REddyProc Muka Head Analysis

Yusri Yusup

2023-07-27

Step 0: Preliminary Work

Install the Package

The package needs to be installed prior to use. You might also need to install other packages to run REddyProc. You can do so by using the `install.packages` command for their installation.

```
#install.packages("REddyProc", repos = "http://cran.us.r-project.org")
```

You will need to load the package after a successful installation.

```
library(openair)
library(REddyProc)
```

Step 1: Prepare the Data

Step 1-1: Import the Data

First,we create a data frame suitable for REddyProc.The script co2_longTerm 2.R needs to be run first so that `df` becomes available.

Make a data frame that contains only the necessary variables for REddyProc. They are:

1. `DateTime` in the POSIX format.
2. `NEE` or carbon dioxide flux.
3. `Ustar` or friction velocity
4. Meteorological data for the gap-filling and partitioning steps.
    - `Rg`, *`Tair`,
    - `rH`, and or
    - `VPD`

```
load('~/OneDrive - Universiti Sains Malaysia/Documents/research/Data_analysis/muka_head_co2_longTerm/Mu
Muka_Head <- data.frame(DateTime = df$date, NEE = df$FCO2, Ustar = df$USTAR,
                        Rg = df$RG, Tair = df$TA_ema, rH = df$RH)
```

Check the Data for Continuity of DateTime

We cannot create the REddyProc object if the `DateTime` variable is not continuous. I use the `openair` package and the `timeAverage` function to fill in the missing time stamps.

```
Muka_Head2 <- Muka_Head
names(Muka_Head2)[1] <- "date"
Muka_Head2 <- timeAverage(Muka_Head2, avg.time = "30 min")
names(Muka_Head2)[1] <- "DateTime"
```

Data Overview

Get an overview of the data. Look at the data parameters and take note of missing data or `NA`.

Characteristics

- Surface: Coastal water
- Time zone: +8 GMT
- Latitude, Longitude: 5.49N, 100.2025E

```
summary(Muka_Head2)
```

```
##     DateTime                       NEE              Ustar        
##  Min.   :2015-11-12 00:30:00   Min.   :-4.950   Min.   :0.0010  
##  1st Qu.:2017-02-11 00:22:30   1st Qu.:-0.134   1st Qu.:0.0234  
##  Median :2018-05-14 00:15:00   Median :-0.004   Median :0.0339  
##  Mean   :2018-05-14 00:15:00   Mean   :-0.019   Mean   :0.0407  
##  3rd Qu.:2019-08-14 00:07:30   3rd Qu.: 0.109   3rd Qu.:0.0532  
##  Max.   :2020-11-13 00:00:00   Max.   :44.000   Max.   :0.3489  
##                                NA's   :26826    NA's   :2477    
##        Rg                Tair             rH        
##  Min.   :   0.000   Min.   :22.97   Min.   : 25.10  
##  1st Qu.:   0.097   1st Qu.:27.06   1st Qu.: 76.27  
##  Median :  61.572   Median :28.16   Median : 84.87  
##  Mean   : 219.464   Mean   :28.20   Mean   : 84.43  
##  3rd Qu.: 420.588   3rd Qu.:29.36   3rd Qu.: 94.32  
##  Max.   :1020.553   Max.   :33.56   Max.   :100.00  
##  NA's   :16102      NA's   :19573   NA's   :10854
```

Step 1-2: Calculate Needed Parameters

Essential parameters can be calculated from existing parameters using functions available in REddyProc.

Some useful functions are:

1. `fConvertTimeToPosix`.
2. `fCalcVPDfromRHandTair`. We will use this in the demo.
3. `fCalcETfromLE`
4. `fConvertCtoK`

There are other functions in the package and the function name begins with the prefix `f`.

Step 1-3: Missing VPD in the Data

The dataset does not have the VPD parameter, which could be useful for gap-filling and partitioning.

Calculate VPD

We can calculate VPD using the function `fCalcVPDfromRHandTair`. The input arguments' units are stated in the documentation, `?fCalcVPDfromRHandTair`.

```
VPD <- fCalcVPDfromRHandTair(Muka_Head2$rH,   # The unit is %
                             Muka_Head2$Tair) # The unit is degree Celsius
Muka_Head2 <- cbind(Muka_Head2,VPD)
rm(VPD) # A house-keeping step.
head(Muka_Head2)
```

```
##              DateTime      NEE     Ustar  Rg     Tair       rH      VPD
## 1 2015-11-12 00:30:00 0.474235 0.0322828 NaN 25.38409 93.99698 1.948605
## 2 2015-11-12 01:00:00 0.414857 0.0288856 NaN 25.59989 93.87791 2.012859
## 3 2015-11-12 01:30:00      NaN 0.0224402 NaN 25.62008 93.94426 1.993429
## 4 2015-11-12 02:00:00 0.355199 0.0309368 NaN 25.49872 93.57683 2.099222
## 5 2015-11-12 02:30:00 0.477219 0.0268177 NaN 25.46746 92.12846 2.567812
## 6 2015-11-12 03:00:00 0.308423 0.0239833 NaN 25.39918 91.22686 2.850351
```

Step 2: Create the Gebesee REddyProc Object Class

Before REddyProc can work on your data, the data has to be converted to the REddyProc object.

Create the data object for the data. The ID is `MY-MkH` and the parameters are:

1. `NEE`
2. `Rg`
3. `Tair`
4. `VPD`
5. `Ustar`

```
EProcMYMkH <- sEddyProc$new('MY-MkH', Muka_Head2, c('NEE','Rg','Tair','VPD','Ustar'))
```

```
## New sEddyProc class for site 'MY-MkH'
```

Check the Object

Check the additional info of the data.

```
EProcMYMkH$sLOCATION
```

```
## $LatDeg
## [1] NA
##
```

```
## $LongDeg
## [1] NA
##
## $TimeZoneHour
## [1] NA
```

Add the location information. This is important for the daytime-nighttime partitioning analysis because it requires the time to be accurate.

```
EProcMYMkH$sSetLocationInfo(LatDeg = 5.49, LongDeg = 100.202, TimeZoneHour = 8)
EProcMYMkH$sLOCATION
```

```
## $LatDeg
## [1] 5.49
##
## $LongDeg
## [1] 100.202
##
## $TimeZoneHour
## [1] 8
```

Step 3: u_*-Threshold Estimation

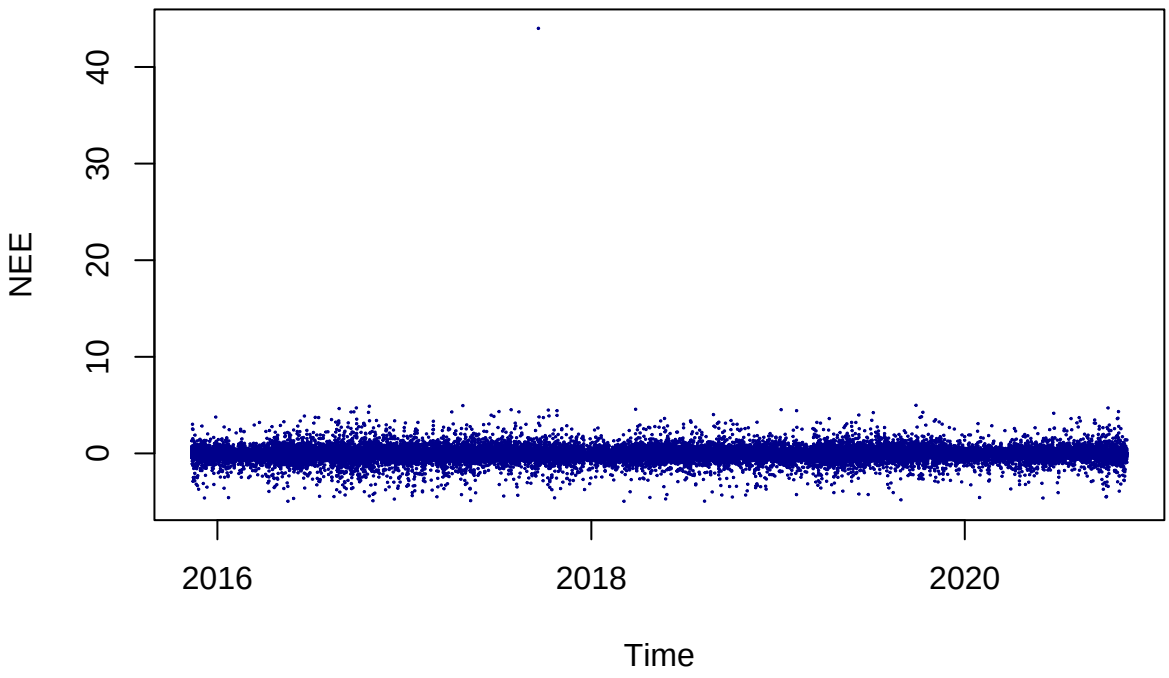
Friction velocity, or u_*, varies seasonally. Thus, the u_*-threshold needs to be estimated for each season. We do this because u_* changes with surface cover.

However, at Muka Head, u_* should not change appreciably because the water surface condition was unvarying.

Optional: Viewing the Data

Plotting the NEE against time shows the trend.

```
plot(Muka_Head2$DateTime, Muka_Head2$NEE, pch=19, xlab = "Time", ylab = 'NEE', cex = 0.1, col = "darkblu
```



Step 3-2-2: Calculate the u_*-Thresholds Distributions

We will estimate the (u_*) limits using the `sEstimateUstarScenarios` function. The function will write to the data object. The `seasonFactor` is needed here to tell `REddyProc` the season intervals that it must estimate the u_* thresholds.

The u_* threshold estimation uses the `usEstUstarThreshold` function, which requires the `NEE`, `Tair`, and `seasonFactor`. The function returns the median value.

The u_*-Threshold Distributions

In this example, the u_*-threshold is estimated, using the `usEstUstarThreshold` function, `nSample` times, and the u_* limits are reported using the default quantiles of 5%, 50%, and 95%: The low, median, and high values of u_*-threholds.

The function adds data to the REddyProc object. It creates the u_* scenarios and place it in the object.

```
EProcMYMkH$sEstimateUstarScenarios(nSample = 100,
                                   probs = c(0.05,0.50,0.95))
```

```
##
```

```
## Estimated UStar distribution of:
```

```
##         uStar       5%        50%        95%
## 1 0.02047025 0.015288 0.01933912 0.02207603
## by using  100 bootstrap samples and controls:
##                          taClasses                       UstarClasses
##                                  7                                 20
##                              swThr              minRecordsWithinTemp
##                                 10                                100
##           minRecordsWithinSeason              minRecordsWithinYear
##                                160                               3000
## isUsingOneBigSeasonOnFewRecords
##                                1
```

Viewing the Results

The function `sGetEstimatedUstarThresholdDistribution` displays the results.

Useful functions for handling the data in the object:

1. `sExportData`: Export class internal `sDATA` data frame.
2. `sExportResults`: Export class internal `sTEMP` data frame with result columns. We can use this after gap-filling the data.

