REddyProc for Eddy Covariance Data Analysis

Yusri Yusup

2023-07-31

Acknowledgements

This content was built on top of the talk by Thomas Wutzler. Credit goes to him for his:

1. Presentation (https://www.youtube.com/watch?v=-b0vc4u8kls&t=354s) and
2. Material at https://github.com/bgctw/EGU19EddyCourse/blob/master/Source/DEGebExample_complete.Rmd.

You can view more details about the package at https://bgc.iwww.mpg.de/5622399/REddyProc.

Prerequisites

Knowledge

It would be easier for you to follow the talk if you have brief backgrounds on:

1. R language
2. Eddy covariance
3. Net Ecosystem Exchange

Software

You would need the software below to perform the steps discussed in the talk.

1. R
2. RStudio
3. Packages: REddyProc

Introduction to the REddyProc Package: When Do We Use It?

The REddyProc package become useful after the pre-processing of the raw, high-frequency (10 Hz, 20 Hz, etc.) eddy covariance data. Using software, such as EddyPro, the data was checked using quality-control protocols, e.g., 0-1-2, 0-9 quality flag systems, etc., to discard low-quality fluxes.

We use this package to:

1. Post-process the Net Ecosystem Exchange (NEE) data.
2. Estimate NEE, e.g., the annual sum.
3. Determine the contributions of processes within NEE, e.g., production and respiration.

Steps in Using REddyProc

There are three main steps in a typical REddyProc analysis workflow.

1. The REddyProc package is utilized after the pre-processing step in which the fluxes are checked further for under-developed turbulence or low friction velocity (u_*).
    - The package can suggest threshold values of u_* and calculate uncertainties in different scenarios.
2. Fill gaps introduced due to the discarding of fluxes below the thresholds.
3. Separating fluxes into groups: Gross Primary Production (GPP) and Ecosystem Resipiration (R_{eco}).

Learning Outcomes

At the end of the talk, you would be able to:

1. Understand the capabilities of the REddyProc package.
2. Explain the analysis steps of the package.

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")
```

```
##
## The downloaded binary packages are in
##  /var/folders/rz/dkw8y98n1sv5r9gy22djhgbm0000gn/T//RtmphqrAb2/downloaded_packages
```

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

```
library(REddyProc)
```

Step 1: Prepare the Data

Step 1-1: Import the Data

The eddy covariance data that will be used in this walk-through are included in the package. They are for demo purposes.

1. `DEGebExample`, the Gebesee, Germany, data from 2004 to 2006.
2. `Example_DETha98`, the Tharandt, Germany, data for the year 1998.

The full data is downloadable at http://www.europe-fluxdata.eu/home/ after registration.

Load the Gebesee data.

```
data(DEGebExample)
```

Load the Tharandt data.

```
data(Example_DETha98)
```

Data Overview: Gebesee

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

The Gebesee, Germany, Data

Characteristics

- Surface: Agriculture
- Time zone: +1 GMT
- Latitude, Longitude: 51.1N, 10.9E

Notice that VPD is not in the dataset.

```
summary(DEGebExample)
```

```
##     DateTime                         NEE               Ustar        
##  Min.   :2004-01-01 00:30:00   Min.   :-49.919   Min.   :0.0000  
##  1st Qu.:2004-10-01 00:22:30   1st Qu.: -1.864   1st Qu.:0.0640  
##  Median :2005-07-02 00:15:00   Median :  0.635   Median :0.1490  
##  Mean   :2005-07-02 00:15:00   Mean   : -1.935   Mean   :0.1884  
##  3rd Qu.:2006-04-02 00:07:30   3rd Qu.:  1.834   3rd Qu.:0.2800  
##  Max.   :2007-01-01 00:00:00   Max.   : 19.008   Max.   :2.0450  
##                                NA's   :21849     NA's   :1149    
##       Tair               rH               Rg         
##  Min.   :-16.710   Min.   : 15.87   Min.   :   0.00  
##  1st Qu.:  3.360   1st Qu.: 66.61   1st Qu.:   0.00  
##  Median :  9.970   Median : 79.10   Median :   2.04  
##  Mean   :  9.664   Mean   : 75.24   Mean   : 124.71  
##  3rd Qu.: 15.520   3rd Qu.: 87.07   3rd Qu.: 176.03  
##  Max.   : 34.680   Max.   :100.00   Max.   :1046.03  
##                    NA's   :1
```

Data Overview: Tharandt

The Tharandt, Germany, Data

Characteristics

- Surface: Forest

- Time zone: +2 GMT
- Latitude, Longitude: 51.0N, 13.6E

Note that the timestamp is not in REddyProc-usable format.

```
names(Example_DETha98)
```

```
##  [1] "Year"  "DoY"   "Hour"  "NEE"   "LE"    "H"     "Rg"    "Tair"  "Tsoil"
## [10] "rH"    "VPD"   "Ustar"
```

```
head(Example_DETha98)
```

```
##   Year DoY Hour   NEE   LE      H Rg Tair Tsoil    rH VPD Ustar
## 1 1998   1  0.5 -1.21 1.49 -11.77  0  7.4  4.19 55.27 4.6  0.72
## 2 1998   1  1.0  1.72 3.80 -13.50  0  7.5  4.20 55.95 4.6  0.52
## 3 1998   1  1.5    NA 1.52 -18.30  0  7.1  4.22 57.75 4.3  0.22
## 4 1998   1  2.0    NA 3.94 -17.47  0  6.6  4.23 60.20 3.9  0.20
## 5 1998   1  2.5  2.55 8.30 -21.42  0  6.6  4.22 59.94 3.9  0.33
## 6 1998   1  3.0    NA 1.33 -20.55  0  6.5  4.21 59.25 4.0  0.15
```

Important Parameters

Parameters required for REddyProc 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`

Note that:

- `Rg` is global solar radiation.
- `Tair` is air temperature.
- `rH` is relative humidity.
- `VPD` is vapor pressure deficit.

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`. We will use this in the demo.
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 1-3-1: Tharandt Dataset: Addressing the Unsupported Timestamp Format

In the Tharandt dataset, the date-time columns are not suitable for REddyProc. It needs to be converted to the `POSIX` format.

```
head(Example_DETha98)
```

```
##   Year DoY Hour   NEE   LE      H Rg Tair Tsoil    rH VPD Ustar
## 1 1998   1  0.5 -1.21 1.49 -11.77  0  7.4  4.19 55.27 4.6  0.72
## 2 1998   1  1.0  1.72 3.80 -13.50  0  7.5  4.20 55.95 4.6  0.52
## 3 1998   1  1.5    NA 1.52 -18.30  0  7.1  4.22 57.75 4.3  0.22
## 4 1998   1  2.0    NA 3.94 -17.47  0  6.6  4.23 60.20 3.9  0.20
## 5 1998   1  2.5  2.55 8.30 -21.42  0  6.6  4.22 59.94 3.9  0.33
## 6 1998   1  3.0    NA 1.33 -20.55  0  6.5  4.21 59.25 4.0  0.15
```

Convert the Timestamp to `POSIX`

We can convert the timestamp using the `fConvertTimeToPosix` function. It will add the `DateTime` column into the data frame.

The `'YDH'` means Year-Day-Hour, and the `Year`, `Day`, and `Hour` arguments require the columns that contains the Year, DoY, and Hour information.

You can find other valid time configurations in the documentation by running the command `?fConvertTimeToPosix`.

```
Example_DETha98V1 <- fConvertTimeToPosix(Example_DETha98, TFormat = c('YDH'),
                     Year = 'Year',
                     Day = 'DoY',
                     Hour = 'Hour')
```

```
## Converted time format 'YDH' to POSIX with column name 'DateTime'.
```

```
head(Example_DETha98V1)
```

```
##              DateTime Year DoY Hour   NEE   LE      H Rg Tair Tsoil    rH VPD
## 1 1998-01-01 00:30:00 1998   1  0.5 -1.21 1.49 -11.77  0  7.4  4.19 55.27 4.6
## 2 1998-01-01 01:00:00 1998   1  1.0  1.72 3.80 -13.50  0  7.5  4.20 55.95 4.6
## 3 1998-01-01 01:30:00 1998   1  1.5    NA 1.52 -18.30  0  7.1  4.22 57.75 4.3
## 4 1998-01-01 02:00:00 1998   1  2.0    NA 3.94 -17.47  0  6.6  4.23 60.20 3.9
## 5 1998-01-01 02:30:00 1998   1  2.5  2.55 8.30 -21.42  0  6.6  4.22 59.94 3.9
## 6 1998-01-01 03:00:00 1998   1  3.0    NA 1.33 -20.55  0  6.5  4.21 59.25 4.0
##   Ustar
## 1  0.72
## 2  0.52
## 3  0.22
## 4  0.20
## 5  0.33
## 6  0.15
```

Step 1-3-2: Missing VPD in the Gebesee Data

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

```
head(DEGebExample)
```

```
##                  DateTime NEE Ustar  Tair    rH Rg
## 35041 2004-01-01 00:30:00  NA 0.092 -0.06 96.13  0
## 35042 2004-01-01 01:00:00  NA 0.090 -0.14 96.10  0
## 35043 2004-01-01 01:30:00  NA 0.023 -0.16 95.93  0
## 35044 2004-01-01 02:00:00  NA 0.038 -0.17 95.80  0
## 35045 2004-01-01 02:30:00  NA 0.077 -0.19 95.67  0
## 35046 2004-01-01 03:00:00  NA 0.025 -0.23 95.47  0
```

Calculate VPD

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

```
VPD <- fCalcVPDfromRHandTair(DEGebExample$rH,   # The unit is %
                             DEGebExample$Tair) # The unit is degree Celsius
DEGebExampleV1 <- cbind(DEGebExample,VPD)
rm(VPD) # A house-keeping step.
head(DEGebExampleV1)
```

```
##                  DateTime NEE Ustar  Tair    rH Rg       VPD
## 35041 2004-01-01 00:30:00  NA 0.092 -0.06 96.13  0 0.2353394
## 35042 2004-01-01 01:00:00  NA 0.090 -0.14 96.10  0 0.2357827
## 35043 2004-01-01 01:30:00  NA 0.023 -0.16 95.93  0 0.2457012
## 35044 2004-01-01 02:00:00  NA 0.038 -0.17 95.80  0 0.2533640
## 35045 2004-01-01 02:30:00  NA 0.077 -0.19 95.67  0 0.2608249
## 35046 2004-01-01 03:00:00  NA 0.025 -0.23 95.47  0 0.2720758
```

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 Gebesee data. The ID is `DE-Geb` and the parameters are:

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

```
EProcDEGeb <- sEddyProc$new('DE-Geb', DEGebExampleV1,
                            c('NEE','Rg','Tair','VPD','Ustar'))
```

```
## New sEddyProc class for site 'DE-Geb'
```

Check the Object

Check the additional info of the data.

