MaxEnt-ETRHEQ-WET Tutorial

Introduction

This tutorial provides an example demonstrating how to run the MaxEnt-ETRHEQ-WET model at the hourly scale using friction velocity data obtained from ERA5 (Hourly data on single levels from 1940 to present).The FLUXNET CZ-wet site (https://fluxnet.org/doi/FLUXNET2015/CZ-wet) is used as the example application.

Requirements

- R version 4.0 or higher
- R packages: listed below

```
library(dplyr)
library(purrr)
library(lubridate)
library(ggplot2)
library(gridExtra)
library(cowplot)
library(ggpubr)
library(Metrics)
library(ggpmisc)
library(grid)
```

Step 1: Prepare Weather Data for Modelling

The first step is to prepare the weather data required to run the MaxEnt-ETRHEQ-WET model. At minimum, the following variables are needed:

```
Air temperature (Ta, in Kelvin)

Atmospheric pressure (p, in Pa)

Wind speed (u, in m·s ⁻¹)

Friction velocity (Ustar, in m·s ⁻¹)

Air relative humidity (RH, in decimal form)

Net radiation (Rn, in W·m ⁻²)

Measurement height (z, in meters)
```

```
Vegetation height (zveg, in meters)
```

Although this dataset includes measured friction velocity (Ustar), for the purpose of this tutorial, it is assumed that Ustar is not available. Instead, this tutorial demonstrates how friction velocity can be obtained from ERA5 data following the procedures outlined below.

Please make sure that the dataset is properly time-aligned (e.g., at hourly resolution) and that any missing data are appropriately handled before proceeding to modelling.

```r
## This block demonstrates how to download ERA5 friction velocity data (ustar) via the
 ↪  ECMWF Climate Data Store

# Load the required libraries
library(ecmwfr)
library(ncdf4)
library(tidyverse)

# Check the current working directory where downloaded data will be saved
getwd()

# Set ECMWF API key (provided upon registration with the ECMWF data service)
wf_set_key(key = "Your key")

# Request the friction velocity data for the site CZ-wet
request <- wf_request(
  request = list(
    "dataset_short_name" = "reanalysis-era5-single-levels",
    "product_type" = "reanalysis",
    "variable" = "friction_velocity",
    "year" = as.character(2006:2014),
    "month" = sprintf("%02d", 1:12),
    "day" = sprintf("%02d", 1:31),
    "time" = sprintf("%02d:00", 0:23),  # Hourly data
    "area" = c(49.0246, 14.7704, 49.0247, 14.7704), #north, west, south, east
    "format" = "netcdf"
  ),
  transfer = TRUE,
  path = 'C:/Users/...'  # Use a valid directory
)
```

After running the block above, you should obtain the friction velocity data in .nc format.

In this example, the downloaded file was named ecmwfr_6fe82c3f40ae.nc.

To facilitate combining it with other weather data, it is recommended to reformat the friction velocity data and its corresponding timestamps into a CSV file. The following block of code handles the formatting and saves the output as a CSV file named era5_cz_wet_ustar.csv.

```r
## This block demonstrates how to format and saves the friction velocity data as a CSV
 ↪  file

# Load NetCDF file for CZ-wet site
nc_data <- nc_open("ecmwfr_6fe82c3f40ae.nc") # Put your own downloaded file
```

```r
# List all available variables
print(nc_data$var)
print(nc_data$dim)  # Lists all dimensions

# You will see that the time variable is labeled as "valid_time", and friction velocity
 ↪  is stored as "zust".

# Extract time values
time <- ncvar_get(nc_data, "valid_time")  # Extract time

# Convert to readable format
timestamp <- as.POSIXct(time, origin = "1970-01-01", tz = "UTC")

# Format timestamp as "YYYY-MM-DD HH:MM:SS AM/PM"
timestamp <- format(timestamp, "%Y-%m-%d %I:%M:%S %p")

# Extract ustar data
ustar <- ncvar_get(nc_data, "zust")  # Extract friction velocity

# Close the NetCDF file
nc_close(nc_data)

# Convert to dataframe
df <- tibble(timestamp = timestamp, ustar = ustar)

# Convert tibble to data.frame
df <- as.data.frame(df)

# Save as CSV
write.csv(df, "era5_cz_wet_ustar.csv", row.names = FALSE)
```

Now you can load the weather data and combine it with the ERA5 friction velocity data.