Note that you can create the plots of NEE versus (u_*) by using the function `sPlotNEEVersusUStarForSeason`.

```
EProcMYMkH$sGetEstimatedUstarThresholdDistribution()
```

```
##    aggregationMode seasonYear  season      uStar         5%        50%
## 1           single         NA    <NA> 0.02047025 0.01528800 0.01933912
## 2             year       2015    <NA> 0.02047025 0.01528800 0.01933912
## 3             year       2016    <NA> 0.02062139 0.01370249 0.01925271
## 4             year       2017    <NA> 0.01929567 0.01642366 0.02125099
## 5             year       2018    <NA> 0.02066897 0.01291633 0.02081543
## 6             year       2019    <NA> 0.02047025 0.01303925 0.01963183
## 7             year       2020    <NA> 0.01393319 0.01134935 0.01457000
## 8           season       2015 2015009 0.02047025 0.01528800 0.01933912
## 9           season       2016 2015012 0.02062139 0.01470516 0.01950048
## 10          season       2016 2016003 0.02062139 0.01370249 0.01925271
## 11          season       2016 2016006 0.02062139 0.01370249 0.01925271
## 12          season       2016 2016009 0.01779098 0.01048099 0.01714230
## 13          season       2017 2016012 0.01929567 0.01641918 0.01999067
## 14          season       2017 2017003 0.01817400 0.01267184 0.01796974
## 15          season       2017 2017006 0.01429328 0.01281235 0.01779714
## 16          season       2017 2017009 0.01871804 0.01479314 0.01959153
## 17          season       2018 2017012 0.02066897 0.01453023 0.02004616
## 18          season       2018 2018003 0.02066897 0.01291633 0.02081543
## 19          season       2018 2018006 0.02066897 0.01194918 0.01984176
## 20          season       2018 2018009 0.01081017 0.01130363 0.02065342
## 21          season       2019 2018012 0.02047025 0.01375993 0.01996184
## 22          season       2019 2019003 0.02047025 0.01303925 0.01963183
## 23          season       2019 2019006 0.01865939 0.01389917 0.01916076
## 24          season       2019 2019009 0.02047025 0.01303925 0.01963183
## 25          season       2020 2019012 0.02491567 0.01504616 0.02256483
```

```
## 26          season       2020 2020003 0.01612548 0.01395017 0.01606421
## 27          season       2020 2020006 0.01393319 0.01134935 0.01457000
## 28          season       2020 2020009 0.01393319 0.01134935 0.01457000
##           95%
## 1  0.02207603
## 2  0.02207603
## 3  0.02311499
## 4  0.02518152
## 5  0.02422442
## 6  0.02447745
## 7  0.01833482
## 8  0.02207603
## 9  0.02356814
## 10 0.02311499
## 11 0.02311499
## 12 0.02069243
## 13 0.02440668
## 14 0.02352896
## 15 0.02296223
## 16 0.02471328
## 17 0.02301459
## 18 0.02422442
## 19 0.02393608
## 20 0.02424592
## 21 0.02393556
## 22 0.02447745
## 23 0.02100453
## 24 0.02447745
## 25 0.02646330
## 26 0.01801804
## 27 0.01833482
## 28 0.01833482
```

```
#EProcMYMkH$sPlotNEEVersusUStarForSeason(season = '2016006')
```

Step 4-1: Gap-Filling the Data

Step 4-1-1: Check the Use of Seasonal u*-Thresholds

First, we have to ensure the use of seasonal u_*-thresholds. If it is not set in the previous step, check that it is used now.

Show the default thresholds: annual

```
EProcMYMkH$sGetUstarScenarios()
```

```
##     season      uStar        U05        U50        U95
## 1  2015009 0.02047025 0.01528800 0.01933912 0.02207603
## 2  2015012 0.02062139 0.01370249 0.01925271 0.02311499
## 3  2016003 0.02062139 0.01370249 0.01925271 0.02311499
## 4  2016006 0.02062139 0.01370249 0.01925271 0.02311499
## 5  2016009 0.02062139 0.01370249 0.01925271 0.02311499
```

```
## 6  2016012 0.01929567 0.01642366 0.02125099 0.02518152
## 7  2017003 0.01929567 0.01642366 0.02125099 0.02518152
## 8  2017006 0.01929567 0.01642366 0.02125099 0.02518152
## 9  2017009 0.01929567 0.01642366 0.02125099 0.02518152
## 10 2017012 0.02066897 0.01291633 0.02081543 0.02422442
## 11 2018003 0.02066897 0.01291633 0.02081543 0.02422442
## 12 2018006 0.02066897 0.01291633 0.02081543 0.02422442
## 13 2018009 0.02066897 0.01291633 0.02081543 0.02422442
## 14 2018012 0.02047025 0.01303925 0.01963183 0.02447745
## 15 2019003 0.02047025 0.01303925 0.01963183 0.02447745
## 16 2019006 0.02047025 0.01303925 0.01963183 0.02447745
## 17 2019009 0.02047025 0.01303925 0.01963183 0.02447745
## 18 2019012 0.01393319 0.01134935 0.01457000 0.01833482
## 19 2020003 0.01393319 0.01134935 0.01457000 0.01833482
## 20 2020006 0.01393319 0.01134935 0.01457000 0.01833482
## 21 2020009 0.01393319 0.01134935 0.01457000 0.01833482
```

Instruct REddyProc to use the seasonal thresholds.

```
EProcMYMkH$useSeaonsalUStarThresholds()
```

Confirm that the seasonal thresholds are used by displaying it.

```
EProcMYMkH$sGetUstarScenarios()
```

```
##     season      uStar        U05        U50        U95
## 8  2015009 0.02047025 0.01528800 0.01933912 0.02207603
## 9  2015012 0.02062139 0.01470516 0.01950048 0.02356814
## 10 2016003 0.02062139 0.01370249 0.01925271 0.02311499
## 11 2016006 0.02062139 0.01370249 0.01925271 0.02311499
## 12 2016009 0.01779098 0.01048099 0.01714230 0.02069243
## 13 2016012 0.01929567 0.01641918 0.01999067 0.02440668
## 14 2017003 0.01817400 0.01267184 0.01796974 0.02352896
## 15 2017006 0.01429328 0.01281235 0.01779714 0.02296223
## 16 2017009 0.01871804 0.01479314 0.01959153 0.02471328
## 17 2017012 0.02066897 0.01453023 0.02004616 0.02301459
## 18 2018003 0.02066897 0.01291633 0.02081543 0.02422442
## 19 2018006 0.02066897 0.01194918 0.01984176 0.02393608
## 20 2018009 0.01081017 0.01130363 0.02065342 0.02424592
## 21 2018012 0.02047025 0.01375993 0.01996184 0.02393556
## 22 2019003 0.02047025 0.01303925 0.01963183 0.02447745
## 23 2019006 0.01865939 0.01389917 0.01916076 0.02100453
## 24 2019009 0.02047025 0.01303925 0.01963183 0.02447745
## 25 2019012 0.02491567 0.01504616 0.02256483 0.02646330
## 26 2020003 0.01612548 0.01395017 0.01606421 0.01801804
## 27 2020006 0.01393319 0.01134935 0.01457000 0.01833482
## 28 2020009 0.01393319 0.01134935 0.01457000 0.01833482
```

Step 4-1-2: Gap-Fill the Data

Gap-fill the data using the function `sMDSGapFillUStarScens`. It will filter the data using the u_*-thresholds and gap-fill it.

`MDS` means Marginal Distribution Sampling, which combines:

1. the Look Up Table (LUT)
2. Mean Diurnal Course (MDC)

Quality flags are created for the gap-filled data:

- 0: original data
- 1: good quality gap-filled data, i.e., *more parameters* and *shorter time-windows* used.
- More than 1: low quality, i.e., *less parameters* and *longer time-windows* used.

The function also calculates the random error for non-gap records by replacing the original values with gap-filled values.

```
EProcMYMkH$sMDSGapFillUStarScens("NEE", FillAll = TRUE)
```

Check the New Columns

Check the columns created. Examples are:

- `NEE_05_f`
- `NEE_95_fall`
- `NEE_50_fqc`

Definitions:

- NEE___f: gaps replaced by modeled values (gap-filled).
- NEE___fall: all NEE replaced by modeled values.
- NEE___fqc: quality flag: 0 observations, 1 good quality of gap-filling.
- The non-bootstrapped data has the `uStar` suffix.
- The bootstrapped data has the scenario suffix, e.g., `U50`, `U95`, etc.

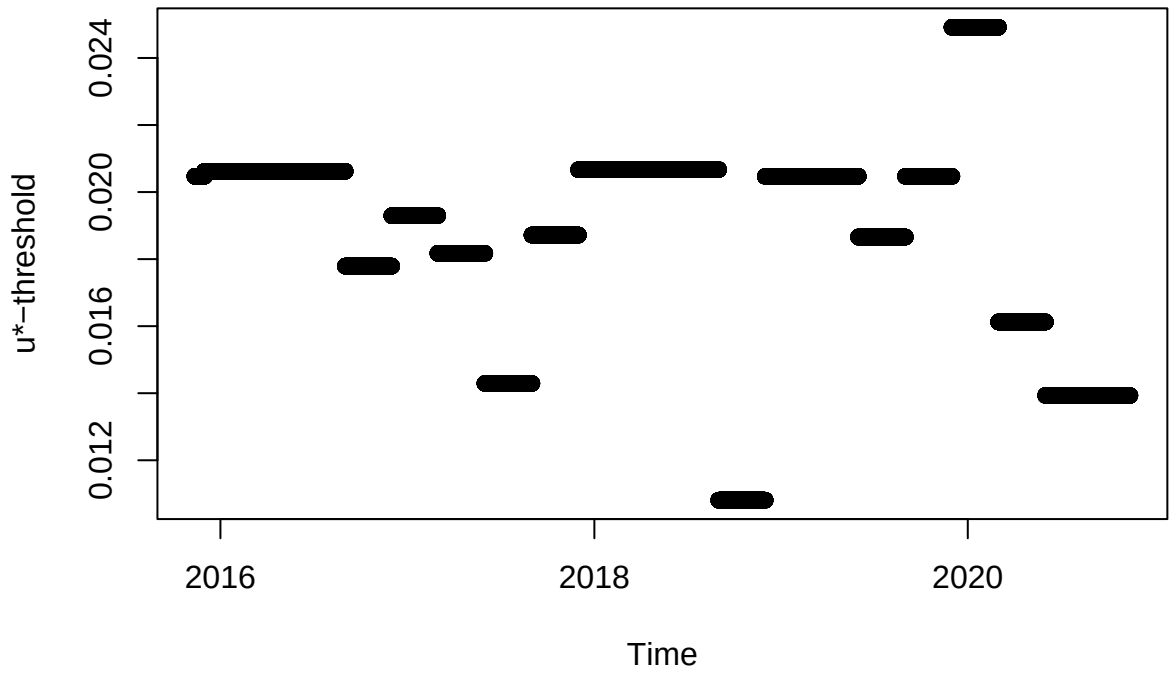
```
colnames(EProcMYMkH$sExportResults())
```

```
##  [1] "season"            "Ustar_uStar_Thres" "Ustar_uStar_fqc"
##  [4] "NEE_uStar_orig"    "NEE_uStar_f"       "NEE_uStar_fqc"
##  [7] "NEE_uStar_fall"    "NEE_uStar_fall_qc" "NEE_uStar_fnum"
## [10] "NEE_uStar_fsd"     "NEE_uStar_fmeth"   "NEE_uStar_fwin"
## [13] "Ustar_U05_Thres"   "Ustar_U05_fqc"     "NEE_U05_orig"
## [16] "NEE_U05_f"         "NEE_U05_fqc"       "NEE_U05_fall"
## [19] "NEE_U05_fall_qc"   "NEE_U05_fnum"      "NEE_U05_fsd"
## [22] "NEE_U05_fmeth"     "NEE_U05_fwin"      "Ustar_U50_Thres"
## [25] "Ustar_U50_fqc"     "NEE_U50_orig"      "NEE_U50_f"
## [28] "NEE_U50_fqc"       "NEE_U50_fall"      "NEE_U50_fall_qc"
## [31] "NEE_U50_fnum"      "NEE_U50_fsd"       "NEE_U50_fmeth"
## [34] "NEE_U50_fwin"      "Ustar_U95_Thres"   "Ustar_U95_fqc"
## [37] "NEE_U95_orig"      "NEE_U95_f"         "NEE_U95_fqc"
## [40] "NEE_U95_fall"      "NEE_U95_fall_qc"   "NEE_U95_fnum"
## [43] "NEE_U95_fsd"       "NEE_U95_fmeth"     "NEE_U95_fwin"
```