```
EProcDEGeb$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.

```
EProcDEGeb$sSetLocationInfo(LatDeg = 51.1, LongDeg = 10.9, TimeZoneHour = 1)
EProcDEGeb$sLOCATION
```

```
## $LatDeg
## [1] 51.1
##
## $LongDeg
## [1] 10.9
##
## $TimeZoneHour
## [1] 1
```

Step 2: Create the Tharandt REddyProc Object Class

Create the class for the Tharandt data. The ID is `DE-Tha` and the parameters are:

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

```
EProcDETha <- sEddyProc$new('DE-Tha', Example_DETha98V1, c('NEE','Rg','Tair','VPD','Ustar'))
```

```
## New sEddyProc class for site 'DE-Tha'
```

Check the Object

Check the additional info of the data.

```
EProcDETha$sLOCATION
```

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

Add the location information.

```
EProcDETha$sSetLocationInfo(LatDeg = 51.0, LongDeg = 13.6, TimeZoneHour = 2)
EProcDETha$sLOCATION
```

```
## $LatDeg
## [1] 51
##
## $LongDeg
## [1] 13.6
##
## $TimeZoneHour
## [1] 2
```

Step 3: u_*-Threshold Estimation

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

A previous study determined the days on which the seasons and u_* shifted. It can be determined by the visual inspection of the data.

Year	Start Day
2004	70, 210, 320
2005	70, 180, 320
2006	120, 350

Step 3-1-2: Adding the Start Days for the Gebesee Data.

Create a data frame for the start days.

```
df_startDays <- data.frame(day=c(70,210,320,70,180,320,120,305),
                           year=c(2004,2004,2004,2005,2005,2005,2006,2006))
df_startDays
```

```
##   day year
## 1  70 2004
```

```
## 2 210 2004
## 3 320 2004
## 4  70 2005
## 5 180 2005
## 6 320 2005
## 7 120 2006
## 8 305 2006
```

Creating the Seasonal Factors for Row-Tagging

We can use `usCreateSeasonFactorYdayYear` to change the `df_startDays` data frame to a factor vector that contains values that tag each rows to their respective seasons.

Create the factor vector.

Note that the product `15*60` is used to make the time be between 00:00 and 00:30.

The `summary` shows that there are 3312 observations for season `2004001`, i.e., between days 1 and 70, etc.

```
seasonFactor <- usCreateSeasonFactorYdayYear(DEGebExampleV1$DateTime - 15*60,
                                             starts = df_startDays)
summary(seasonFactor)
```

```
## 2004001 2004070 2004210 2004320 2005070 2005180 2005320 2006120 2006305
##    3312    6720    5280    5568    5280    6720    7920    8880    2928
```

```
head(seasonFactor)
```

```
## [1] 2004001 2004001 2004001 2004001 2004001 2004001
## 9 Levels: 2004001 2004070 2004210 2004320 2005070 2005180 2005320 ... 2006305
```

Optional: Viewing the Gebesee Data with Season Demarcation-Lines

Because the start days data frame is a collection of integers, we need to change it to the POSIX format.

Create timestamps in the POSIX format from `df_startDays`. Here, we embed a new data frame with the additional column `Hour` into the `fConvertTimeToPosix` function call. The hour is set at `0.25` to be between 00:00 and 00:30, i.e., 00:15.

```
seasonStartsDate <- fConvertTimeToPosix(data.frame(Year = df_startDays$year,
                                                   DoY = df_startDays$day,
                                                   Hour = 0.25),
                                        TFormat = 'YDH',
                                        Year = "Year",
                                        Day = "DoY",
                                        Hour = "Hour")
```

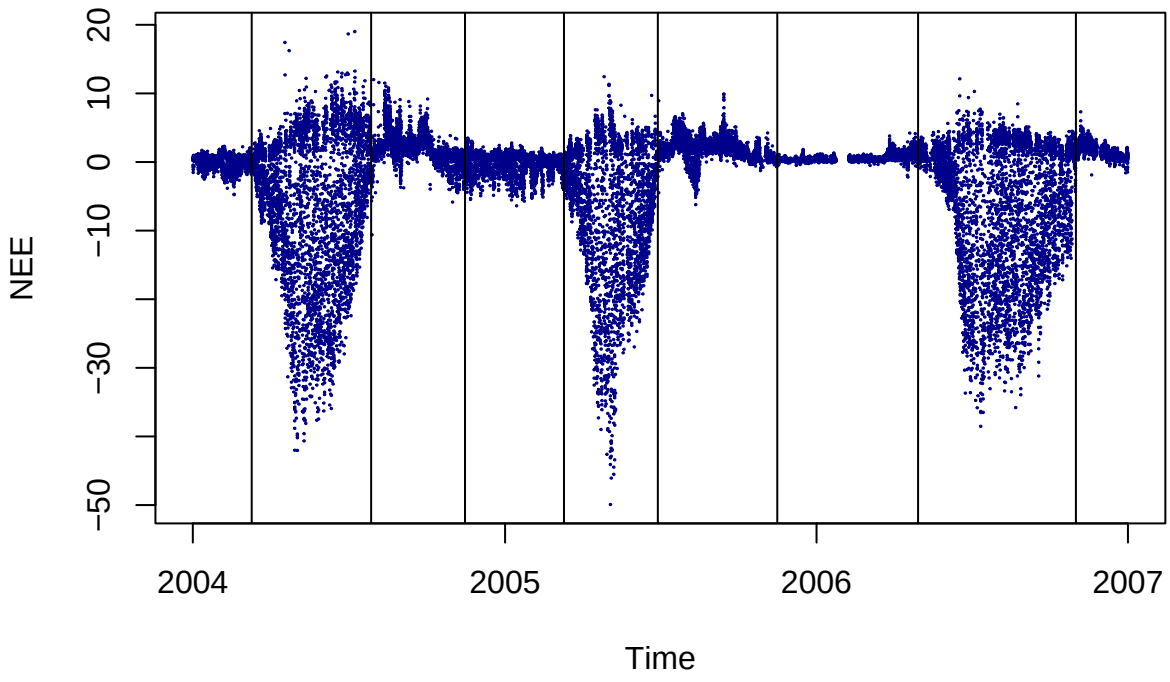
```
## Converted time format 'YDH' to POSIX with column name 'DateTime'.
```

```
seasonStartsDate
```

```
##              DateTime Year DoY Hour
## 1 2004-03-10 00:15:00 2004  70 0.25
## 2 2004-07-28 00:15:00 2004 210 0.25
## 3 2004-11-15 00:15:00 2004 320 0.25
## 4 2005-03-11 00:15:00 2005  70 0.25
## 5 2005-06-29 00:15:00 2005 180 0.25
## 6 2005-11-16 00:15:00 2005 320 0.25
## 7 2006-04-30 00:15:00 2006 120 0.25
## 8 2006-11-01 00:15:00 2006 305 0.25
```

You can check the dates by plotting them on the time series.

```
plot(DEGebExample$DateTime, DEGebExample$NEE, pch=19, xlab = "Time", ylab = 'NEE', cex = 0.1, col = "da
abline(v = seasonStartsDate$DateTime)
```



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, 30 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.

```
EProcDEGeb$sEstimateUstarScenarios(seasonFactor = seasonFactor,
                                   nSample = 30,
                                   probs = c(0.05,0.50,0.95))
```

```
##

## Estimated UStar distribution of:
##        uStar        5%       50%       95%
## 1 0.1604118 0.1173781 0.1608725 0.2147833
## by using  30 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`.

