Ameriflux Data Submission Landscape Flux

Riasad Bin Mahbub and Benjamin Runkle

2024-05-13

Introduction and instruction links

This is an R Markdown document. In this document we are trying to compile the information of the landscape flux 2018-2023 data for AmeriFlux submission. The guidelines for this submission can be obtained from these links:

- YTvideo AMP webinar series: Submitting BADM in CSV format
- YTvideo AMP webinar series: Post-submission data life cycle: FP-In to BASE publishing
- AmeriFlux submission instructions
- AmeriFlux Data Submission PDF
- AmeriFlux Variable Information Instructions
- Uploading Half-Hourly/Hourly Data to AmeriFlux

Location of the files

The location of the data can be obtained from these directories. Shared directory is the directory of the landscape flux group where the data are kept. The data were copied from the shared directory to local directory (rbmhabub's computer) to do the processing of the data

In shared directory:

Way3 Directory: "Y:/Rice/MasterFileSets/Way3/2021_11_20"

Way4 Directory: "Y:/Rice/MasterFileSets/Way4/2021_11_20"

In local directory:

Way3 Directory: "C:/Users/rbmahbub/Documents/RProjects/AmerifluxDataSubmission_LandscapeFlux/Data/Way3"

Way4 Directory: "C:/Users/rbmahbub/Documents/RProjects/AmerifluxDataSubmission_LandscapeFlux/Data/Way4"

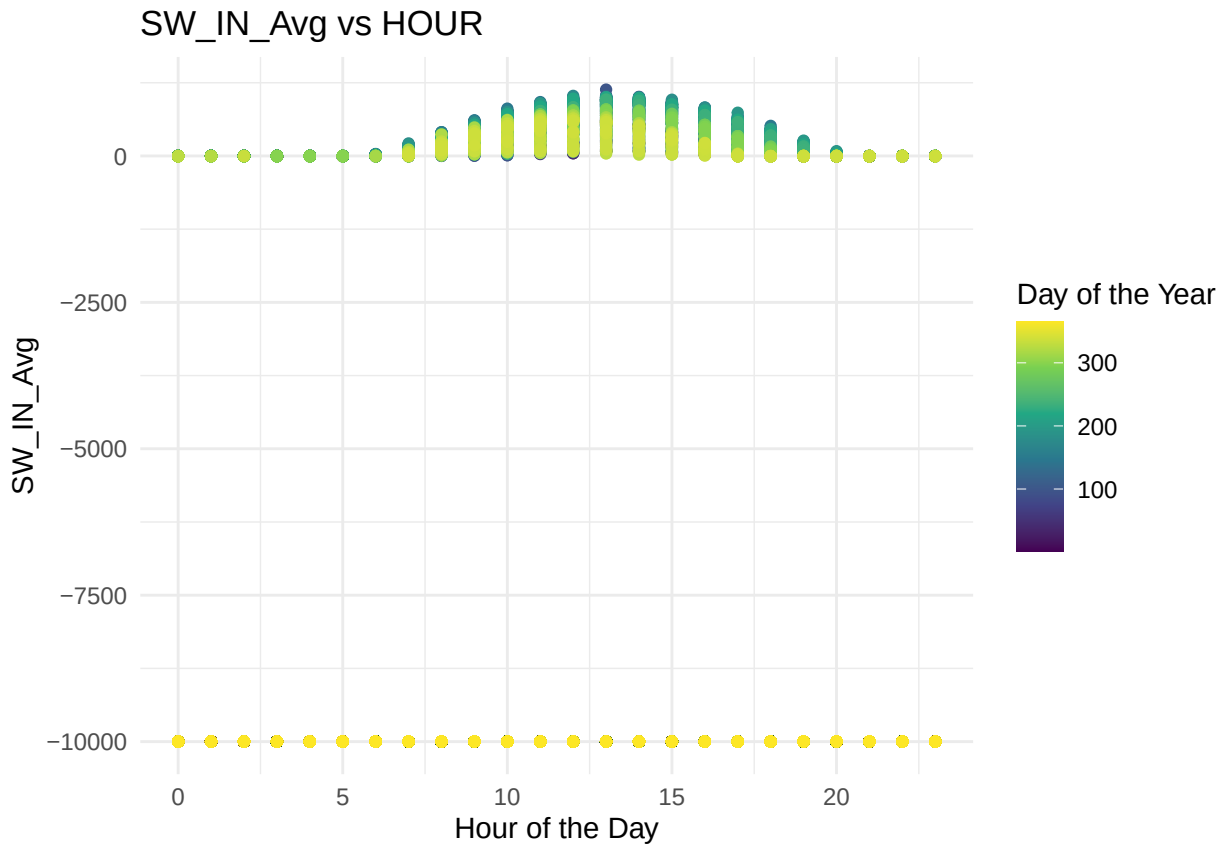
Reading the files and fixing the timestamp

```
## read the files
# Set the directory path and file name
directory_path <- "C:/Users/rbmahbub/Documents/RProjects/AmerifluxDataSubmission_LandscapeFlux/Data/Way3
file_name <- "Way3 2018.csv"
file_path <- file.path(directory_path, file_name)

# Read the CSV file
way3_2018_data <- read.csv(file_path)
```

```r