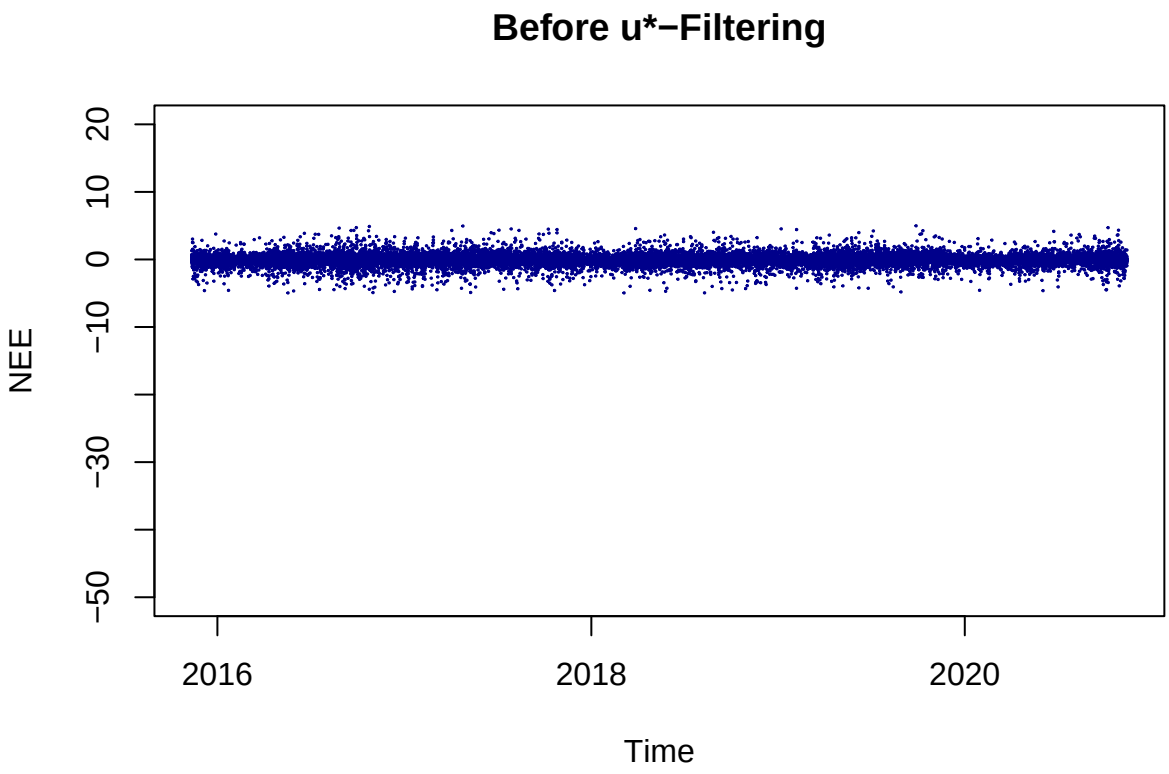
View Some Columns

Plotting a column of the REddyProc object.

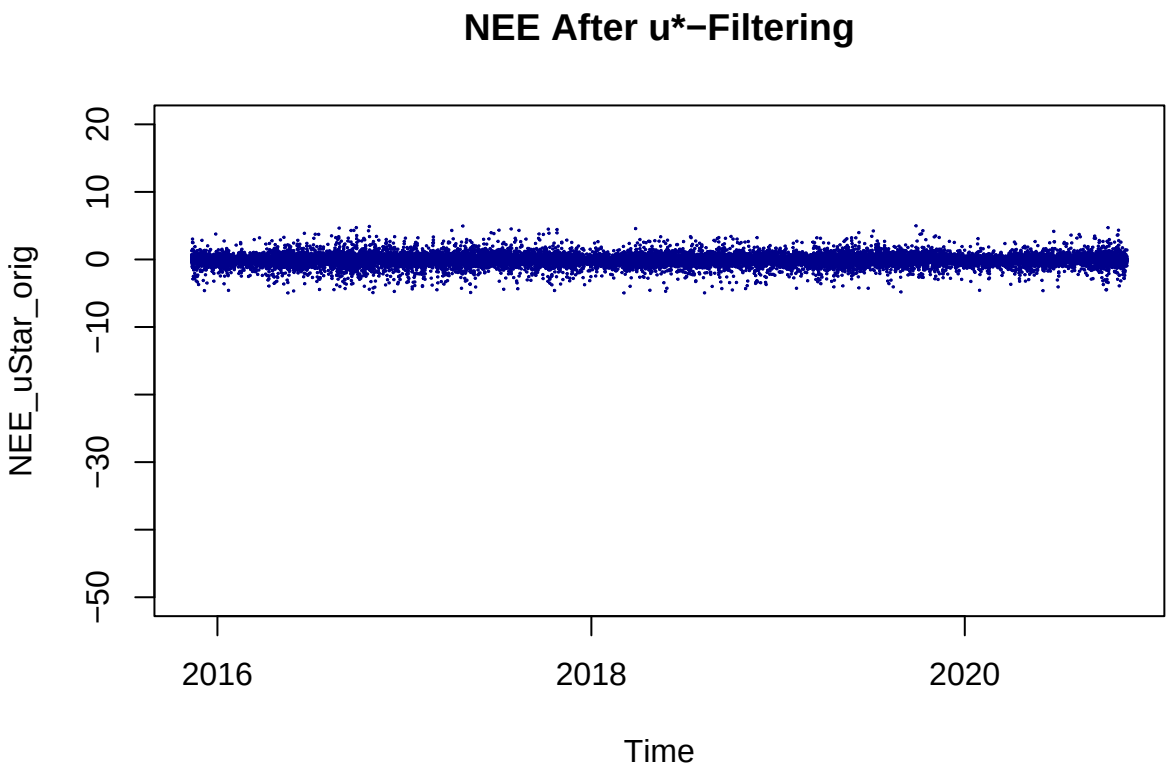
```
plot(EProcMYMkH$sDATA$sDateTime, EProcMYMkH$sExportResults()$Ustar_uStar_Thres, pch = 19,
     xlab = 'Time', ylab = 'u*-threshold')
```



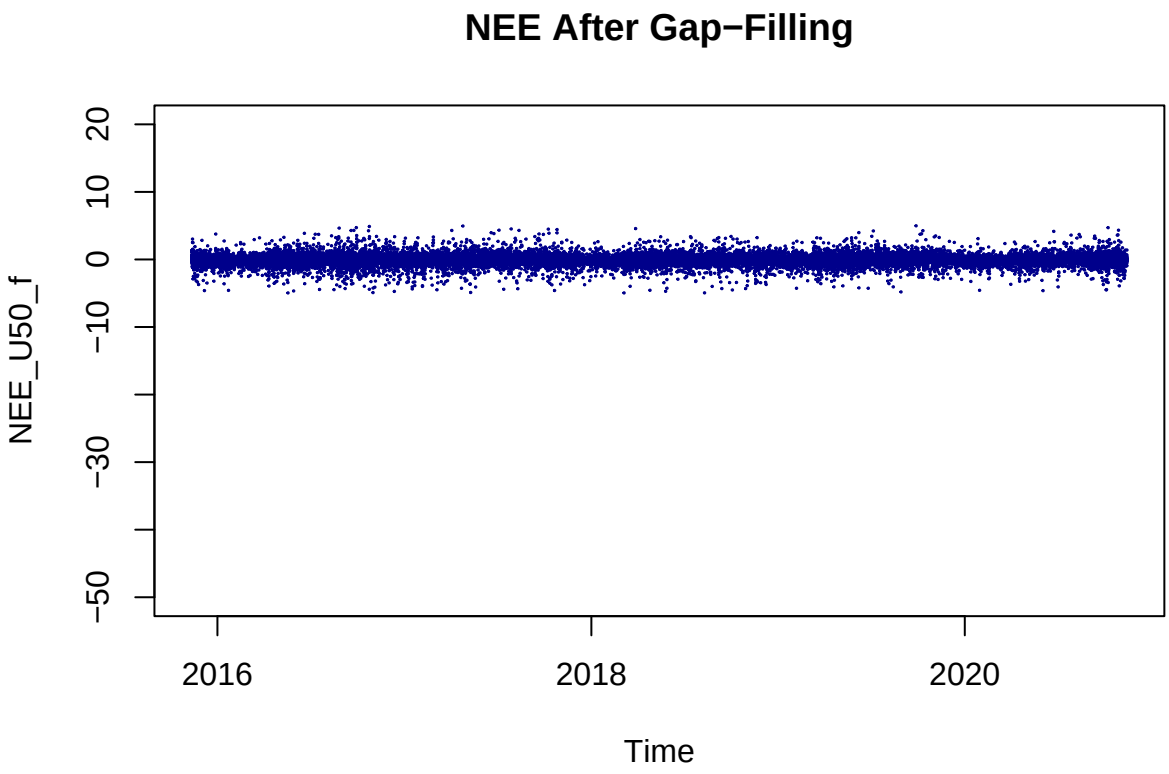

```
plot(Muka_Head2$DateTime,Muka_Head2$NEE, pch = 19, cex = 0.1, col = "darkblue",
     xlab = 'Time', ylab = 'NEE', ylim=c(-50,20), main = "Before u*-Filtering")
```