```
EProcDEGeb$sPlotNEEVersusUStarForSeason(dir = "../figs")
```

```
## Saved plot to: ../figs/DE-Geb_04-06_NEEvsUStar_2004001_none.pdf
```

```
EProcDEGeb$sGetEstimatedUstarThresholdDistribution()
```

```
##    aggregationMode seasonYear  season      uStar         5%        50%
## 1           single         NA    <NA> 0.16041176 0.11737812 0.16087255
## 2             year       2004    <NA> 0.13500000 0.11646250 0.16011111
## 3             year       2005    <NA> 0.16041176 0.10558945 0.15465278
## 4             year       2006    <NA> 0.25094444 0.05938125 0.19860000
```

```
## 5           season       2004 2004001 0.13500000 0.10701000 0.13624167
## 6           season       2004 2004070 0.12037500 0.08235000 0.10892778
## 7           season       2004 2004210 0.08925000 0.08223750 0.14619375
## 8           season       2005 2004320 0.16041176 0.08249035 0.13498754
## 9           season       2005 2005070 0.12533333 0.11524375 0.14516667
## 10          season       2005 2005180 0.13473214 0.09510714 0.12714286
## 11          season       2006 2005320 0.04842361 0.04952299 0.06817739
## 12          season       2006 2006120 0.06966667 0.04986558 0.06983654
## 13          season       2006 2006305 0.25094444 0.10356563 0.20894444
##           95%
## 1  0.21478333
## 2  0.22673333
## 3  0.18581875
## 4  0.26875000
## 5  0.17479111
## 6  0.12190500
## 7  0.22673333
## 8  0.17176967
## 9  0.18125833
## 10 0.16010000
## 11 0.08293125
## 12 0.09890801
## 13 0.26875000
```

Step 4-1: Gap-Filling the Gebesee 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

```
EProcDEGeb$sGetUstarScenarios()
```

```
##    season     uStar        U05       U50       U95
## 1 2004001 0.1350000 0.11646250 0.1601111 0.2267333
## 2 2004070 0.1350000 0.11646250 0.1601111 0.2267333
## 3 2004210 0.1350000 0.11646250 0.1601111 0.2267333
## 4 2004320 0.1604118 0.10558945 0.1546528 0.1858187
## 5 2005070 0.1604118 0.10558945 0.1546528 0.1858187
## 6 2005180 0.1604118 0.10558945 0.1546528 0.1858187
## 7 2005320 0.2509444 0.05938125 0.1986000 0.2687500
## 8 2006120 0.2509444 0.05938125 0.1986000 0.2687500
## 9 2006305 0.2509444 0.05938125 0.1986000 0.2687500
```

Instruct REddyProc to use the seasonal thresholds.

```
EProcDEGeb$useSeaonsalUStarThresholds()
```

Confirm that the seasonal thresholds are used by displaying it.

```
EProcDEGeb$sGetUstarScenarios()
```

```
##     season      uStar        U05        U50        U95
## 5  2004001 0.13500000 0.10701000 0.13624167 0.17479111
## 6  2004070 0.12037500 0.08235000 0.10892778 0.12190500
## 7  2004210 0.08925000 0.08223750 0.14619375 0.22673333
## 8  2004320 0.16041176 0.08249035 0.13498754 0.17176967
## 9  2005070 0.12533333 0.11524375 0.14516667 0.18125833
## 10 2005180 0.13473214 0.09510714 0.12714286 0.16010000
## 11 2005320 0.04842361 0.04952299 0.06817739 0.08293125
## 12 2006120 0.06966667 0.04986558 0.06983654 0.09890801
## 13 2006305 0.25094444 0.10356563 0.20894444 0.26875000
```

Step 4-1-2: Gap-Fill the Gebesee 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 for non-gap records by replacing the original values with gap-filled values for uncertainties calculations.

```
EProcDEGeb$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.

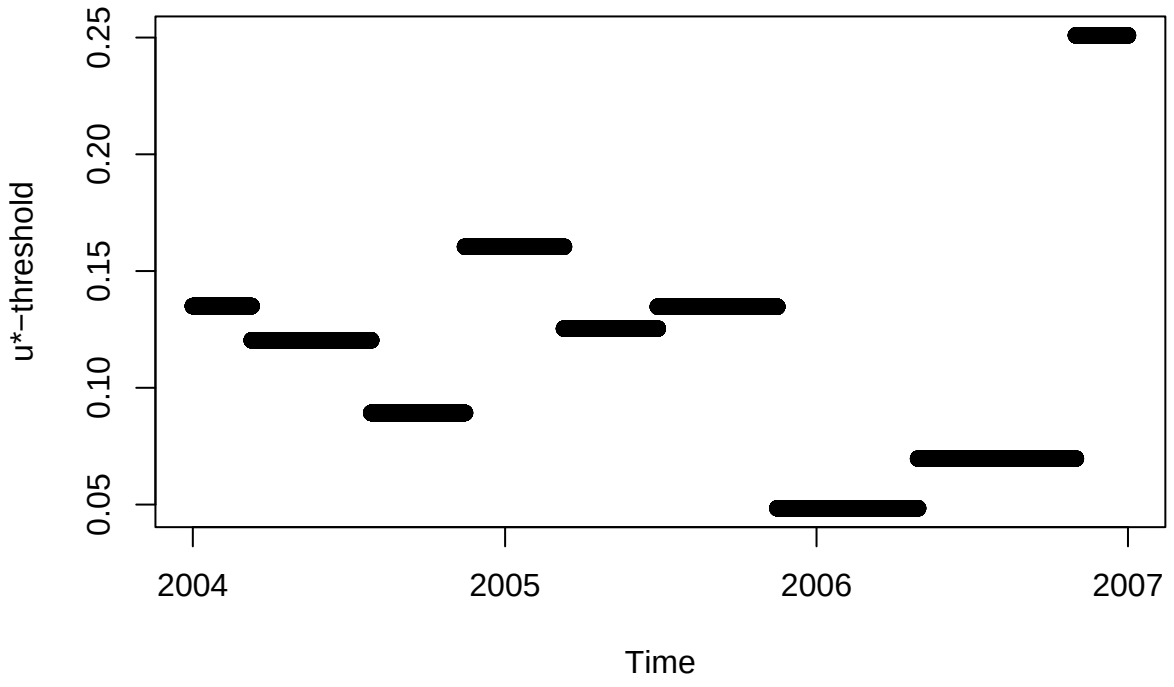
```
colnames(EProcDEGeb$sExportResults())
```

```
##  [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"
```

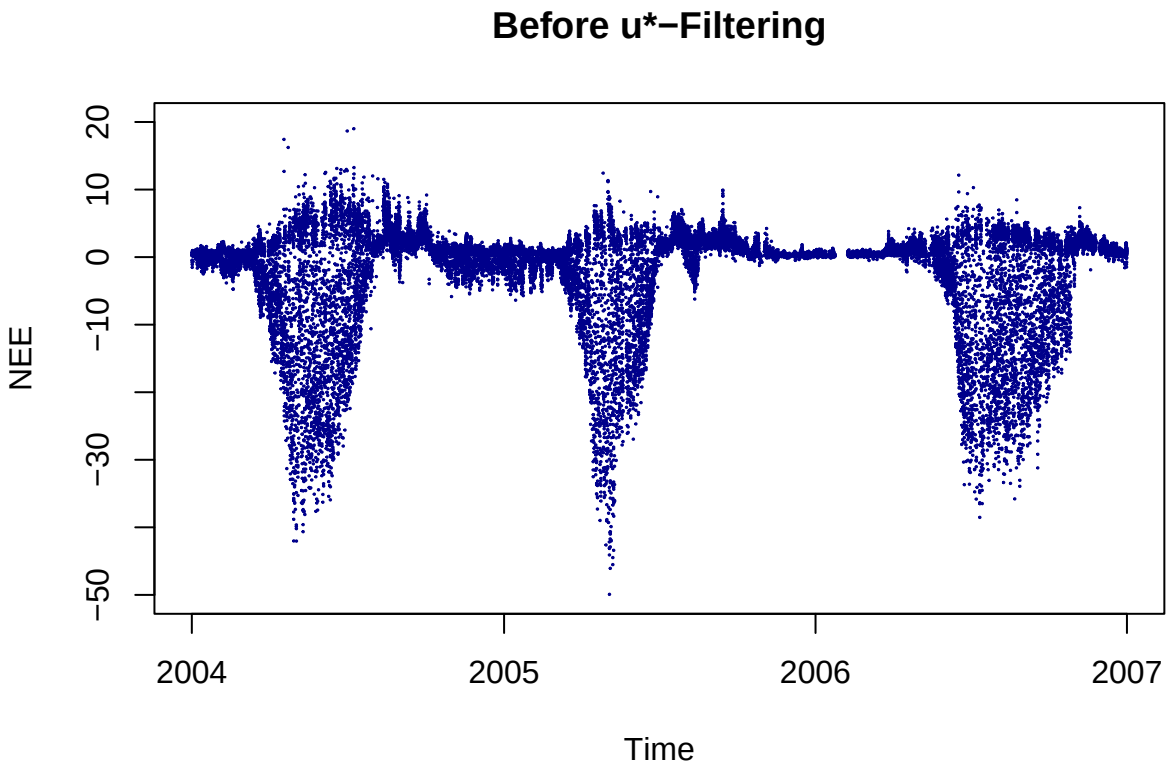
View Some Columns

Plotting a column of the REddyProc object.