It is important to note that the input data from the FLUXNET product are provided at a half-hourly timescale, while the ERA5 data are at an hourly timescale. Therefore, it is recommended to first aggregate the FLUXNET data to an hourly resolution before combining it with the ERA5 friction velocity data.

```r
## This block demonstrates how to load the weather data needed for MaxEnt-ETRHEQ-WET,
 ↪  aggregate the FLUXNET half-hourly data to an hourly timescale, and combine it with
 ↪  the ERA5 friction velocity data.

#Import fluxnet data from the CZ_wet site
CZ_wet <- read.csv("FLX_CZ-wet_FLUXNET2015_FULLSET_HH_2006-2014_1-4.csv")

# Select relevant variables and rename for clarity
CZ_wet <- CZ_wet %>% dplyr::select(
    TIMESTAMP_START,PA_F,PA_F_QC,TA_F,TA_F_QC,WS_F,WS_F_QC,RH,
 ↪  NETRAD,G_F_MDS,G_F_MDS_QC,LE_F_MDS, LE_F_MDS_QC,H_F_MDS,H_F_MDS_QC, LE_CORR,H_CORR)
 ↪  %>%
  rename(Pressure=PA_F,Ta=TA_F,WS=WS_F,Rn=NETRAD)

# Filter out records with poor quality control flags (QC > 1)
CZ_wet <- CZ_wet  %>%
```

```r
  filter(!(PA_F_QC >1),
         !(TA_F_QC >1),
         !(WS_F_QC >1),
         !(G_F_MDS_QC >1),
         !(LE_F_MDS_QC >1),
         !(H_F_MDS_QC >1))

# Replace missing values coded as -9999 with NA
CZ_wet <- CZ_wet %>%
  mutate_all(~na_if(.,-9999))

# Convert units
CZ_wet <- CZ_wet  %>%
  mutate( Pressure=Pressure*1000,
          Ta = Ta + 273.15,
          RH = RH/100)

# Convert TIMESTAMP_START to datetime
CZ_wet_input <- CZ_wet %>%
  mutate(TIMESTAMP_START = ymd_hm(as.character(TIMESTAMP_START)))

# Create hourly timestamp by rounding to the nearest hour
CZ_wet_input_1 <- CZ_wet_input %>%
  mutate(Hour = floor_date(TIMESTAMP_START, "hour")) %>%
  group_by(Hour) %>%
  filter(n() == 2) %>%  # Keep only hours with exactly 2 records
  summarise(
            Pressure = mean(Pressure, na.rm = TRUE),
            Ta = mean(Ta, na.rm = TRUE),
            WS = mean(WS, na.rm = TRUE),
            RH = mean(RH, na.rm = TRUE),
            Rn = mean(Rn, na.rm = TRUE),
            G_F_MDS = mean(G_F_MDS, na.rm = TRUE),
            LE_F_MDS = mean(LE_F_MDS, na.rm = TRUE),
            H_F_MDS = mean(H_F_MDS, na.rm = TRUE),
            LE_CORR = mean(LE_CORR, na.rm = TRUE),
            H_CORR = mean(H_CORR, na.rm = TRUE))

# Set measurement and vegetation heights
CZ_wet_input_1$z <- 2.6     # Measurement height (m)
CZ_wet_input_1$zveg <- 1    # Vegetation height (m)

# Import the ERA5 friction velocity data
CZ_wet_ERA <- read.csv("era5_cz_wet_ustar.csv")

# Convert timestamp from 12-hour AM/PM format to 24-hour POSIXct format
CZ_wet_ERA <- CZ_wet_ERA %>%
  mutate(timestamp = as.POSIXct(timestamp, format = "%Y-%m-%d %I:%M:%S %p",
                                tz = "UTC")) %>% rename(Ustar = ustar)
                                             # Rename column ustar → Ustar

# Merge eddy covariance data with ERA5 reanalysis data based on hourly timestamps
CZ_wet_input_2 <- CZ_wet_input_1 %>%
  left_join(CZ_wet_ERA, by = c("Hour" = "timestamp"))
```

```r
# Keep only rows with complete data
CZ_wet_input_2 <- CZ_wet_input_2[complete.cases(CZ_wet_input_2), ]

# Remove records violating the surface energy balance constraint
CZ_wet_input_2 <- CZ_wet_input_2  %>%
  filter(!(Rn - G_F_MDS - LE_F_MDS - H_F_MDS > 50),
         !(Rn - G_F_MDS < 0))

# Assign cleaned data to 'data' for modelling input
data <-CZ_wet_input_2
```