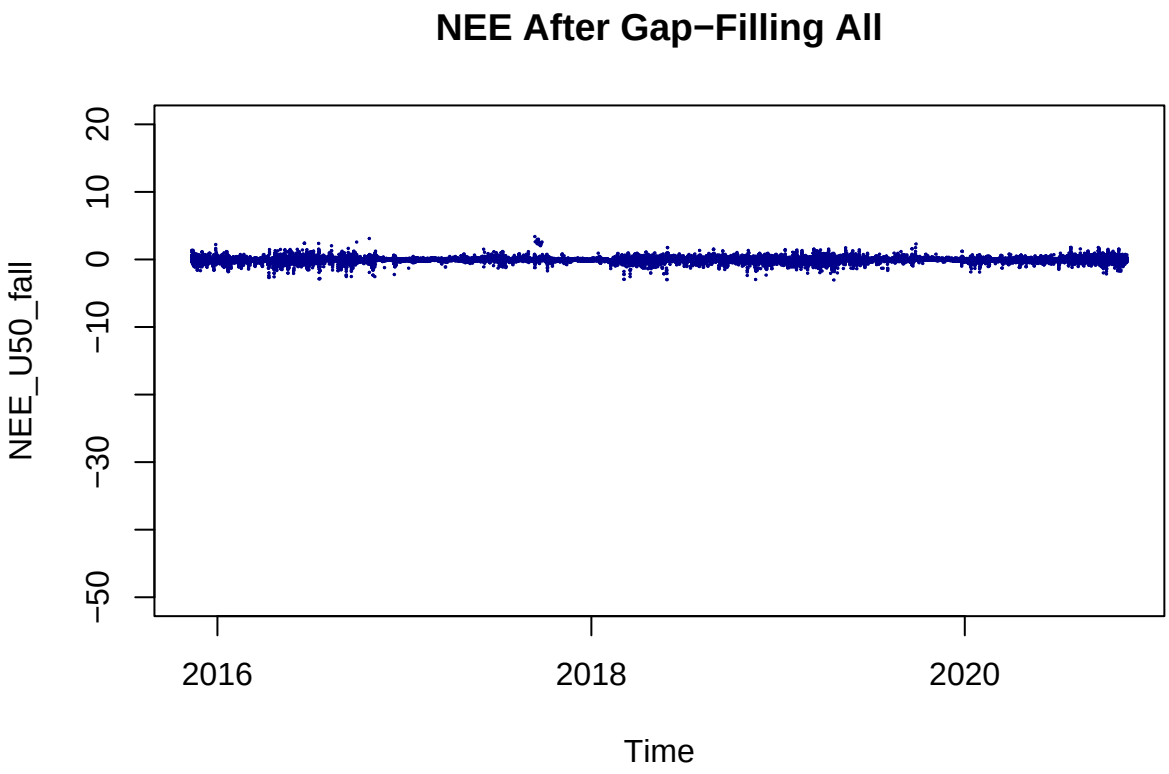

```
plot(EProcMYMkH$sDATA$sDateTime, EProcMYMkH$sExportResults()$NEE_uStar_orig, pch = 19,
     cex = 0.1, col = "darkblue",
     xlab = 'Time', ylab = 'NEE_uStar_orig', ylim=c(-50,20),
     main = "NEE After u*-Filtering")
```




```
plot(EProcMYMkH$sDATA$sDateTime, EProcMYMkH$sExportResults()$NEE_U50_f, pch = 19,
     cex = 0.1, col = "darkblue",
     xlab = 'Time', ylab = 'NEE_U50_f', ylim=c(-50,20),
     main = "NEE After Gap-Filling")
```



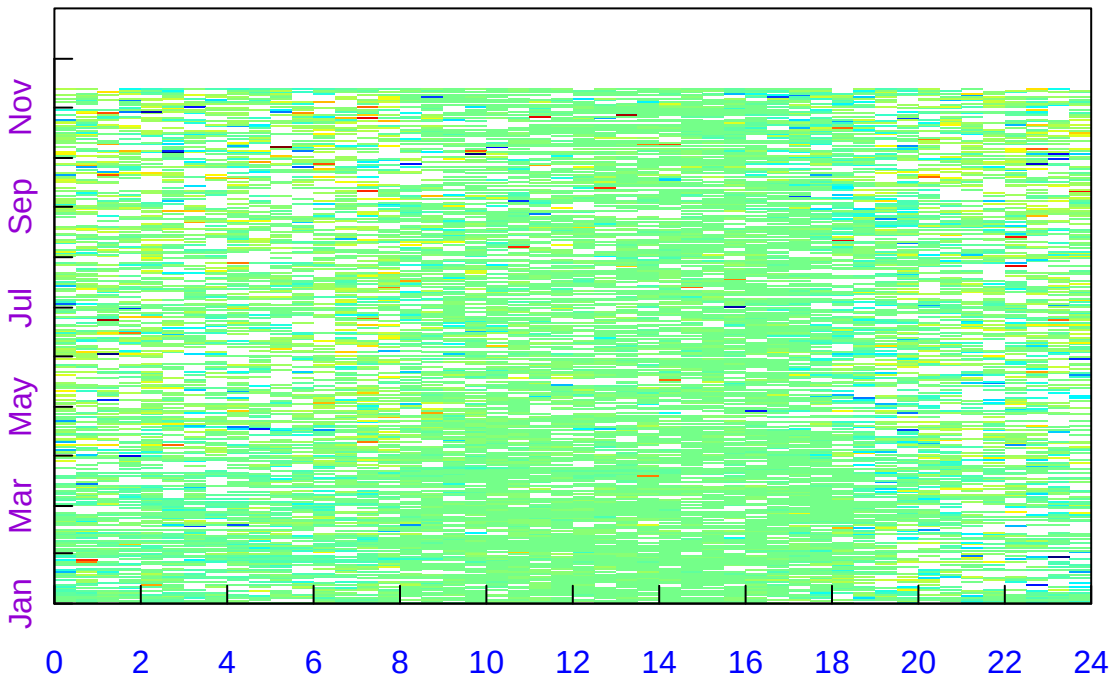

```
plot(EProcMYMkH$sDATA$sDateTime, EProcMYMkH$sExportResults()$NEE_U50_fall, pch = 19,
     cex = 0.1, col = "darkblue",
     xlab = 'Time', ylab = 'NEE_U50_fall', ylim=c(-50,20),
     main = "NEE After Gap-Filling All")
```



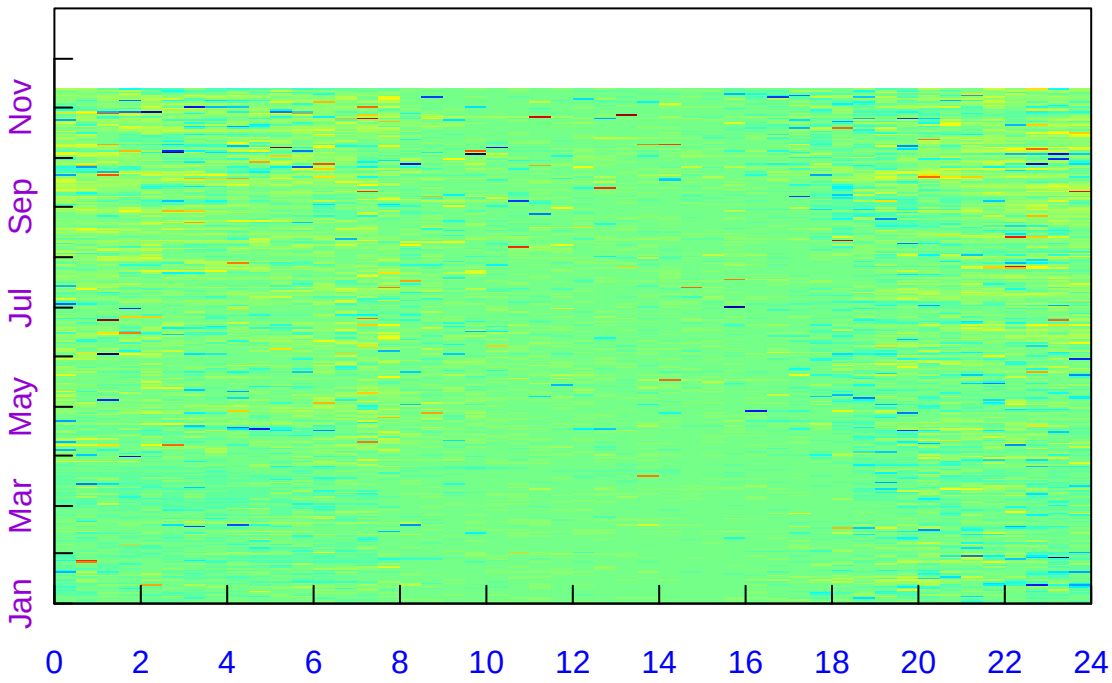
Step 4-1-3: Fingerprint Plot

We can also generate a fingerprint plot using the function `sPlotFingerprintY`. This is for the `NEE_U50_f` parameter and the year 2004.

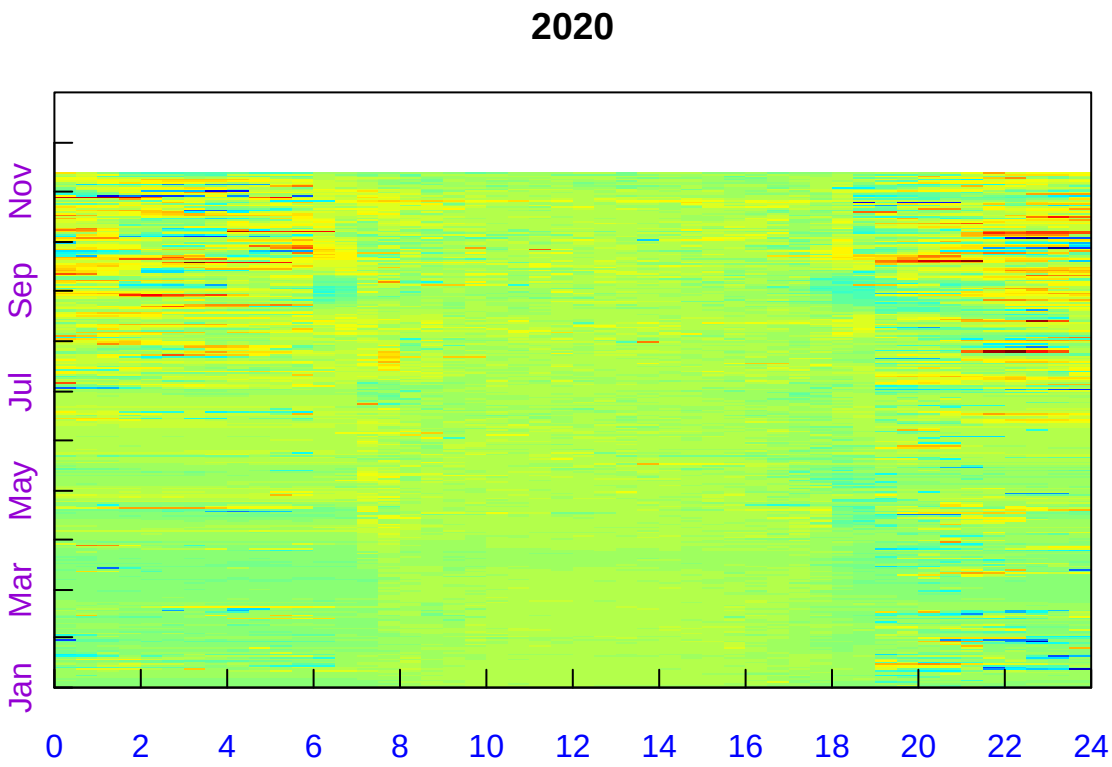
```
EProcMYMkH$sPlotFingerprintY('NEE_uStar_orig', Year = 2020)
```




```
EProcMYMkH$sPlotFingerprintY('NEE_U50_f', Year = 2020)
```




```
EProcMYMkH$sPlotFingerprintY('NEE_U50_fall', Year = 2020)
```