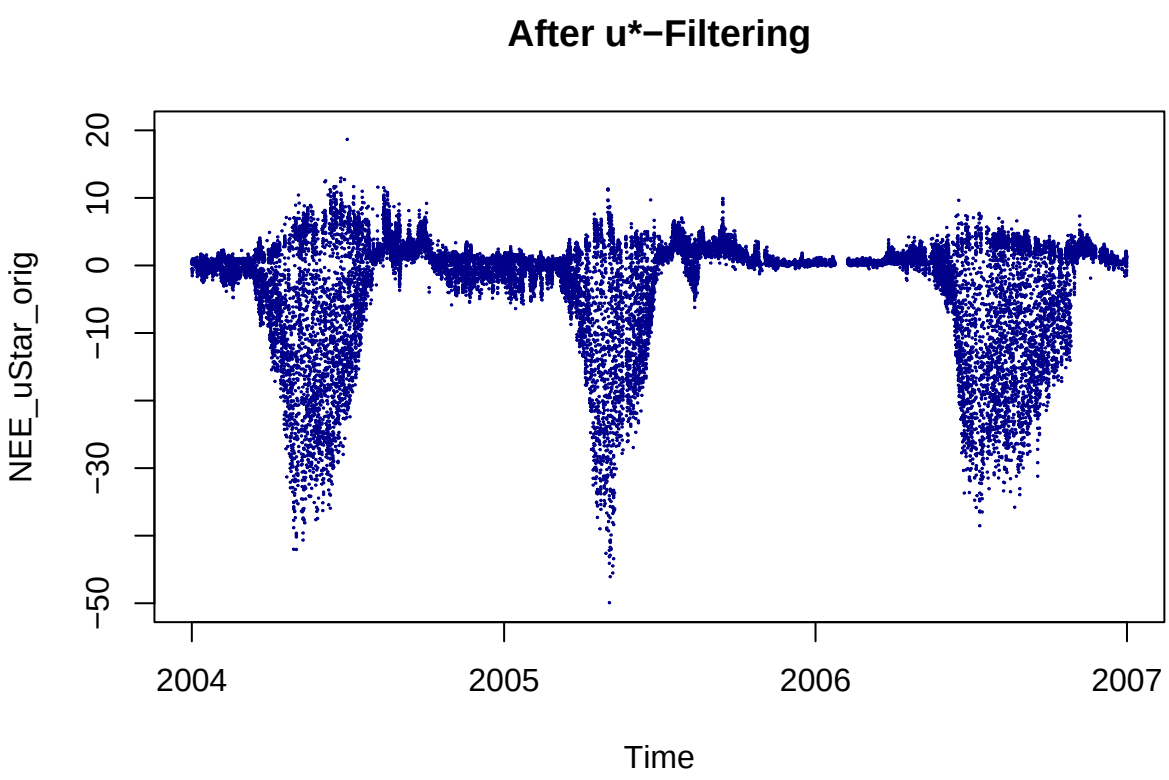
```
plot(EProcDEGeb$sDATA$sDateTime, EProcDEGeb$sExportResults()$Ustar_uStar_Thres, pch = 19,
     xlab = 'Time', ylab = 'u*-threshold')
```



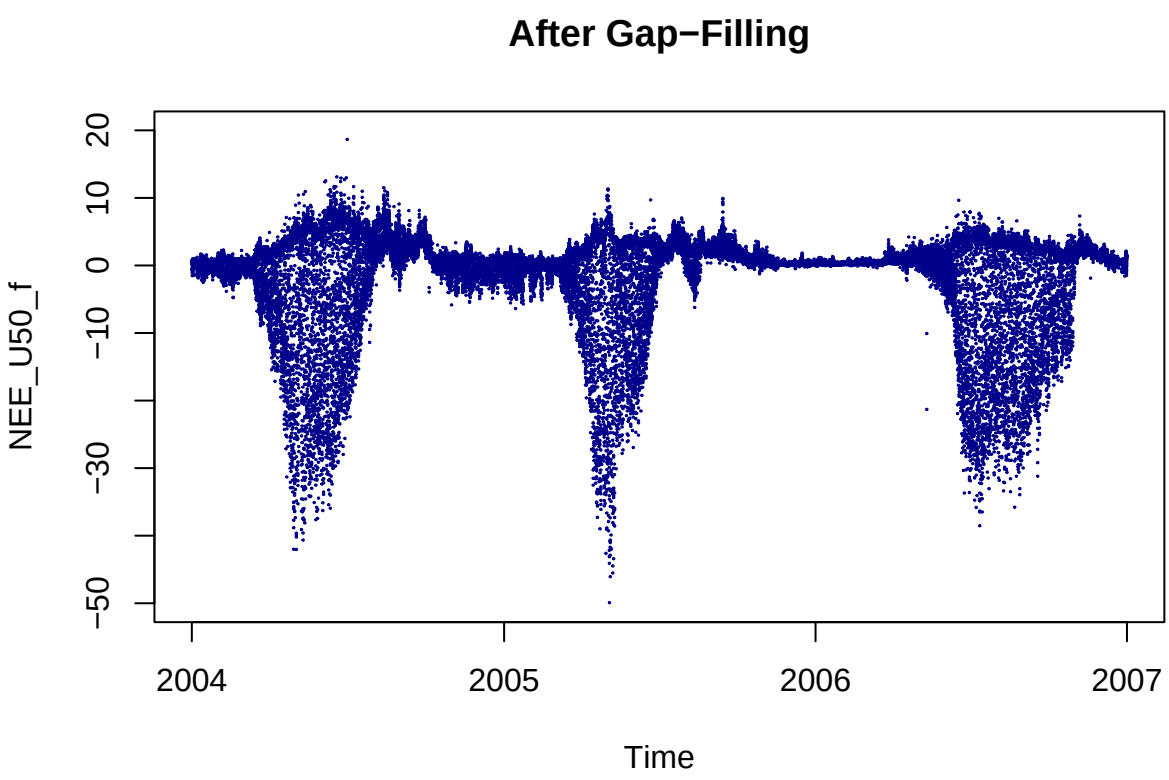
```
plot(DEGebExampleV1$DateTime,DEGebExampleV1$NEE, pch = 19, cex = 0.1, col = "darkblue",
     xlab = 'Time', ylab = 'NEE', ylim=c(-50,20), main = "Before u*-Filtering")
```



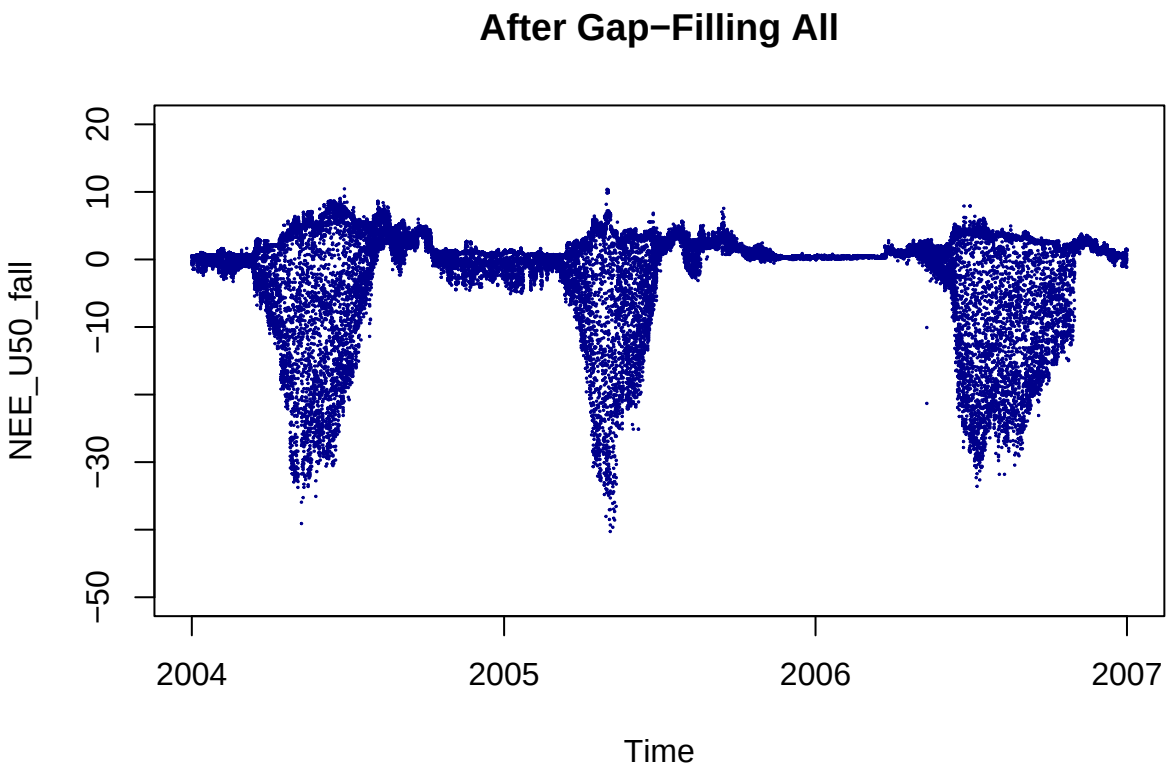

```
plot(EProcDEGeb$sDATA$sDateTime, EProcDEGeb$sExportResults()$NEE_uStar_orig, pch = 19,
     cex = 0.1, col = "darkblue",
     xlab = 'Time', ylab = 'NEE_uStar_orig', ylim=c(-50,20),
     main = "After u*-Filtering")
```