# Display the first few rows of the data
# Create TIMESTAMP_START and TIMESTAMP_END columns
way3_2018_data <- cbind(TIMESTAMP_START = NA, TIMESTAMP_END = NA, way3_2018_data)
# Convert TIMESTAMP column to POSIXct format
# Load necessary library
library(lubridate)

# Convert TIMESTAMP to datetime objects
way3_2018_data$TIMESTAMP <- ymd_hms(way3_2018_data$TIMESTAMP)

# Create TIMESTAMP_START in the desired format
way3_2018_data$TIMESTAMP_START <- format(way3_2018_data$TIMESTAMP, "%Y%m%d%H%M")

# Create TIMESTAMP_END by adding 30 minutes to TIMESTAMP and formatting it
way3_2018_data$TIMESTAMP_END <- format(way3_2018_data$TIMESTAMP + minutes(30), "%Y%m%d%H%M")

# Convert TIMESTAMP_START and TIMESTAMP_END to character type
way3_2018_data$TIMESTAMP_START <- as.character(way3_2018_data$TIMESTAMP_START)
way3_2018_data$TIMESTAMP_END <- as.character(way3_2018_data$TIMESTAMP_END)
# Create a new column 'HOUR' to store the hour extracted from the TIMESTAMP
way3_2018_data$HOUR <- hour(way3_2018_data$TIMESTAMP)

# Create a new column 'MONTH' to store the month extracted from the TIMESTAMP
way3_2018_data$MONTH <- month(way3_2018_data$TIMESTAMP)

# Create a new column 'DAY_OF_YEAR' to store the day of the year extracted from the TIMESTAMP
way3_2018_data$DOY <- yday(way3_2018_data$TIMESTAMP)

# Replace NaN and NA values with -9999
# Define custom function to handle NaN values in data frames
is.nan.data.frame <- function(x) {
  do.call(cbind, lapply(x, is.nan))
}

# Replace NaN with -9999
way3_2018_data[is.nan.data.frame(way3_2018_data)] <- -9999


# Assuming way3_2018_data is your dataset
print(way3_2018_data[1:3, 1:3])
```

```
##   TIMESTAMP_START TIMESTAMP_END           TIMESTAMP
## 1    201801010000  201801010030 2018-01-01 00:00:00
## 2    201801010030  201801010100 2018-01-01 00:30:00
## 3    201801010100  201801010130 2018-01-01 01:00:00
```