Step 2: Build the MaxEnt-ETRHEQ method

Now that the data for MaxEnt-ETRHEQ are prepared, you can proceed to build the MaxEnt-ETRHEQ method for estimating H and LE.

```r
## This block demonstrates how to create a function that estimates H and LE by minimizing
 ↪  the dissipation function (D) in the MaxEnt-ETRHEQ-WET method.

# This method does not require land surface properties,
# except for vegetation height (zveg) and measurement height (z).

# It generates a series of surface temperature (Ts) and relative humidity (RHs) values,
# and selects the combination that minimizes the dissipation function (D).

Modelling_method_ns <-function(data){
  # Create a new data frame for the best results
  best_results <- data.frame(RHs = numeric(nrow(data)), Ts = numeric(nrow(data)),H =
 ↪  numeric(nrow(data)), LE = numeric(nrow(data)))

  # Create a function to calculate D for a given pair of RHs and Ts
  calculate_D <- function(Ts, RHs, p, Ta, u, ustar, RH, Rn, z, zveg) {
    #Define constants:
    epsilon <- 0.622 #the dimensionless ratio of the gas constant for dry air to water
 ↪  vapor
    cp <- 1004.7 #the specific heat of air at constant pressure (J·kg-1·K-1),
    lambda <- 2.502 * 10^6 #the latent heat of vaporization (J·kg-1).
    g <- 9.81 #the gravitational acceleration (m·s-2),
    Rd <- 287 #gas constant for dry air (J·kg-1·K-1)
    k <- 0.41 #von Karman constant
    gamma <- cp/lambda

    ps <- p / exp(-g * z / (Rd * Ta))
    rho <- p / (Rd * Ta)
    estar_Ts <- 611.2 * exp(17.67 * (Ts - 273.15) / (Ts - 29.65))
    qstar_Ts <- epsilon * estar_Ts / (ps - (1 - epsilon) * estar_Ts)
    estar_T <- 611.2 * exp(17.67 * (Ta - 273.15) / (Ta - 29.65))
    qstar_T <- epsilon * estar_T / (p - (1 - epsilon) * estar_T)
    delta <- (qstar_Ts - qstar_T) / (Ts - Ta)

    if(zveg==0){
```

```
    zoh <-0.001
    zom <-0.001
    d <- 0
    ra <-log(z/zoh)*log(z/zom)/((k^2)*u)
  }else{
    d <- 0.7 * zveg
    zom <- 0.1 * zveg
    Re <- ustar * zom / (1.45E-5)
    zoh <- zom / exp(k * (6 * Re^(1/4) - 5))
    ra <- (log((z - d) / zom) * log((z - d) / zoh)) / ((k^2) * u)
  }
  Ri <- (5 * g * (z-d) * (Ts - Ta)) / (Ta * (u^2))
  eta <- ifelse(Ts > Ta, 0.75, 2)
  rah <- ra / ((1 + Ri)^eta)
  L <- -ustar^3*Ta*rah/(k*g*(Ts-Ta))
  zeta <- (z-d)/L
  if (!is.na(zeta)&&zeta > 0) {
    Kh <- k * ustar * (z-d) / (1 + 5 * zeta)
  } else if (!is.na(zeta)&&zeta == 0) {
    Kh <- k * ustar * (z-d)
  } else if (!is.na(zeta)&&zeta < 0) {
    Kh <- k * ustar * (z-d) / (1 * (1 - 16 * zeta)^(-1/2))
  }
  ga <- 1 / rah
  H <- rho * cp * ga * (Ts - Ta)
  LE <- lambda*rho*ga*(RHs*qstar_Ts-RH*qstar_T)
  G <- Rn - LE - H
  G_limit <- 0.2*Rn
  gamma_PM <- cp*p/(0.622*lambda)
  delta_PM <-
↪ 1000*4098*(0.6108*exp(17.27*(Ta-273.15)/((Ta-273.15)+237.3)))/((Ta-273.15+237.3)^2)
  RHs_limit <-1.26*gamma_PM*lambda/(delta*(1-1.26*lambda)+gamma_PM)

  if (!is.na(H) && is.numeric(LE) && is.numeric(G) && RHs >=RHs_limit && G<=G_limit) {
    Is <- 1300
    Ia <- rho * cp * sqrt(Kh)
    Ie <- delta / gamma * RHs * Ia
    D <- 2 * (G^2) / Is + 2 * (H^2) / Ia + 2 * (LE^2) / Ie
    return(list(D = D, RHs=RHs,Ts=Ts,H = H, LE = LE))

  } else {
    return(list(D=9999, RHs=RHs,Ts=Ts,H=H,LE=LE))
  }
}

# Loop through rows and find the best RHs and Ts for each row
for (i in 1:nrow(data)) {
  row_data <- data[i, ]
  # Create a sequence of Ts and RHs
  Ts_range <- seq(row_data$Ta-30, row_data$Ta + 30, 0.1)
  RHs_range <- seq(row_data$RH, 1, 0.1)

  # Create all combinations of Ts and RHs
  combinations <- expand.grid(Ts = Ts_range, RHs = RHs_range)
```

```
    # Calculate D for all combinations
    results <- pmap(list(combinations$Ts, combinations$RHs, row_data$Pressure,
 ↪  row_data$Ta, row_data$WS, row_data$Ustar, row_data$RH ,row_data$Rn, row_data$z,
 ↪  row_data$zveg),
                    ~ calculate_D(..1, ..2, ..3, ..4, ..5, ..6, ..7, ..8, ..9, ..10))

    # Find the best pair of Ts and RHs that lead to the minimum D for this row
    best_combination_index <- which.min(map_dbl(results, "D"))
    best_combination <- combinations[best_combination_index, ]

    # Assign the best RHs and Ts to the best_results data frame
    best_results$Ts[i] <- best_combination$Ts

    # Retrieve the values from the calculate_D result
    D_values <- results[[best_combination_index]]
    best_results$RHs[i] <- D_values$RHs
    best_results$H[i] <- D_values$H
    best_results$LE[i] <- D_values$LE
  }

  return(best_results)
}
```