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



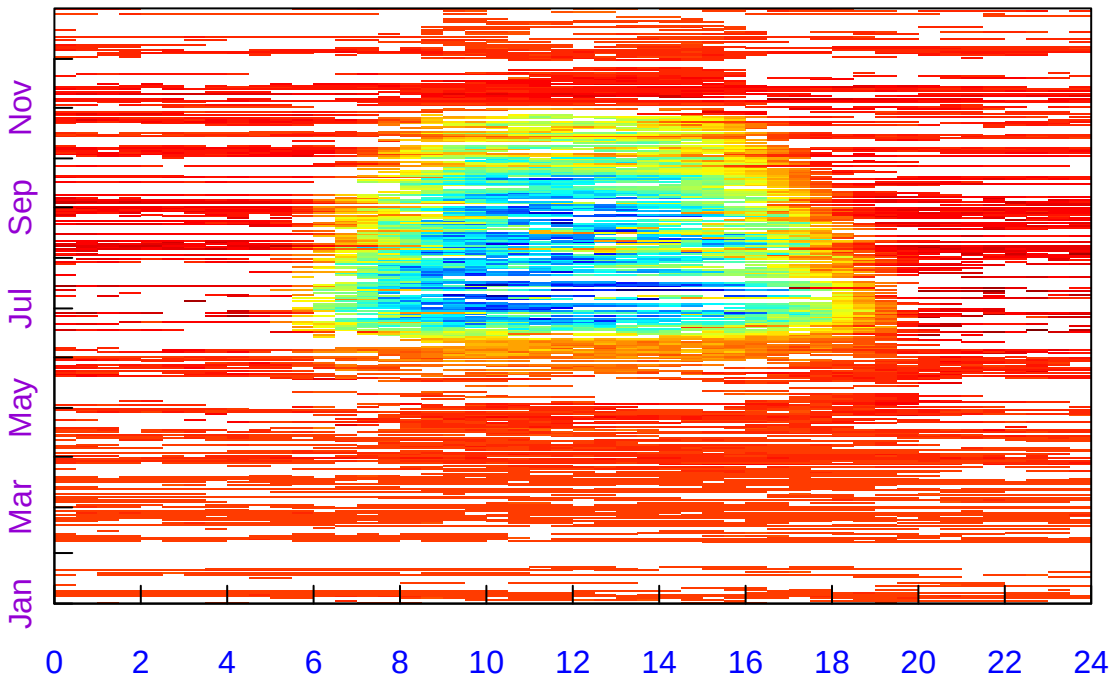

```
plot(EProcDEGeb$sDATA$sDateTime, EProcDEGeb$sExportResults()$NEE_U50_fall, pch = 19,
     cex = 0.1, col = "darkblue",
     xlab = 'Time', ylab = 'NEE_U50_fall', ylim=c(-50,20),
     main = "After Gap-Filling All")
```



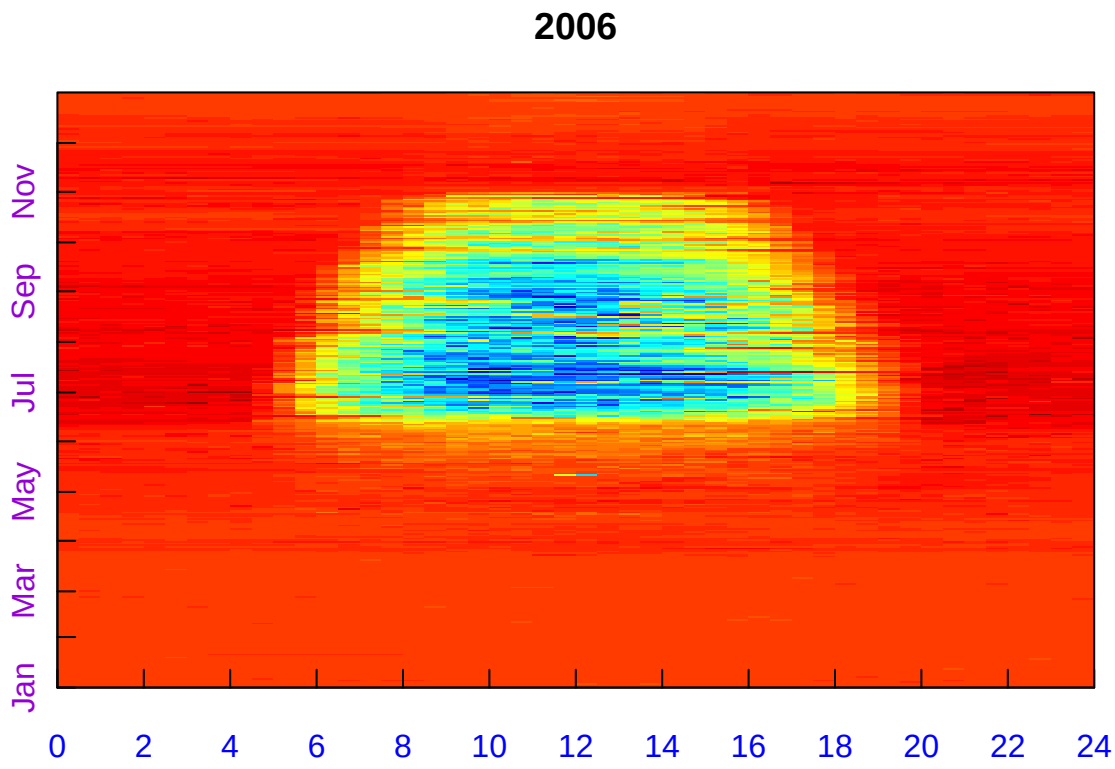
Step 4-1-3: Gebesee Fingerprint Plot

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

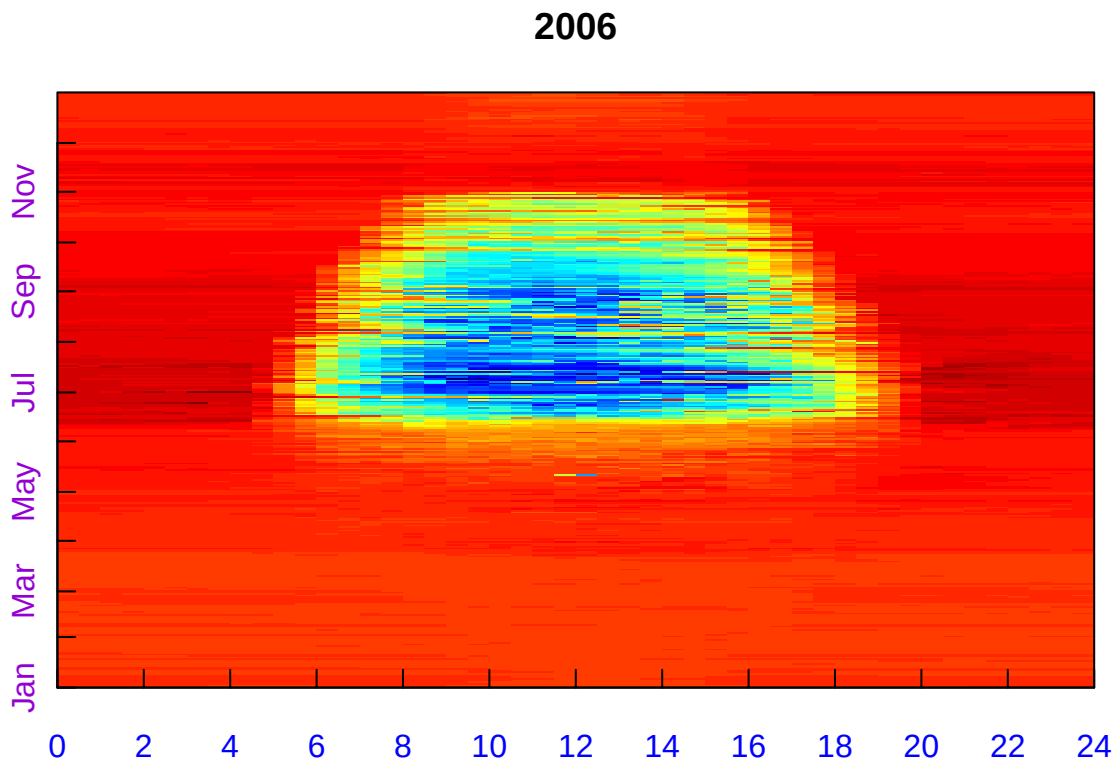
```
EProcDEGeb$sPlotFingerprintY('NEE_uStar_orig', Year = 2006)
```




```
EProcDEGeb$sPlotFingerprintY('NEE_U50_f', Year = 2006)
```




```
EProcDEGeb$sPlotFingerprintY('NEE_U50_fall', Year = 2006)
```



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

```
EProcDEGeb$sPlotFingerprint('NEE_U50_f', Dir = "../figs")
```

```
## Saved plot to: ../figs/DE-Geb_04-06_FP_NEE_U50_f.pdf
```

Step 5-1: Gebesee 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`.

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

Step 5-1-1: Gebesee 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).

```
EProcDEGeb$sMRFluxPartitionUStarScens()
```