We can also produce PDF files with legend for all years in sub-directory "figs."

```
EProcMYMkH$sPlotFingerprint('NEE_U50_f', Dir = "figs")
```

```
## Saved plot to: figs/MY-MkH_15-20_FP_NEE_U50_f.pdf
```

Step 5-1: Preparing the Data for Partitioning

This step requires the data to have the location (lat, lon) and time zone info because REddyProc uses time to estimate day and night hours. We already did this in the *Step 2*.

There are some weather values that are missing and can be gap-filled here. However, we do not need to replace the original values with gap-filled values because we are not going to calculate random error, `FillAll = FALSE`.

```
EProcMYMkH$sMDSGapFill('Rg', FillAll = FALSE)
EProcMYMkH$sMDSGapFill('Tair', FillAll = FALSE)
EProcMYMkH$sMDSGapFill('VPD', FillAll = FALSE)
```

Step 5-1-1:Reichstein Partitioning

In this part, we will partition the data into fractions of the Gross Primary Production (GPP) and ecosystem respiration (R_{eco}) using all u_* scenarios. This uses the 'sMRFluxPartitionUStarScens' function.

Results are added to the object.

More details on the equations used can be found in the paper Reichstein et al. (2005).

```
EProcMYMkH$sMRFluxPartitionUStarScens()
```

```
## Warning in fRegrE0fromShortTerm(NightFlux, Temp, DayCounter, ..., MinE0 = MinE0, : sMRFluxPartition:
## You may try relaxing the temperature range constraint by setting TempRange = 3 (instead of default 5

## Warning in fRegrE0fromShortTerm(NightFlux, Temp, DayCounter, ..., MinE0 = MinE0, : sMRFluxPartition:
## You may try relaxing the temperature range constraint by setting TempRange = 3 (instead of default 5

## Warning in fRegrE0fromShortTerm(NightFlux, Temp, DayCounter, ..., MinE0 = MinE0, : sMRFluxPartition:
## You may try relaxing the temperature range constraint by setting TempRange = 3 (instead of default 5

## Warning in fRegrE0fromShortTerm(NightFlux, Temp, DayCounter, ..., MinE0 = MinE0, : sMRFluxPartition:
## You may try relaxing the temperature range constraint by setting TempRange = 3 (instead of default 5
```

Step 5-1-2: Plotting the GPP

View the result columns. Columns [46] to [104] are added.

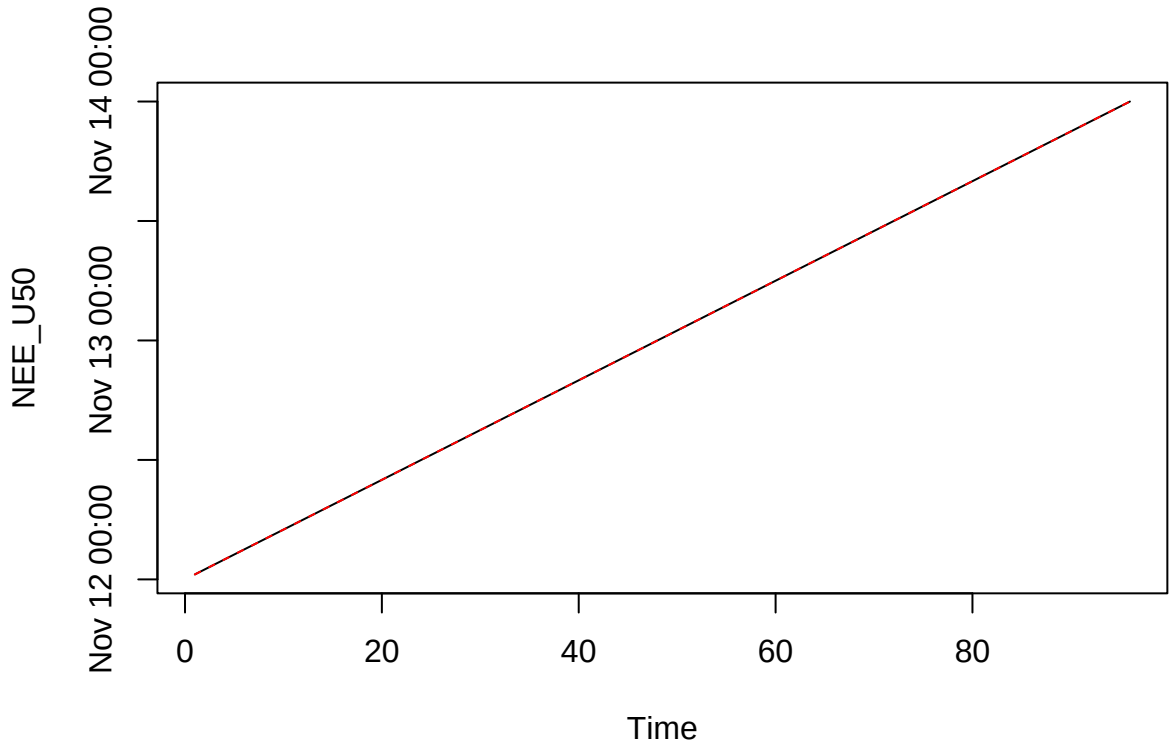
```
names(EProcMYMkH$sExportResults())
```

```
##  [1] "season"            "Ustar_uStar_Thres" "Ustar_uStar_fqc"
##  [4] "NEE_uStar_orig"    "NEE_uStar_f"       "NEE_uStar_fqc"
##  [7] "NEE_uStar_fall"    "NEE_uStar_fall_qc" "NEE_uStar_fnum"
## [10] "NEE_uStar_fsd"     "NEE_uStar_fmeth"   "NEE_uStar_fwin"
## [13] "Ustar_U05_Thres"   "Ustar_U05_fqc"     "NEE_U05_orig"
## [16] "NEE_U05_f"         "NEE_U05_fqc"       "NEE_U05_fall"
## [19] "NEE_U05_fall_qc"   "NEE_U05_fnum"      "NEE_U05_fsd"
## [22] "NEE_U05_fmeth"     "NEE_U05_fwin"      "Ustar_U50_Thres"
## [25] "Ustar_U50_fqc"     "NEE_U50_orig"      "NEE_U50_f"
## [28] "NEE_U50_fqc"       "NEE_U50_fall"      "NEE_U50_fall_qc"
## [31] "NEE_U50_fnum"      "NEE_U50_fsd"       "NEE_U50_fmeth"
## [34] "NEE_U50_fwin"      "Ustar_U95_Thres"   "Ustar_U95_fqc"
## [37] "NEE_U95_orig"      "NEE_U95_f"         "NEE_U95_fqc"
## [40] "NEE_U95_fall"      "NEE_U95_fall_qc"   "NEE_U95_fnum"
## [43] "NEE_U95_fsd"       "NEE_U95_fmeth"     "NEE_U95_fwin"
## [46] "Rg_orig"           "Rg_f"              "Rg_fqc"
## [49] "Rg_fall"           "Rg_fall_qc"        "Rg_fnum"
## [52] "Rg_fsd"            "Rg_fmeth"          "Rg_fwin"
## [55] "Tair_orig"         "Tair_f"            "Tair_fqc"
## [58] "Tair_fall"         "Tair_fall_qc"      "Tair_fnum"
## [61] "Tair_fsd"          "Tair_fmeth"        "Tair_fwin"
## [64] "VPD_orig"          "VPD_f"             "VPD_fqc"
## [67] "VPD_fall"          "VPD_fall_qc"       "VPD_fnum"
## [70] "VPD_fsd"           "VPD_fmeth"         "VPD_fwin"
## [73] "PotRad_NEW"        "FP_VARnight"       "FP_Temp_NEW"
## [76] "E_0_NEW"
```

Plot the GPP and Reco

Plot the GPP and Reco for `U50` scenario against time for two days (`48*2`).

```
nRec = 48*2
plot(head(Muka_Head2$DateTime, nRec),
     head(EProcMYMkH$sExportResults()$GPP_U50_f, nRec),
     type = "l", xlab = 'Time', ylab = 'NEE_U50')
lines(head(Muka_Head2$DateTime, nRec),
      head(EProcMYMkH$sExportResults()$Reco_U50, nRec),
      type = "l", lty = 2,
      col = 'red')
```



Step 5-1-3:Lasslop Partitioning

Partitioning the data into the fractions of the Gross Primary Production (GPP) and ecosystem respiration (R_{eco}) using all u_* scenarios. This uses the 'sGLFluxPartitionUStarScens' function.

Results are added to the object.