Step 3: Estimating hourly H and LE

Once the function is built, you can simply call it with the prepared data inputs to obtain the modelling results. The modelled outputs can then be merged with the measured flux data at corresponding timestamps to allow for comparison between modelled and measured H and LE.

```
## This block demonstrates how to call the function to estimate H and LE using the
 ↪  prepared weather data.

# Run the MaxEnt-ETRHEQ-WET modelling method
results_CZ_wet <-Modelling_method_ns(data)

# Merge modelled results with input data
results_CZ_wet <- cbind(CZ_wet_input_2,results_CZ_wet)

# Remove rows with any NA in the results
results_CZ_wet <- results_CZ_wet[complete.cases(results_CZ_wet), ]

# Export the final results to CSV
write.csv(results_CZ_wet, "Results_CZ_wet.csv", row.names = FALSE)
```

Step 4: Plot modelled vs. measured H and LE fluxes

In this step, you will create scatter plots to compare the modelled and measured values of sensible heat flux (H) and latent heat flux (LE).

```
## This block demonstrates how to calculate RMSE and create scatter plots to compare
 ↪  modelled and measured H and LE fluxes.

# Calculate RMSE for LE and H
rmse_value_CZ_wet_LE <-rmse(results_CZ_wet$LE_F_MDS,results_CZ_wet$LE)
rmse_value_CZ_wet_LE
```