Saving the files



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

source: https://www.youtube.com/watch?v=wS95r9JbbB0&t=2336s

Guidelines from the video:

1. Do support list of common variable names (From the table)
2. Use the exact variable names and the units
3. Very first on the list, what to do with the data, remove the known values
4. U-star filtering of the data
5. CSV is a delimited text file that uses a comma to separate values
6. The first two column are TIMESTAMP_START and TIMESTAMP_END (ISO time format: YYYYMMDDHHMM e.g., 201810220930)
7. Dont convert the scientific notations in timestamps
8. An hour column to check the data

Read all way 3 and way 4 files

Check if they have same number of columns

Check if they have same columns

Check if they have same serialized columns

```
# Load necessary libraries
# Load necessary libraries
library(dplyr)

# Function to read all files and return a list of dataframes
read_files <- function(file_paths) {
  lapply(file_paths, read.csv, stringsAsFactors = FALSE)
}

# Function to print the number of rows and columns for each dataframe
print_dimensions <- function(data_list, file_names) {
  for (i in seq_along(data_list)) {
    rows <- nrow(data_list[[i]])
    cols <- ncol(data_list[[i]])
    cat("File:", file_names[i], "- Rows:", rows, "- Columns:", cols, "\n")
  }
}

# Specify the file paths for way 3 and way 4 files
way3_dir <- "C:/Users/rbmahbub/Documents/RProjects/AmerifluxDataSubmission_LandscapeFlux/Data/Way3"
way4_dir <- "C:/Users/rbmahbub/Documents/RProjects/AmerifluxDataSubmission_LandscapeFlux/Data/Way4"

way3_files <- list.files(path = way3_dir, pattern = "*.csv", full.names = TRUE)
way4_files <- list.files(path = way4_dir, pattern = "*.csv", full.names = TRUE)

# Read all files
way3_data <- read_files(way3_files)
way4_data <- read_files(way4_files)

# Print the number of rows and columns for each file
print_dimensions(way3_data, basename(way3_files))
```

```
## File: Way3 2018.csv - Rows: 17520 - Columns: 528
## File: Way3 2019.csv - Rows: 17520 - Columns: 528
## File: Way3 2020.csv - Rows: 17568 - Columns: 528
## File: Way3 2021.csv - Rows: 12602 - Columns: 528
## File: Way3 2022.csv - Rows: 17473 - Columns: 121
## File: Way3 2023.csv - Rows: 17473 - Columns: 121
```

```
print_dimensions(way4_data, basename(way4_files))
```

```
## File: Way4 2018.csv - Rows: 17520 - Columns: 481
## File: Way4 2019.csv - Rows: 17520 - Columns: 481
## File: Way4 2020.csv - Rows: 17568 - Columns: 481
## File: Way4 2021.csv - Rows: 13811 - Columns: 481
## File: Way4 2022 WTD_Corr.csv - Rows: 17473 - Columns: 140
## File: Way4 2022.csv - Rows: 17473 - Columns: 140
## File: Way4 2023 WTD_Corr.csv - Rows: 11185 - Columns: 135
## File: Way4 2023.csv - Rows: 17474 - Columns: 135
```

```r
# Function to check if all dataframes in a list have the same number of columns
check_same_num_columns <- function(data_list) {
  num_columns <- sapply(data_list, ncol)
  return(length(unique(num_columns)) == 1)
}

# Function to check if all dataframes in a list have the same column names
check_same_columns <- function(data_list) {
  column_names <- lapply(data_list, colnames)
  return(length(unique(column_names)) == 1)
}

# Function to check if all dataframes in a list have the same serialized columns
check_same_serialized_columns <- function(data_list) {
  serialized_columns <- sapply(data_list, function(df) paste(colnames(df), collapse = ""))
  return(length(unique(serialized_columns)) == 1)
}

# Check way 3 files
way3_same_num_columns <- check_same_num_columns(way3_data)
way3_same_columns <- check_same_columns(way3_data)
way3_same_serialized_columns <- check_same_serialized_columns(way3_data)

# Check way 4 files
way4_same_num_columns <- check_same_num_columns(way4_data)
way4_same_columns <- check_same_columns(way4_data)
way4_same_serialized_columns <- check_same_serialized_columns(way4_data)

# Compare way 3 and way 4 files
if (way3_same_num_columns && way4_same_num_columns) {
  way3_num_columns <- ncol(way3_data[[1]])
  way4_num_columns <- ncol(way4_data[[1]])
  same_num_columns <- (way3_num_columns == way4_num_columns)
} else {
  same_num_columns <- FALSE
}

if (way3_same_columns && way4_same_columns) {
  way3_columns <- colnames(way3_data[[1]])
  way4_columns <- colnames(way4_data[[1]])
  same_columns <- all(way3_columns %in% way4_columns) && all(way4_columns %in% way3_columns)
} else {
  same_columns <- FALSE
}

if (way3_same_serialized_columns && way4_same_serialized_columns) {
  way3_serialized_columns <- paste(colnames(way3_data[[1]]), collapse = "")
  way4_serialized_columns <- paste(colnames(way4_data[[1]]), collapse = "")
  same_serialized_columns <- (way3_serialized_columns == way4_serialized_columns)
} else {
  same_serialized_columns <- FALSE
}
```

```r
# Output the results
results <- list(
  way3_same_num_columns = way3_same_num_columns,
  way3_same_columns = way3_same_columns,
  way3_same_serialized_columns = way3_same_serialized_columns,
  way4_same_num_columns = way4_same_num_columns,
  way4_same_columns = way4_same_columns,
  way4_same_serialized_columns = way4_same_serialized_columns,
  same_num_columns = same_num_columns,
  same_columns = same_columns,
  same_serialized_columns = same_serialized_columns
)

print(results)
```

```
## $way3_same_num_columns
## [1] FALSE
##
## $way3_same_columns
## [1] FALSE
##
## $way3_same_serialized_columns
## [1] FALSE
##
## $way4_same_num_columns
## [1] FALSE
##
## $way4_same_columns
## [1] FALSE
##
## $way4_same_serialized_columns
## [1] FALSE
##
## $same_num_columns
## [1] FALSE
##
## $same_columns
## [1] FALSE
##
## $same_serialized_columns
## [1] FALSE
```

Export the data

```r
# Create the directory if it doesn't exist
dir.create("C:/Users/rbmahbub/Documents/RProjects/AmerifluxDataSubmission_LandscapeFlux/Data/AFguidedSu

# Specify the file path for saving
file_path <- "C:/Users/rbmahbub/Documents/RProjects/AmerifluxDataSubmission_LandscapeFlux/Data/AFguided
```

```r
# Save the dataframe
write.csv(way3_2018_data, file = file_path, row.names = FALSE)

# Confirmation message
cat("way3_2018_data saved successfully.\n")
```

```
## way3_2018_data saved successfully.
```