More details on the equations used can be found in the Lasslop et al. (2010).

```
EProcMYMkH$sGLFluxPartitionUStarScens()
```

```
## Warning in sqrt(dsAssoc$wBefore^2 * varPred2[[1]][, "varGPP"] +
```

```
## dsAssoc$wAfter^2 * : NaNs produced

## Warning in sqrt(dsAssoc$wBefore^2 * varPred2[[1]][, "varGPP"] +
## dsAssoc$wAfter^2 * : NaNs produced

## Warning in sqrt(dsAssoc$wBefore^2 * varPred2[[1]][, "varGPP"] +
## dsAssoc$wAfter^2 * : NaNs produced

## Warning in sqrt(dsAssoc$wBefore^2 * varPred2[[1]][, "varGPP"] +
## dsAssoc$wAfter^2 * : NaNs produced
```

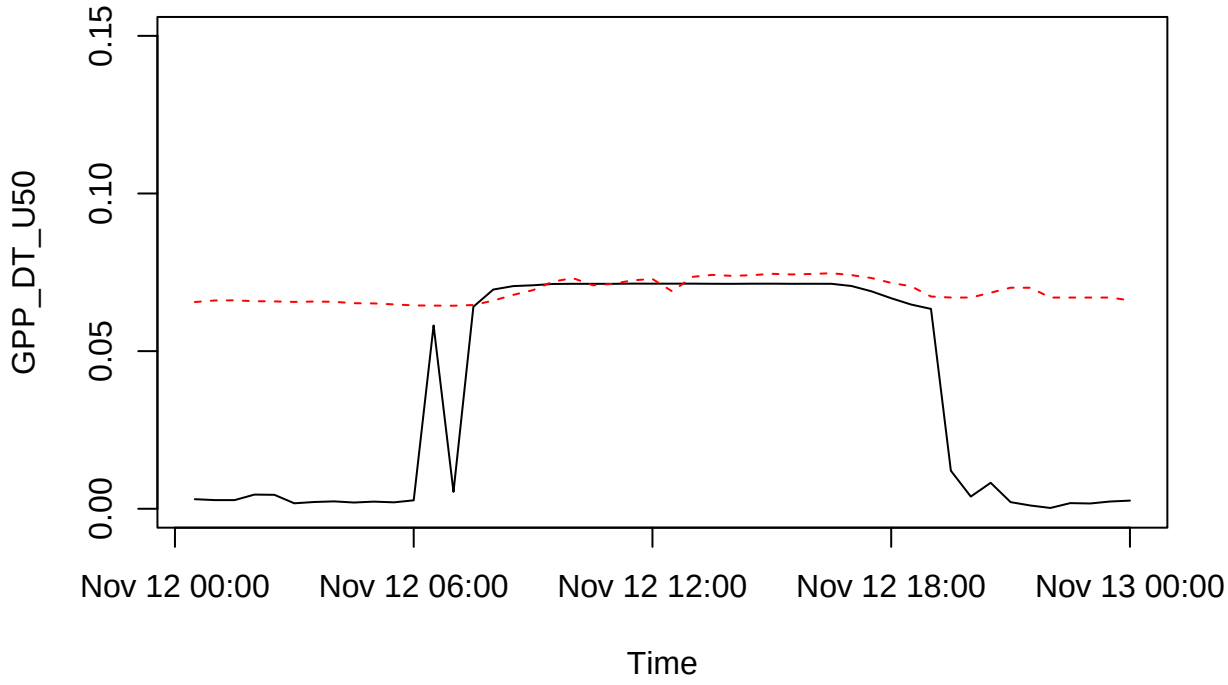
View the result columns. Columns [105] to [140] are added.

```
names(EProcMYMkH$sExportResults())
```

```
##   [1] "season"            "Ustar_uStar_Thres" "Ustar_uStar_fqc"
##   [4] "NEE_uStar_orig"    "NEE_uStar_f"       "NEE_uStar_fqc"
##   [7] "NEE_uStar_fall"    "NEE_uStar_fall_qc" "NEE_uStar_fnum"
##  [10] "NEE_uStar_fsd"     "NEE_uStar_fmeth"   "NEE_uStar_fwin"
##  [13] "Ustar_U05_Thres"   "Ustar_U05_fqc"     "NEE_U05_orig"
##  [16] "NEE_U05_f"         "NEE_U05_fqc"       "NEE_U05_fall"
##  [19] "NEE_U05_fall_qc"   "NEE_U05_fnum"      "NEE_U05_fsd"
##  [22] "NEE_U05_fmeth"     "NEE_U05_fwin"      "Ustar_U50_Thres"
##  [25] "Ustar_U50_fqc"     "NEE_U50_orig"      "NEE_U50_f"
##  [28] "NEE_U50_fqc"       "NEE_U50_fall"      "NEE_U50_fall_qc"
##  [31] "NEE_U50_fnum"      "NEE_U50_fsd"       "NEE_U50_fmeth"
##  [34] "NEE_U50_fwin"      "Ustar_U95_Thres"   "Ustar_U95_fqc"
##  [37] "NEE_U95_orig"      "NEE_U95_f"         "NEE_U95_fqc"
##  [40] "NEE_U95_fall"      "NEE_U95_fall_qc"   "NEE_U95_fnum"
##  [43] "NEE_U95_fsd"       "NEE_U95_fmeth"     "NEE_U95_fwin"
##  [46] "Rg_orig"           "Rg_f"              "Rg_fqc"
##  [49] "Rg_fall"           "Rg_fall_qc"        "Rg_fnum"
##  [52] "Rg_fsd"            "Rg_fmeth"          "Rg_fwin"
##  [55] "Tair_orig"         "Tair_f"            "Tair_fqc"
##  [58] "Tair_fall"         "Tair_fall_qc"      "Tair_fnum"
##  [61] "Tair_fsd"          "Tair_fmeth"        "Tair_fwin"
##  [64] "VPD_orig"          "VPD_f"             "VPD_fqc"
##  [67] "VPD_fall"          "VPD_fall_qc"       "VPD_fnum"
##  [70] "VPD_fsd"           "VPD_fmeth"         "VPD_fwin"
##  [73] "PotRad_NEW"        "FP_Temp_NEW"       "E_0_NEW"
##  [76] "Reco_DT_U05"       "GPP_DT_U05"        "Reco_DT_U05_SD"
##  [79] "GPP_DT_U05_SD"     "Reco_DT_U50"       "GPP_DT_U50"
##  [82] "Reco_DT_U50_SD"    "GPP_DT_U50_SD"     "Reco_DT_U95"
##  [85] "GPP_DT_U95"        "Reco_DT_U95_SD"    "GPP_DT_U95_SD"
##  [88] "FP_VARnight"       "FP_VARday"         "NEW_FP_Temp"
##  [91] "NEW_FP_VPD"        "FP_RRef_Night"     "FP_qc"
##  [94] "FP_dRecPar"        "FP_errorcode"      "FP_GPP2000"
##  [97] "FP_k"              "FP_beta"           "FP_alpha"
## [100] "FP_RRef"           "FP_E0"             "FP_k_sd"
## [103] "FP_beta_sd"        "FP_alpha_sd"       "FP_RRef_sd"
## [106] "FP_E0_sd"          "Reco_DT_uStar"     "GPP_DT_uStar"
## [109] "Reco_DT_uStar_SD"  "GPP_DT_uStar_SD"
```

Plot the GPP and Reco for `U50` scenario against time for two days (`48*2`).

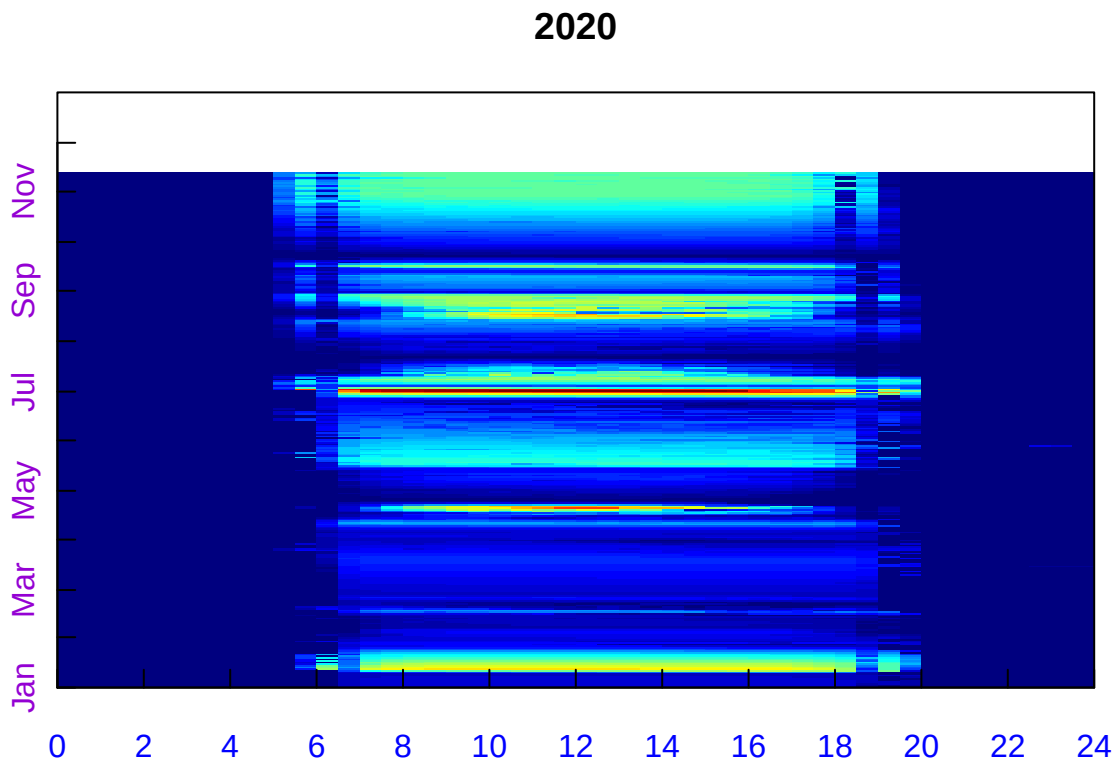
```
nRec <- 48*1
plot(head(Muka_Head2$DateTime, nRec),
     head(EProcMYMkH$sExportResults()$GPP_DT_U50, nRec),
     type = "l", xlab = 'Time', ylab = 'GPP_DT_U50', ylim = c(0, 0.15))
lines(head(Muka_Head2$DateTime, nRec),
      head(EProcMYMkH$sExportResults()$Reco_DT_U50, nRec),
      type = "l", lty = 2, col = 'red')
```



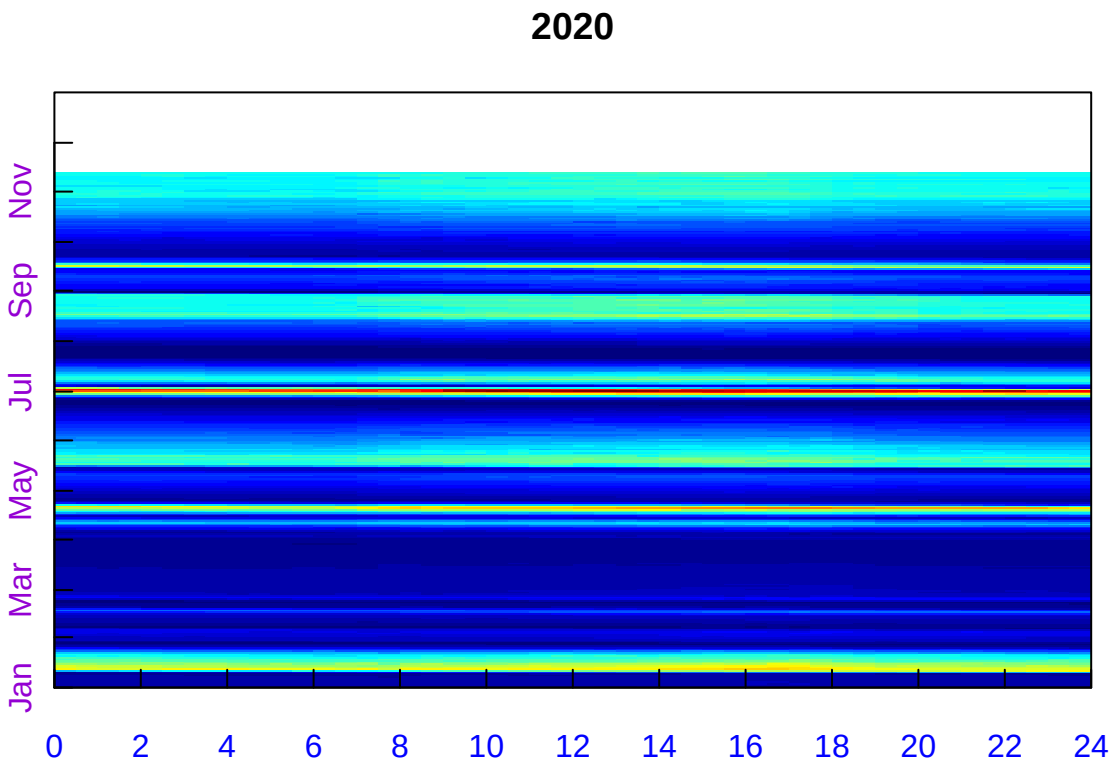
Step 5-1-4: Fingerprint Plots of GPP_DT and Reco_DT