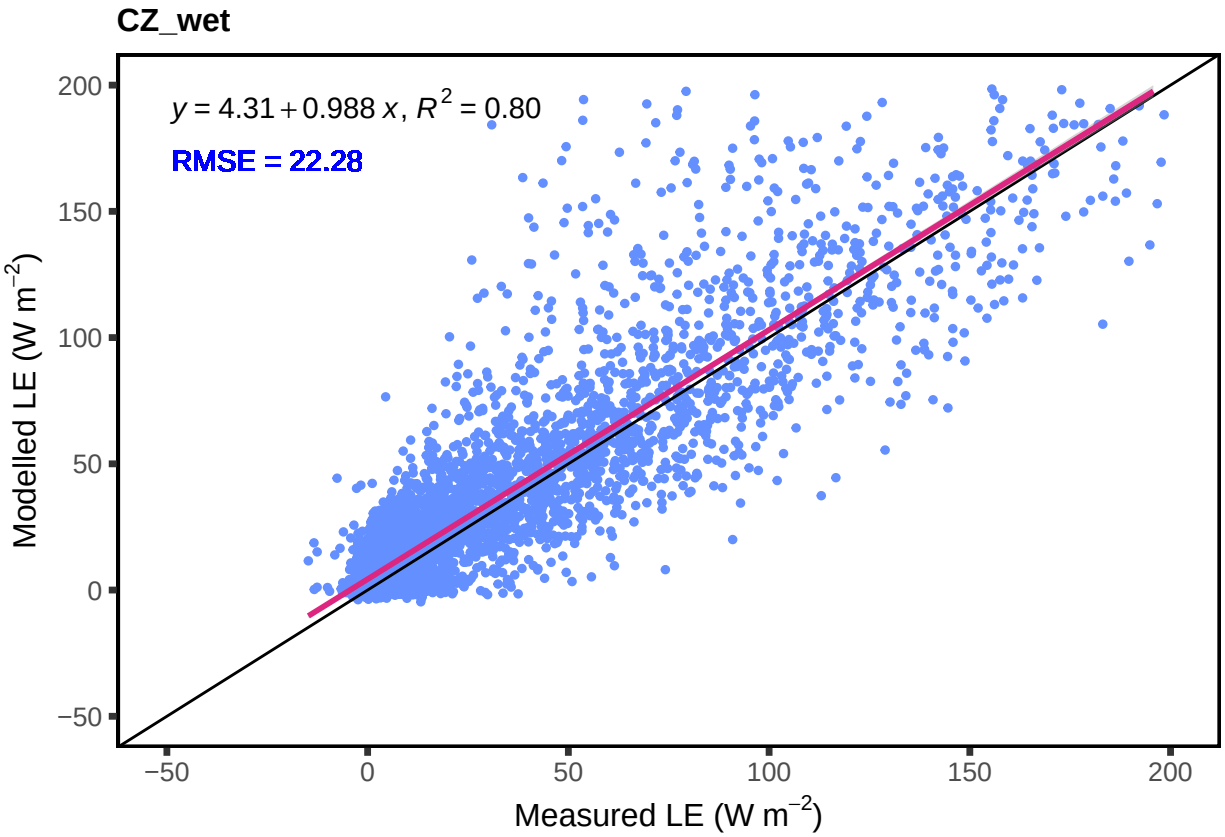
```
## [1] 22.27869
```

```
rmse_value_CZ_wet_H <-rmse(results_CZ_wet$H_F_MDS,results_CZ_wet$H)
rmse_value_CZ_wet_H
```