Step 5-1-2: Plotting the GPP

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

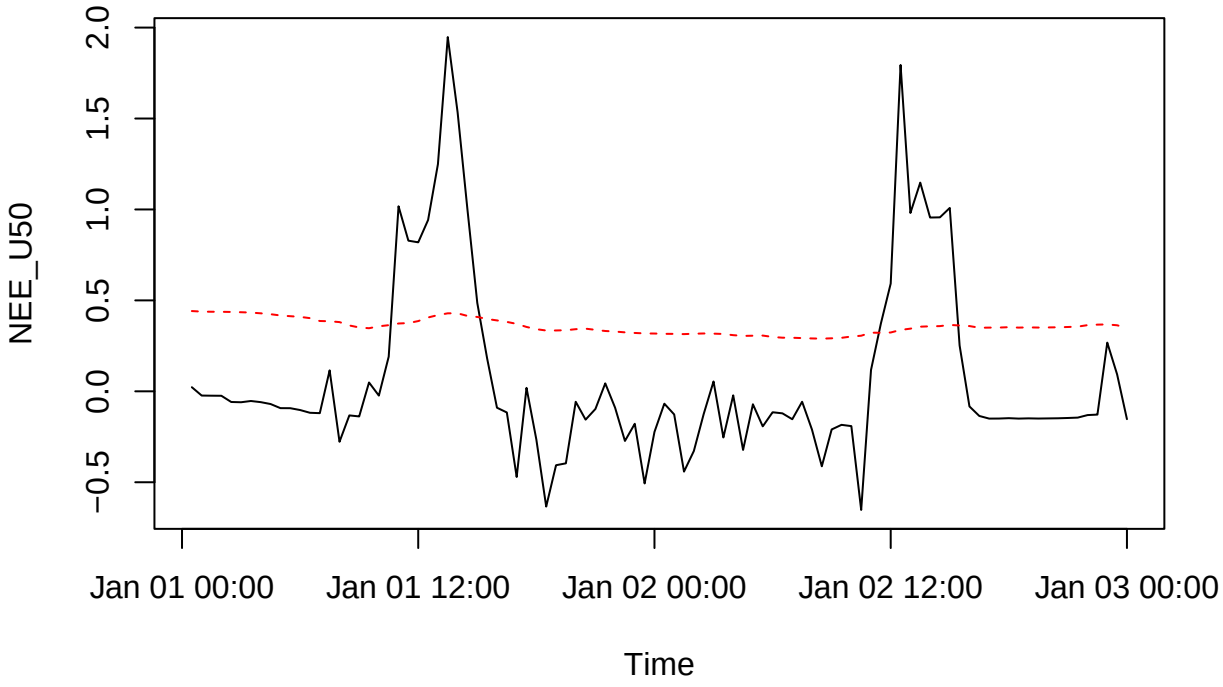
```
names(EProcDEGeb$sExportResults())
```

```
##   [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_U05"        "FP_NEEnight_U05"   "FP_Temp_U05"
##  [76] "E_0_U05"           "R_ref_U05"         "Reco_U05"
##  [79] "GPP_U05_f"         "GPP_U05_fqc"       "PotRad_U50"
##  [82] "FP_NEEnight_U50"   "FP_Temp_U50"       "E_0_U50"
##  [85] "R_ref_U50"         "Reco_U50"          "GPP_U50_f"
##  [88] "GPP_U50_fqc"       "PotRad_U95"        "FP_NEEnight_U95"
##  [91] "FP_Temp_U95"       "E_0_U95"           "R_ref_U95"
##  [94] "Reco_U95"          "GPP_U95_f"         "GPP_U95_fqc"
##  [97] "PotRad_uStar"      "FP_NEEnight_uStar" "FP_Temp_uStar"
## [100] "E_0_uStar"         "R_ref_uStar"       "Reco_uStar"
## [103] "GPP_uStar_f"       "GPP_uStar_fqc"
```

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(DEGebExampleV1$DateTime, nRec),
     head(EProcDEGeb$sExportResults()$GPP_U50_f, nRec),
     type = "l", xlab = 'Time', ylab = 'NEE_U50')
lines(head(DEGebExampleV1$DateTime, nRec),
      head(EProcDEGeb$sExportResults()$Reco_U50,nRec),
      type = "l", lty = 2,
      col = 'red')
```



Step 5-1-3: Gebesee 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).

```
EProcDEGeb$sGLFluxPartitionUStarScens()
```

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

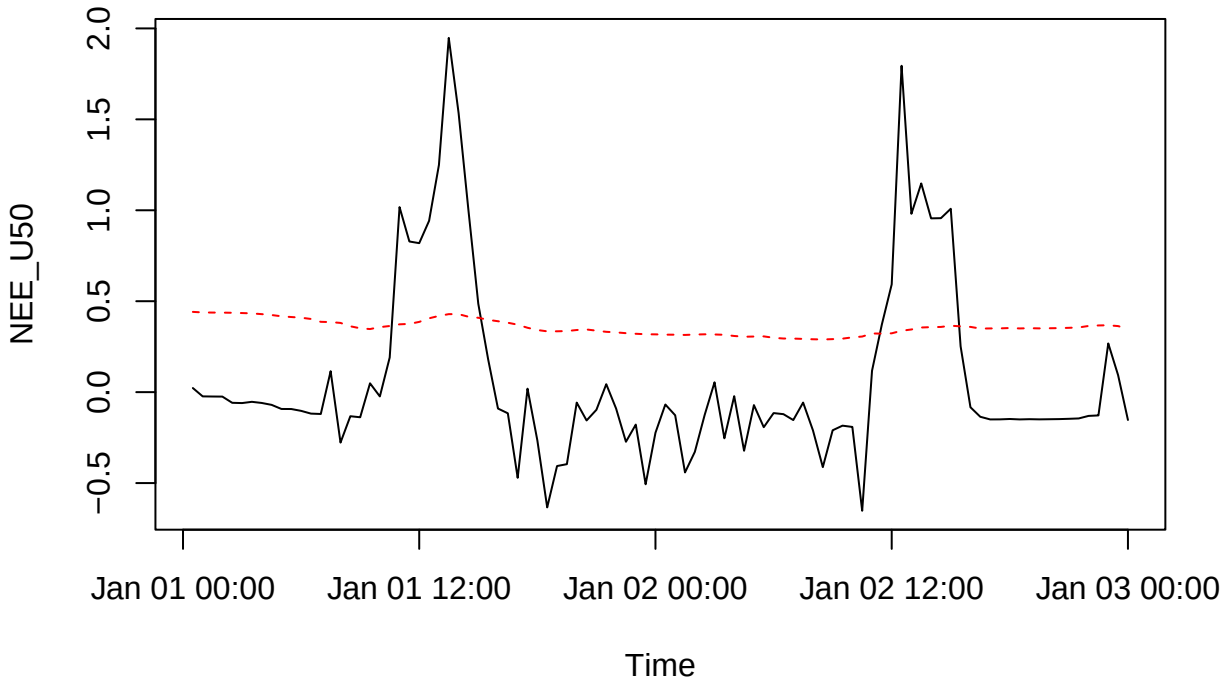
```
names(EProcDEGeb$sExportResults())
```

```
##   [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_U05"        "FP_NEEnight_U05"   "FP_Temp_U05"
##  [76] "E_0_U05"           "R_ref_U05"         "Reco_U05"
##  [79] "GPP_U05_f"         "GPP_U05_fqc"       "PotRad_U50"
##  [82] "FP_NEEnight_U50"   "FP_Temp_U50"       "E_0_U50"
##  [85] "R_ref_U50"         "Reco_U50"          "GPP_U50_f"
##  [88] "GPP_U50_fqc"       "PotRad_U95"        "FP_NEEnight_U95"
##  [91] "FP_Temp_U95"       "E_0_U95"           "R_ref_U95"
##  [94] "Reco_U95"          "GPP_U95_f"         "GPP_U95_fqc"
##  [97] "PotRad_uStar"      "FP_NEEnight_uStar" "FP_Temp_uStar"
## [100] "E_0_uStar"         "R_ref_uStar"       "Reco_uStar"
## [103] "GPP_uStar_f"       "GPP_uStar_fqc"     "PotRad_NEW"
## [106] "Reco_DT_U05"       "GPP_DT_U05"        "Reco_DT_U05_SD"
## [109] "GPP_DT_U05_SD"     "Reco_DT_U50"       "GPP_DT_U50"
## [112] "Reco_DT_U50_SD"    "GPP_DT_U50_SD"     "Reco_DT_U95"
## [115] "GPP_DT_U95"        "Reco_DT_U95_SD"    "GPP_DT_U95_SD"
## [118] "FP_VARnight"       "FP_VARday"         "NEW_FP_Temp"
## [121] "NEW_FP_VPD"        "FP_RRef_Night"     "FP_qc"
## [124] "FP_dRecPar"        "FP_errorcode"      "FP_GPP2000"
## [127] "FP_k"              "FP_beta"           "FP_alpha"
## [130] "FP_RRef"           "FP_E0"             "FP_k_sd"
## [133] "FP_beta_sd"        "FP_alpha_sd"       "FP_RRef_sd"
## [136] "FP_E0_sd"          "Reco_DT_uStar"     "GPP_DT_uStar"
## [139] "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*2
plot(head(DEGebExampleV1$DateTime, nRec),
     head(EProcDEGeb$sExportResults()$GPP_U50_f,nRec),
     type = "l", xlab = 'Time', ylab = 'NEE_U50')
```

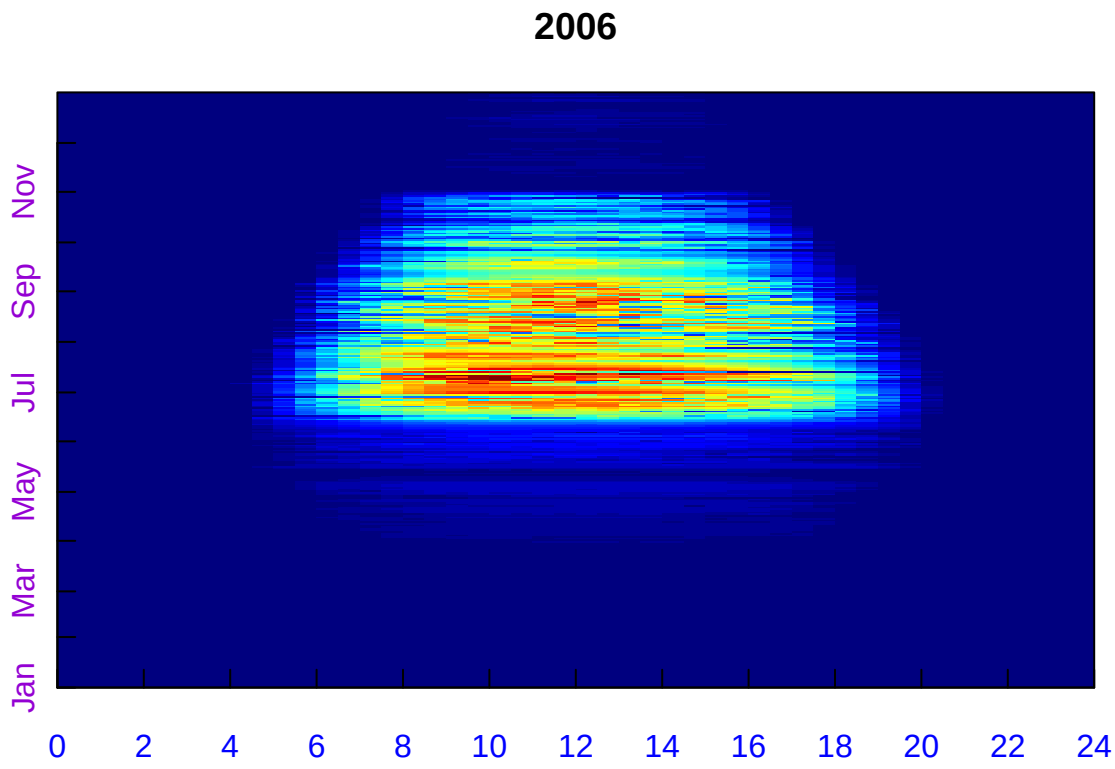
```
lines(head(DEGebExampleV1$DateTime, nRec),
      head(EProcDEGeb$sExportResults()$Reco_U50,nRec),
      type = "l", lty = 2, col = 'red')
```



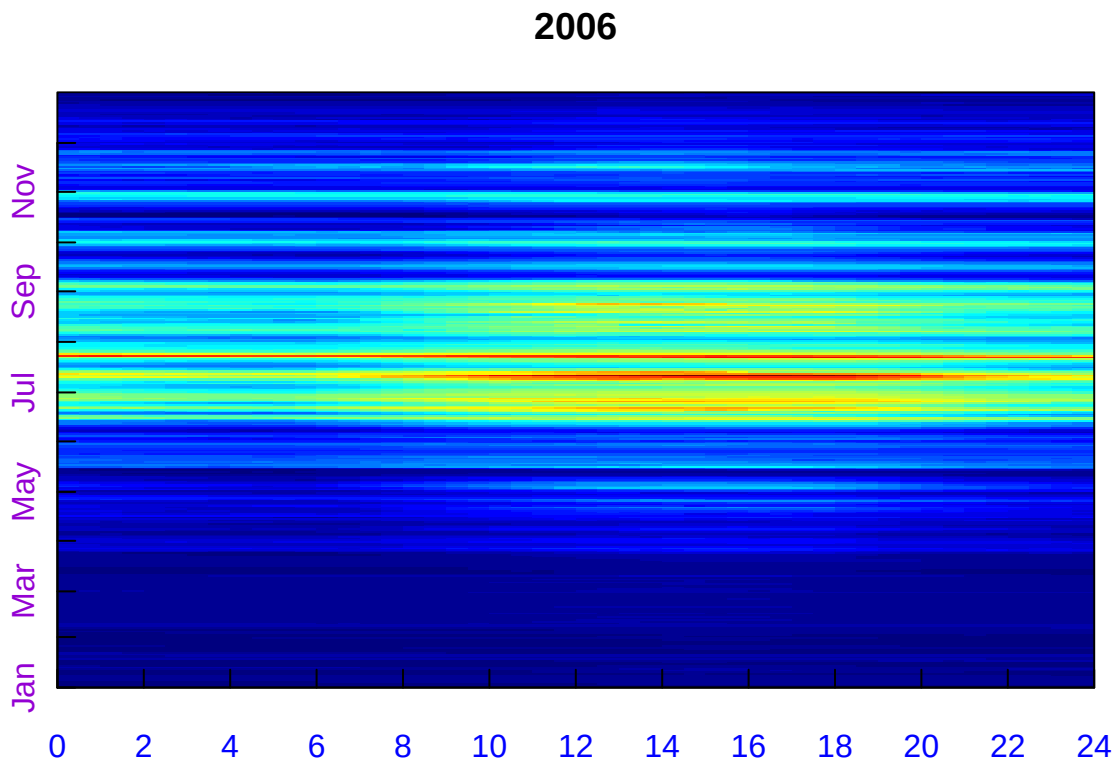
Step 5-1-4: Fingerprint Plots of GPP_DT and Reco_DT

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