The fingerprint plots can be plotted for the GPP and R_{eco}.

```
EProcMYMkH$sPlotFingerprintY('GPP_DT_U50', Year = 2020)
```




```
EProcMYMkH$sPlotFingerprintY('Reco_DT_U50', Year = 2020)
```




```
EProcMYMkH$sPlotFingerprint('GPP_DT_U05', Dir = "figs")
```

```
## Saved plot to: figs/MY-MkH_15-20_FP_GPP_DT_U05.pdf
```

```
EProcMYMkH$sPlotFingerprint('Reco_DT_U05', Dir = "figs")
```

```
## Saved plot to: figs/MY-MkH_15-20_FP_Reco_DT_U05.pdf
```

Step 5-1-5: Export the Results

This part will produce a text file for analysis outside of R. It will be placed in the folder `results`.

```
MkHData <- EProcMYMkH$sExportData() # Write the original data to MkHData.
MkHResults <- EProcMYMkH$sExportResults() # Write the results of REddyProc to MkHResults.
MkHCombResults <- cbind(MkHData, MkHResults)
fWriteDataframeToFile(MkHCombResults, "MY-MkH_Part.txt", Dir = "results")
```

```
## Number of NA convertered to '-9999': 2558017
```

```
## Wrote tab separated textfile: results/MY-MkH_Part.txt
```

Step 6-1: Bias with u_*-Threshold

Calculating the Bias for the Year 2016

We will be calculating the bias of NEE due to the u_*-threshold for 2016.

Check the names of the columns of `MkHCombResults`.

```
names(MkHCombResults)
```

```
##   [1] "DateTime"          "NEE"               "Rg"
##   [4] "Tair"              "VPD"               "Ustar"
##   [7] "season"            "Ustar_uStar_Thres" "Ustar_uStar_fqc"
##  [10] "NEE_uStar_orig"    "NEE_uStar_f"       "NEE_uStar_fqc"
##  [13] "NEE_uStar_fall"    "NEE_uStar_fall_qc" "NEE_uStar_fnum"
##  [16] "NEE_uStar_fsd"     "NEE_uStar_fmeth"   "NEE_uStar_fwin"
##  [19] "Ustar_U05_Thres"   "Ustar_U05_fqc"     "NEE_U05_orig"
##  [22] "NEE_U05_f"         "NEE_U05_fqc"       "NEE_U05_fall"
##  [25] "NEE_U05_fall_qc"   "NEE_U05_fnum"      "NEE_U05_fsd"
##  [28] "NEE_U05_fmeth"     "NEE_U05_fwin"      "Ustar_U50_Thres"
##  [31] "Ustar_U50_fqc"     "NEE_U50_orig"      "NEE_U50_f"
##  [34] "NEE_U50_fqc"       "NEE_U50_fall"      "NEE_U50_fall_qc"
##  [37] "NEE_U50_fnum"      "NEE_U50_fsd"       "NEE_U50_fmeth"
##  [40] "NEE_U50_fwin"      "Ustar_U95_Thres"   "Ustar_U95_fqc"
##  [43] "NEE_U95_orig"      "NEE_U95_f"         "NEE_U95_fqc"
##  [46] "NEE_U95_fall"      "NEE_U95_fall_qc"   "NEE_U95_fnum"
##  [49] "NEE_U95_fsd"       "NEE_U95_fmeth"     "NEE_U95_fwin"
##  [52] "Rg_orig"           "Rg_f"              "Rg_fqc"
##  [55] "Rg_fall"           "Rg_fall_qc"        "Rg_fnum"
##  [58] "Rg_fsd"            "Rg_fmeth"          "Rg_fwin"
##  [61] "Tair_orig"         "Tair_f"            "Tair_fqc"
##  [64] "Tair_fall"         "Tair_fall_qc"      "Tair_fnum"
##  [67] "Tair_fsd"          "Tair_fmeth"        "Tair_fwin"
##  [70] "VPD_orig"          "VPD_f"             "VPD_fqc"
##  [73] "VPD_fall"          "VPD_fall_qc"       "VPD_fnum"
##  [76] "VPD_fsd"           "VPD_fmeth"         "VPD_fwin"
##  [79] "PotRad_NEW"        "FP_Temp_NEW"       "E_0_NEW"
##  [82] "Reco_DT_U05"       "GPP_DT_U05"        "Reco_DT_U05_SD"
##  [85] "GPP_DT_U05_SD"     "Reco_DT_U50"       "GPP_DT_U50"
##  [88] "Reco_DT_U50_SD"    "GPP_DT_U50_SD"     "Reco_DT_U95"
##  [91] "GPP_DT_U95"        "Reco_DT_U95_SD"    "GPP_DT_U95_SD"
##  [94] "FP_VARnight"       "FP_VARday"         "NEW_FP_Temp"
##  [97] "NEW_FP_VPD"        "FP_RRef_Night"     "FP_qc"
## [100] "FP_dRecPar"        "FP_errorcode"      "FP_GPP2000"
## [103] "FP_k"              "FP_beta"           "FP_alpha"
## [106] "FP_RRef"           "FP_E0"             "FP_k_sd"
## [109] "FP_beta_sd"        "FP_alpha_sd"       "FP_RRef_sd"
## [112] "FP_E0_sd"          "Reco_DT_uStar"     "GPP_DT_uStar"
## [115] "Reco_DT_uStar_SD"  "GPP_DT_uStar_SD"
```

Create a Factor Column to Distinguish the Year

First, create an integer column `year`.

```
MkHCombResults$year <- as.POSIXlt(MkHCombResults$DateTime)$year + 1900
str(MkHCombResults$year)
```

```
##  num [1:87744] 2015 2015 2015 2015 2015 ...
```

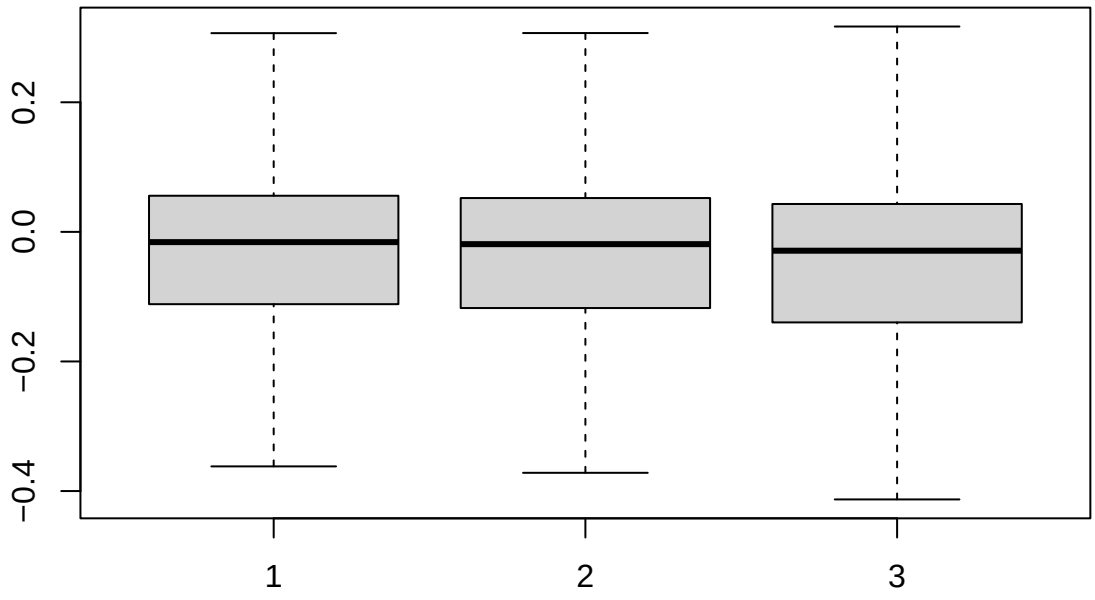
Create a subset data frame from the combined results.

```
MkH2017 <- subset(MkHCombResults, year == 2017)
```

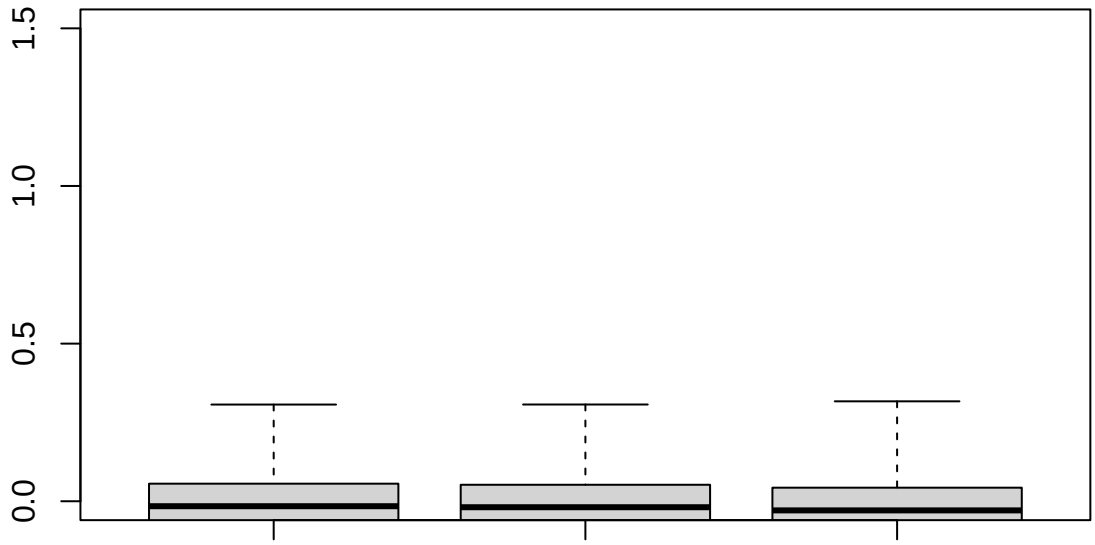
Visualize the Difference of NEE Among the Scenarios

Using a boxplot we can see the changes.

```
boxplot(MkH2017$NEE_U05_f, MkH2017$NEE_U50_f, MkH2017$NEE_U95_f, outline = FALSE)
```




```
boxplot(MkH2017$NEE_U05_f, MkH2017$NEE_U50_f, MkH2017$NEE_U95_f, outline = FALSE, ylim = c(0,1.5))
```



Step 6-1-1: Calculate the Annual Mean of NEE for each u_{*Th} Scenario for 2004

We will use the gap-filled 2017 data of the difference scenarios.