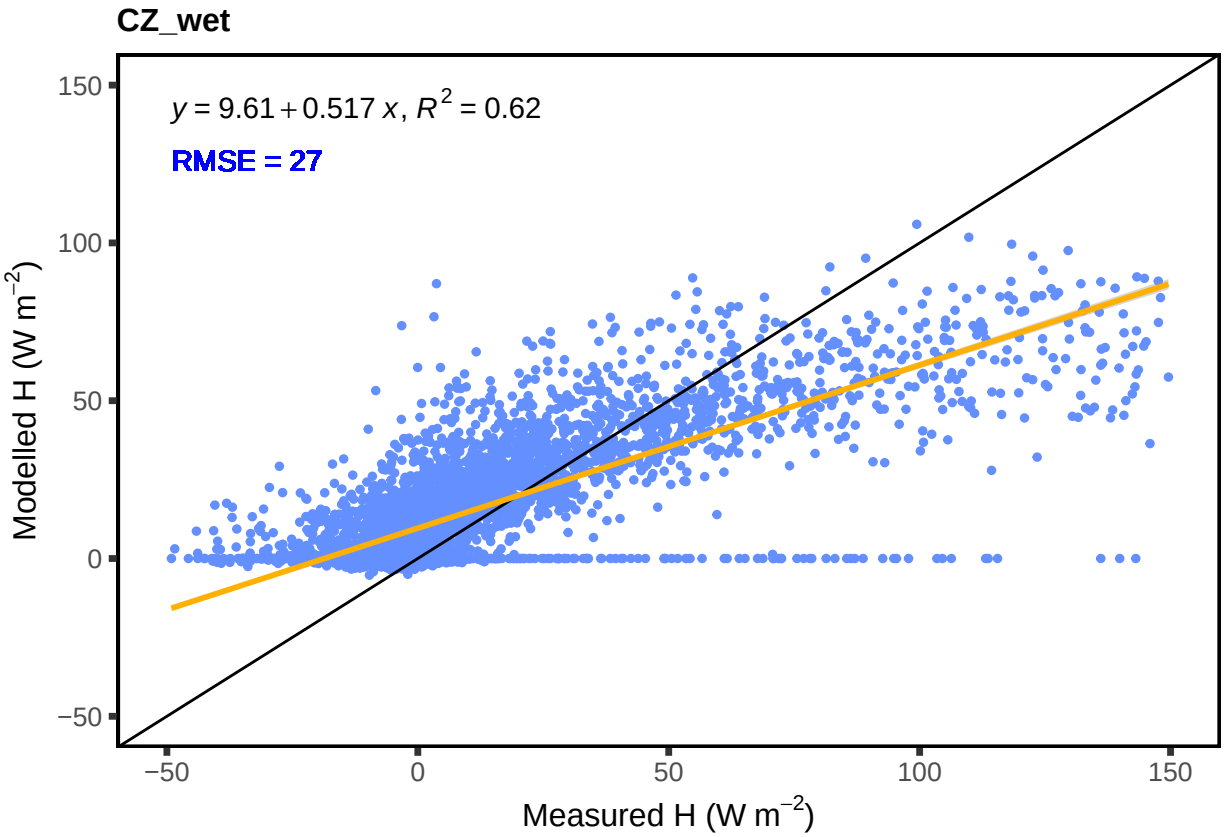
```
## [1] 27.00012
```

```
# Define a custom theme for publication-quality plots
theme_pub <- theme_bw(base_size = 12) +
  theme(
    plot.title = element_text(size = 12, face = "bold"),
    axis.text = element_text(size = 10),
    panel.border = element_rect(colour = "black", size = 1.2),
    axis.ticks = element_line(size = 1.2),
    panel.grid = element_blank(),
    legend.position = "none"
  )

# Plot modelled vs. observed LE
CZ_wet_modelled_LE <-ggplot(results_CZ_wet, aes(x=LE_F_MDS,y=LE)) +
  geom_point(size=1,color="#648fff")+ geom_abline(intercept = 0, slope = 1)+
  labs(title="CZ_wet")+xlab(expression("Measured LE (W"~m^{-2}*")")) +
ylab(expression("Modelled LE (W"~m^{-2}*")"))+
  stat_poly_line(color="#dc267f") +stat_poly_eq(use_label(c("eq", "R2")))
   ↪  +ylim(-50,200)+xlim(-50,200)+
  geom_text(x = -25, y = 170, label = paste("RMSE =", round(rmse_value_CZ_wet_LE, 2)),
   ↪  size = 3.8, color = "blue") +  # Add RMSE
  theme_pub
CZ_wet_modelled_LE
```




```
# Plot modelled vs. observed H
CZ_wet_modelled_H <-ggplot(results_CZ_wet, aes(x=H_F_MDS,y=H)) +
  geom_point(size=1,color="#648fff")+ geom_abline(intercept = 0, slope = 1)+
  labs(title="CZ_wet")+xlab(expression("Measured H (W"~m^{-2}*")")) +
ylab(expression("Modelled H (W"~m^{-2}*")"))+
  stat_poly_line(color="#ffb000") +stat_poly_eq(use_label(c("eq", "R2")))
   ↪  +ylim(-50,150)+xlim(-50,150)+
  geom_text(x = -34, y = 126, label = paste("RMSE =", round(rmse_value_CZ_wet_H, 2)),
   ↪  size = 3.8, color = "blue") +  # Add RMSE
  theme_pub
CZ_wet_modelled_H
```



If you have any questions, please feel free to reach out to Yi (Abby) Wang (yi.wang1@uwaterloo.ca).