```
EProcDEGeb$sPlotFingerprintY('GPP_DT_U50', Year = 2006)
```




```
EProcDEGeb$sPlotFingerprintY('Reco_DT_U50', Year = 2006)
```



Step 5-1-5: Export the Gebesee Results

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

```
GebData <- EProcDEGeb$sExportData() # Write the original data to GebData.
GebResults <- EProcDEGeb$sExportResults() # Write the results of REddyProc to GebResults.
GebCombResults <- cbind(GebData, GebResults)
fWriteDataframeToFile(GebCombResults, "DE-Geb_Part.txt", Dir = "../results")
```

```
## Number of NA convertered to '-9999': 1841420
```

```
## Wrote tab separated textfile: ../results/DE-Geb_Part.txt
```

Step 6-1: Gebesee: Bias with u_{*Th}

Calculating the Bias for the Year 2004

We will be calculating the bias of NEE due to the u_*-threshold for 2004.

Check the names of the columns of `GebCombResults`.

```
names(GebCombResults)
```

```
##   [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_U05"        "FP_NEEnight_U05"   "FP_Temp_U05"
##  [82] "E_0_U05"           "R_ref_U05"         "Reco_U05"
##  [85] "GPP_U05_f"         "GPP_U05_fqc"       "PotRad_U50"
##  [88] "FP_NEEnight_U50"   "FP_Temp_U50"       "E_0_U50"
##  [91] "R_ref_U50"         "Reco_U50"          "GPP_U50_f"
##  [94] "GPP_U50_fqc"       "PotRad_U95"        "FP_NEEnight_U95"
##  [97] "FP_Temp_U95"       "E_0_U95"           "R_ref_U95"
## [100] "Reco_U95"          "GPP_U95_f"         "GPP_U95_fqc"
## [103] "PotRad_uStar"      "FP_NEEnight_uStar" "FP_Temp_uStar"
## [106] "E_0_uStar"         "R_ref_uStar"       "Reco_uStar"
## [109] "GPP_uStar_f"       "GPP_uStar_fqc"     "PotRad_NEW"
## [112] "Reco_DT_U05"       "GPP_DT_U05"        "Reco_DT_U05_SD"
## [115] "GPP_DT_U05_SD"     "Reco_DT_U50"       "GPP_DT_U50"
## [118] "Reco_DT_U50_SD"    "GPP_DT_U50_SD"     "Reco_DT_U95"
## [121] "GPP_DT_U95"        "Reco_DT_U95_SD"    "GPP_DT_U95_SD"
## [124] "FP_VARnight"       "FP_VARday"         "NEW_FP_Temp"
## [127] "NEW_FP_VPD"        "FP_RRef_Night"     "FP_qc"
## [130] "FP_dRecPar"        "FP_errorcode"      "FP_GPP2000"
## [133] "FP_k"              "FP_beta"           "FP_alpha"
## [136] "FP_RRef"           "FP_E0"             "FP_k_sd"
## [139] "FP_beta_sd"        "FP_alpha_sd"       "FP_RRef_sd"
## [142] "FP_E0_sd"          "Reco_DT_uStar"     "GPP_DT_uStar"
## [145] "Reco_DT_uStar_SD"  "GPP_DT_uStar_SD"
```

Create a Factor Column to Distinguish the Year

First, create an integer column `year`.

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

```
##  num [1:52608] 2004 2004 2004 2004 2004 ...
```

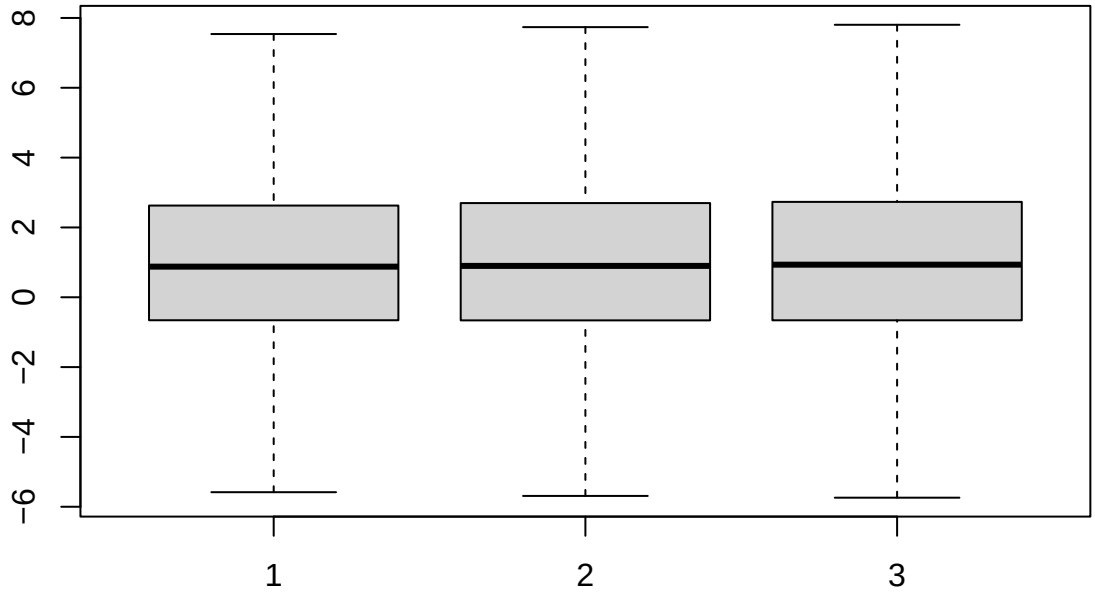
Create a subset data frame from the combined results.

