges and radarsat imagery

Level 4 – Knowledge Products: These products require researcher (PI) driven scientific interpretation and multidisciplinary data integration and include model-based interpretation using other data and/or strong prior assumptions. E.g.: watershed average nutrient runoff concentrations derived from the combination of stream-flow gauges and nutrient values.

Content retrieved from https://lwbin.cc.umanitoba.ca on July 06, 2020.

A.2 Result Value Qualifiers

ADL	Above Detection Limit
BDL	Below Detection Limit
FD	Field Duplicate
LD	Lab Duplicate
$	Incorrect sample container
EFAI	Equipment failure, sample lost
FEF	Field equipment failed
FEQ	Field Equipment Questionable
FFB	Failed. Field blank not acceptable
FFD	Failed. Field Duplicate
FFS	Failed. Field spike not acceptable
H	Holding time exceeded
ISP	Improper sample preservation
ITNA	Incubation time not attained
ITNM	Incubation temperature not maintained
JCW	Sample container damaged, sample lost
NaN	Value is missing and reason is not known
NC	Not collected
ND	Not detected
NR	Sample taken/measured on site but information in this field not recorded
NS	Sample collected but not submitted
OC	Master Coordinate List Used
P	Analysis requested and result pending
prob_good	probably good value. Data value that is probably consistent with real phenomena but this is unconfirmed or data value forming part of a malfunction that is considered too small to affect the overall quality of the data object of which it is a part
prob_bad	probably bad value. Data value recognised as unusual during quality control that forms part of a feature that is probably inconsistent with real phenomena
Interpolated	This value has been derived by interpolation from other values in the data object
Q	Below limit of quantification (LOQ). The value was below the LOQ of the analytical method. The value in the result field is the limit of quantification (limit of detection) for the method

B Glossary of Options and Packages

B.1 R Packages

Visit https://cran.r-project.org/web/packages/available_packages_by_name.html to learn more about R packages

- **Package 1** - Description
- **Package 2** - Description

B.2 Python

B.2.1 Python Script-Specific Options

- **Option 1** - Description
- **Option 2** - Description

B.2.2 Python Packages

Visit https://docs.python.org/3/library/ to learn more about python packages

- **Package 1** - Description
- **Package 2** - Description

Example: Section 2.1 from Victory's semi-hemi codebook