Create a variable that contains the means of the different scenarios: `U05`, `U50`, and `U95`.

```
MkHScenarios <- c("uStar","U05","U50","U95")
NEE_UStar <- sapply(MkHScenarios, function(suffix){
  colName = paste0("NEE_",suffix,"_f")
  mean(MkH2017[[colName]])
})
NEE_UStar
```

```
##       uStar         U05         U50         U95
## -0.02842798 -0.02398072 -0.02908520 -0.04444822
```

Step 6-1-2: Calculate the Statistics

Calculate the mean, standard deviation, and relative error.

```
c(mean(NEE_UStar), sd(NEE_UStar), sd(NEE_UStar)/abs(mean(NEE_UStar)))
```

```
## [1] -0.031485528  0.008934277  0.283758192
```

Step 7-1: Random Uncertainty Aggregation

Step 7-1-1: Gebesee Calculate Error Terms

To calculate the error, the replaced NEE, the NEE calculated using the gap-filling method or `NEE_uStar_fall`, is subtracted from the original NEE values `NEE_ustar_orig`. The resulting value is the residual.

The original number of non-bootstrapped data for all and 2004.

```
n_all <- sum(MkHCombResults$NEE_uStar_fqc == 0)
n_all
```

```
## [1] 55268
```

```
n_2017 <- sum(MkH2017$NEE_uStar_fqc == 0)
n_2017
```

```
## [1] 10894
```

The residuals are calculated for all the results and the year 2004 for comparison.

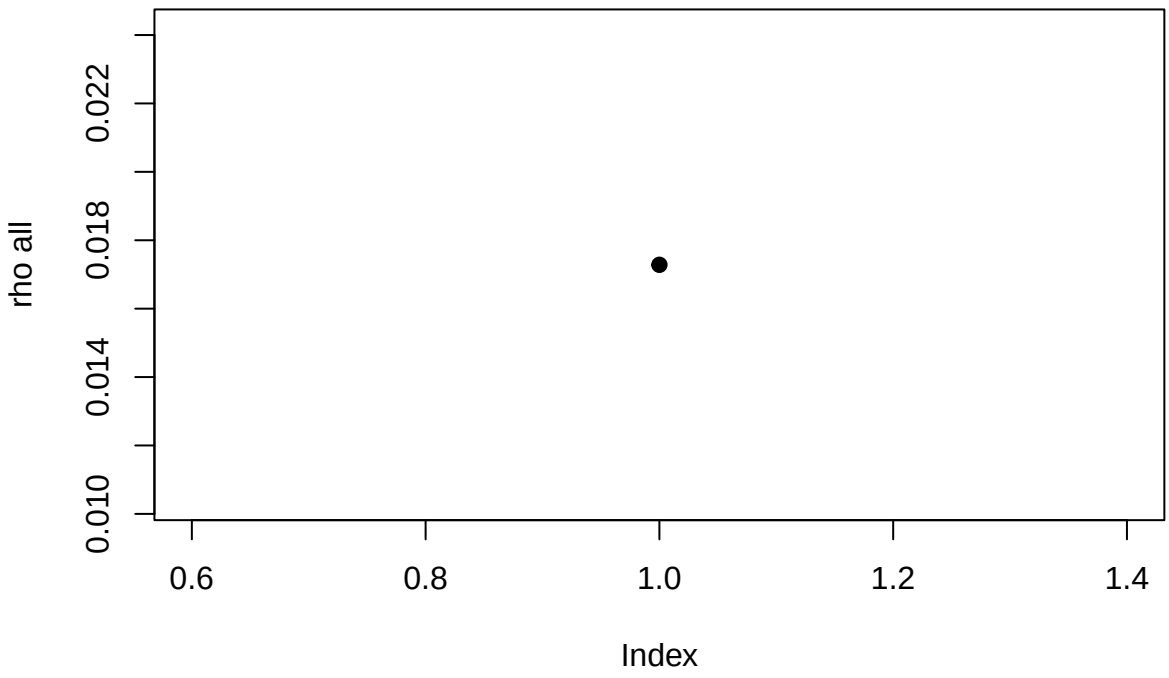
```
MkHCombResults$residual <- ifelse(MkHCombResults$NEE_uStar_fqc == 0,
                                  MkHCombResults$NEE_uStar_orig - MkHCombResults$NEE_uStar_fall,
                                  NA)

MkH2017$residual <- ifelse(MkH2017$NEE_uStar_fqc == 0,
                           MkH2017$NEE_uStar_orig - MkH2017$NEE_uStar_fall,
                           NA)
```

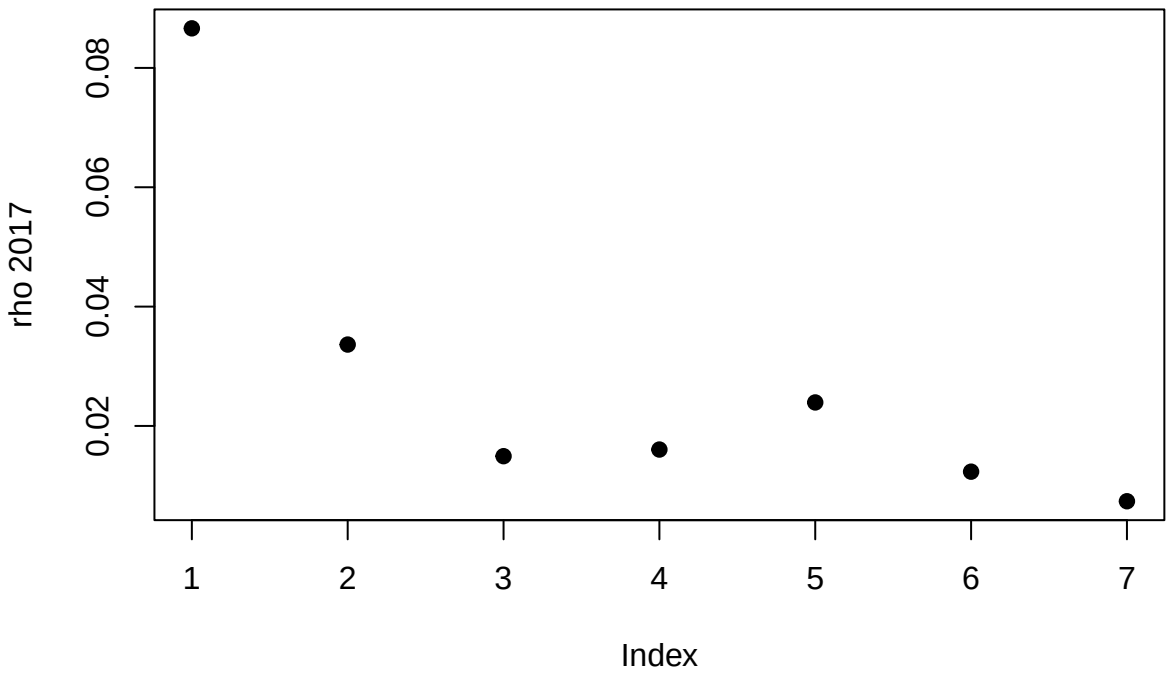
Step 6-1-2: Calculate the Empirical Autocorrelation Function

Calculate the effective autocorrelation components.

```
library(lognorm)
rho_all <- computeEffectiveAutoCorr(MkHCombResults$residual)
plot(rho_all[-1], ylab = 'rho all', pch = 19)
```




```
rho_2017 <- computeEffectiveAutoCorr(MkH2017$residual)
plot(rho_2017[-1], ylab = 'rho 2017', pch = 19)
```



Step 6-1-3: Calculate the Effective Number of Observations

We can calculate the number by using the autocorrelation function. Create the variable `nEff_all` and compare to the number of good observations `n_all`.

```
nEff_all <- computeEffectiveNumObs(MkHCombResults$residual, na.rm = TRUE, effAcf = rho_all)
c(nEff_all, n_all)
```

```
## [1] 53905.5 55268.0
```

Step 6-1-4: Calculate the Effective Number of Observation for 2004

For 2017, create the variable `nEff_2004` and compare to the number of good observations `n_2017`.

```
nEff_2017 <- computeEffectiveNumObs(MkH2017$residual, na.rm = TRUE, effAcf = rho_2017)
c(nEff_2017, n_2017)
```

```
## [1]  8551.439 10894.000
```

Step 6-1-5: Calculate the Mean Annual NEE and Standard Deviation for 2017

Using the non-gap-filled data (`NEE_Ustar_f`), the relative error can be calculated.

Do not use gap-filled records in the uncertainty estimation here.

The mean, standard deviations, and covariance.

```
NEE_notGapFilled <- mean(MkH2017$NEE_uStar_f)

sd_notGapFilled <- MkH2017$NEE_uStar_fsd[MkH2017$NEE_uStar_fqc == 0]

sdNEE_notGapFilled = sqrt(mean(sd_notGapFilled^2)) / sqrt(nEff_all - 1)

c(mean = NEE_notGapFilled, sd = sdNEE_notGapFilled,
  cv = sdNEE_notGapFilled/abs(NEE_notGapFilled))
```

```
##         mean           sd           cv
## -0.028427977  0.002933969  0.103207108
```

Step 6-1-6: Combined Uncertainties for the u_*-Thresholds and Random Uncertainties

Calculate the combined uncertainties of the:

1. NEE for different u_* scenarios.
2. NEE not gap-filled.

The combined uncertainties.

```
sdNEEUStar <- sd(NEE_UStar)
sdNEECombined <- sqrt(sdNEEUStar^2 + sdNEE_notGapFilled^2)
sdNEECombined
```

```
## [1] 0.009403695
```

References

Lasslop G, Reichstein M, Papale D, et al. (2010) Separation of net ecosystem exchange into assimilation and respiration using a light response curve approach: critical issues and global evaluation. Global Change Biology, Volume 16, Issue 1, Pages 187-208

Reichstein M, Falge E, Baldocchi D et al. (2005) On the separation of net ecosystem exchange into assimilation and ecosystem respiration: review and improved algorithm. Global Change Biology, 11, 1424-1439.