```
Geb2004 <- subset(GebCombResults, year == 2004)
```

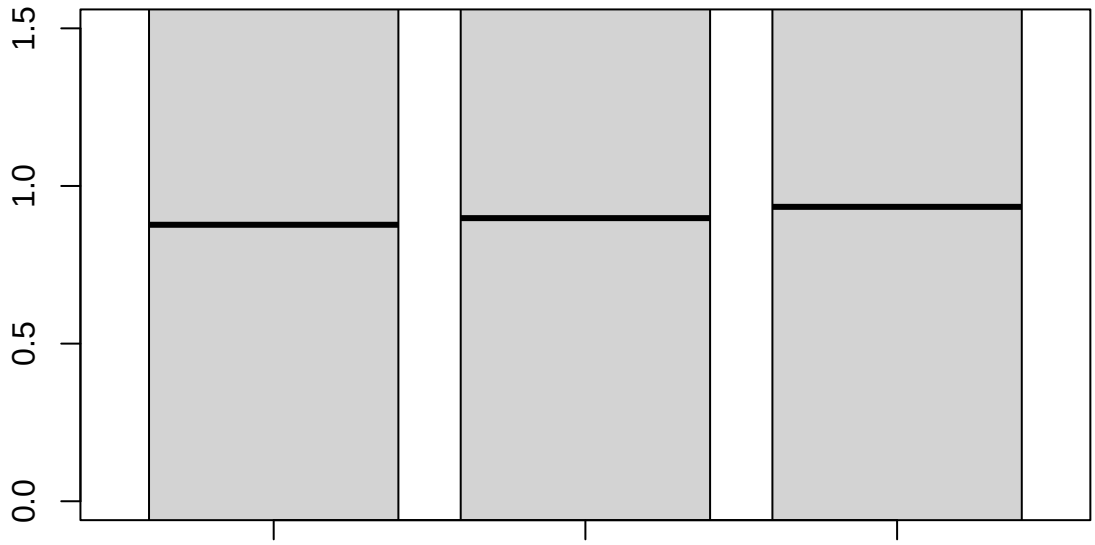
Visualize the Difference of NEE Among the Scenarios

Using a boxplot we can see the changes.

```
boxplot(Geb2004$NEE_U05_f, Geb2004$NEE_U50_f, Geb2004$NEE_U95_f, outline = FALSE)
```




```
boxplot(Geb2004$NEE_U05_f, Geb2004$NEE_U50_f, Geb2004$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 2004 data of the difference scenarios.

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

```
GebScenarios <- c("uStar","U05","U50","U95")
NEE_UStar <- sapply(GebScenarios, function(suffix){
  colName = paste0("NEE_",suffix,"_f")
  mean(Geb2004[[colName]])
})
NEE_UStar
```

```
##      uStar        U05        U50        U95
## -0.5667606 -0.5832546 -0.5626131 -0.5352387
```

Step 6-1-2: Calculate the Statistics

Calculate the mean, standard deviation, and relative error.

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

```
## [1] -0.56196675  0.01992454  0.03545502
```

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.

```r
n_all <- sum(GebCombResults$NEE_uStar_fqc == 0)
n_all
```

```
## [1] 27496
```

```r
n_2004 <- sum(Geb2004$NEE_uStar_fqc == 0)
n_2004
```

```
## [1] 9617
```

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

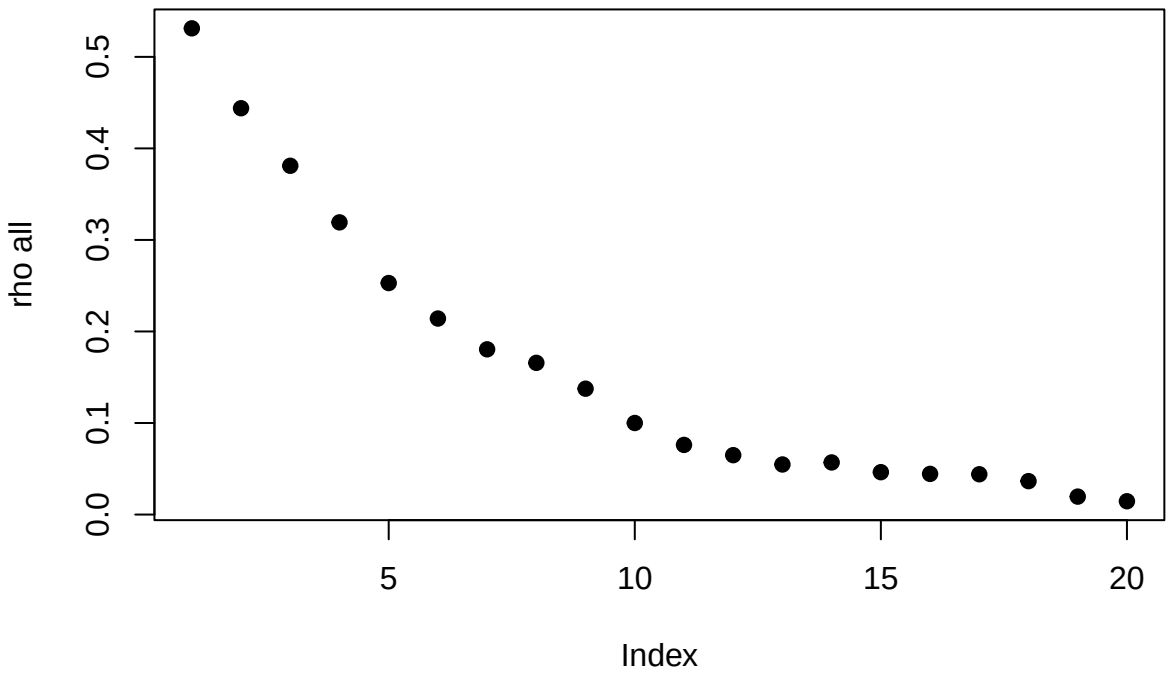
```r
GebCombResults$residual <- ifelse(GebCombResults$NEE_uStar_fqc == 0,
                                  GebCombResults$NEE_uStar_orig - GebCombResults$NEE_uStar_fall,
                                  NA)

Geb2004$residual <- ifelse(Geb2004$NEE_uStar_fqc == 0,
                           Geb2004$NEE_uStar_orig - Geb2004$NEE_uStar_fall,
                           NA)
```

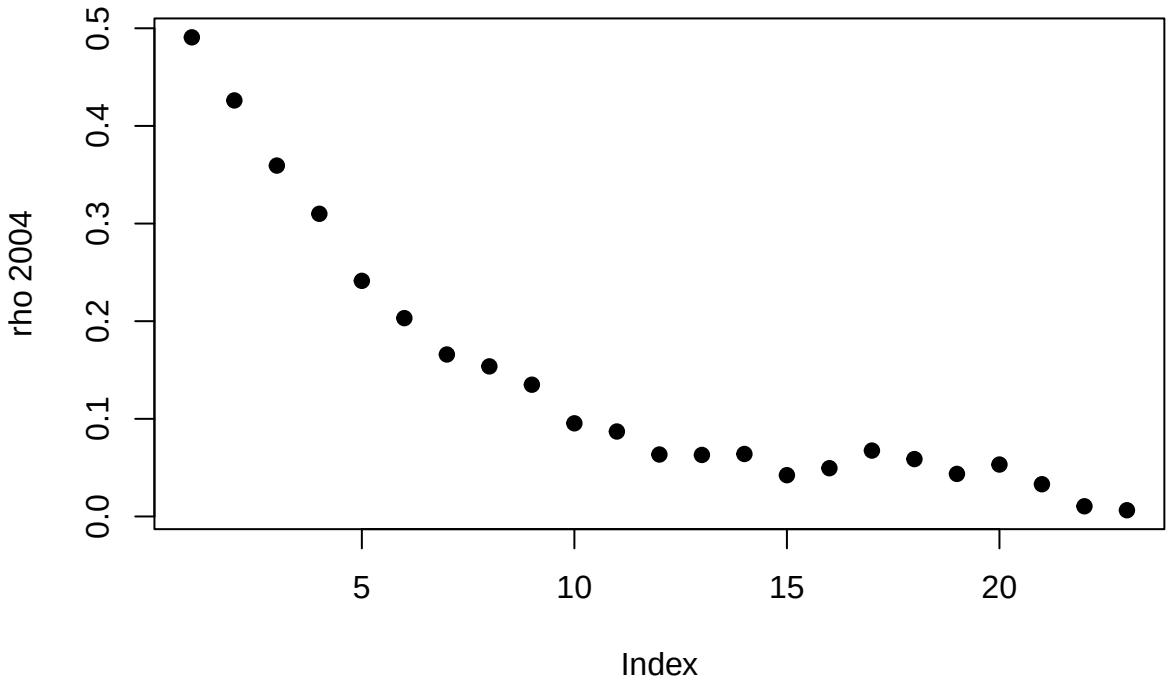
Step 6-1-2: Calculate the Empirical Autocorrelation Function

Calculate the effective autocorrelation components.

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




```
rho_2004 <- computeEffectiveAutoCorr(Geb2004$residual)
plot(rho_2004[-1], ylab = 'rho 2004', 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(GebCombResults$residual, na.rm = TRUE, effAcf = rho_all)
c(nEff_all, n_all)
```

```
## [1]  4699.79 27496.00
```

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

For 2004, create the variable `nEff_2004` and compare to the number of good observations `n_2004`.

```
nEff_2004 <- computeEffectiveNumObs(Geb2004$residual, na.rm = TRUE, effAcf = rho_2004)
c(nEff_2004, n_2004)
```

```
## [1] 1662.043 9617.000
```

Step 6-1-5: Calculate the Mean Annual NEE and Standard Deviation for 2004

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(Geb2004$NEE_uStar_f)

sd_notGapFilled <- Geb2004$NEE_uStar_fsd[Geb2004$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
## -0.56676058  0.02972696  0.05245064
```

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] 0.03578658
```

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.