R package for Microwave links

Bastien Grosso & William Zapdji

May 29, 2016

Abstract

Previous works have shown that it is possible to use the attenuation of a signal between antennas, in ordre to estimate the amount of rain. Several algorithms have been developed in order to separate the dry and wet period and also transform attenuation into rainfall rate.

The aim of this work is to create a package written in R, that gathers different preprocessing functions in order to prepare the ~~raw fi~~les with the attenuation, other functions to predict a baseline in order to calculate the attenuation difference and finally functions to convert the attenuation into rain.

1 Preliminary

We assume that the file that contains the data of the signal intensity has two columns, with in the first column the date and in the second one the intensity of the signal. the two types of data are separated by a semicolon and the first line of the file contains the type of the column, which are ”date” and ”rx_power”. The format of the date is

”YYYY-MM-DD hh:mm:ss”.

The sections of this document represent the file of the package and the subsections the functions. The names are indicated between parenthesis and quotes marks.

2 Preprocessing the data (”Preprocessing”)

The data containing the signal intensity can be with a different time interval, with an irregular rate of measurement or with incorrect values. For these reasons it is important to treat the data first in order to eliminate undefined values or to have a regular data set for example, in case it is needed. Let’s present here different functions for that purpose.

2.1 Averaged intervals (”Average_Datas”)

If one wants a regular data set, it is possible to split the data into intervals and average the values for each interval in order to get one measurement every given time step. This algorithm just aggregate existing data but doesn’t make it regular if measurements are missing.
The function takes three arguments :

1. ”Intensity”, a table containing in the first column the time and in the second one the intensity
2. ”time”, the time unit (seconds,minutes, hours, etc)
3. ”dt”, the size of each time interval

The functions computes then inside each interval the average value and returns a table with the averaged values.

2.2 Check if it is NaN (”Check_NA”)

It can be that there are non valid values in the data set. In order to treat that problem, one can filter the ”not a number” values. If there is more than 2 consecutive invalid values, the function doesn’t change these values.
The function takes only one argument that is the Intensity table. It checks for the indices that are invalid if the previous and next or second next values are valid and takes the average.

2.3 Time differencies (”Time_diff”)

If one wants to have the time expressed as the difference between a starting point and the considered time, this function does the trick. It takes as arguments the data set and the time unit (see section 2.1).

2.4 Import data from a database (”Getpg_SQL”)

If we have one database which contains the raw data in the previous form (see section 1), this function permits you to get theses data and you have the possibility to save in the excel file or use them directly with the different functions.

2.5 Gather the data (”Aggregate_data)

There can be data that have gaps. The previous function to aggregate the data didn’t handle missing values. We want now to define a function that takes any data set, and transforms it into a regular data set with a given period and no gap. This will be done in two steps, first aggregating the values for the existing intervals and putting temporary values (-999999) for missing intervals. The temporary values will be replaced with the next

function.

The function takes three arguments :

1. ”res”, the array that contains the time and intensity
2. ”loop”, the time interval in minutes
3. ”typeformat”, the time format without the seconds (e.g. ”%Y-%m-%d %H:%M”)

The idea is to create an array with the number of values given by the size of the total interval (difference between first time and last time in the initial data) divided by the the size of an interval. This will give the number of intervals to fulfil. We can then check in the original data that each interval exists, if it is the case one can aggregate the data inside the interval in case there are, and if there is no corresponding interval in the original data one can replace it by a temporary value.

The result of this computation will be to have a data set with the number of values corresponding to the number of intervals given by the splitting that was chosen. The next step is to replace the ”-999999” values by average values.

2.6 Complete empty values (”Complete_data”)

This function is a complement to the previous one. It will basically detect the invalid values and look for the closest correct value before and after and average them in order to give a defined value anywhere. It works for any amount of consecutive invalid values.

It takes as arguments the vector to work on and the column in this vector where one wants to replace the values.

3 Baseline estimation (”Baseline”)

The aim of this section is to present a set of algorithm to compute a threshold that will be the reference to classify if a period is dry or wet and compute the rainfall rate consequently.

3.1 Mean value (”Bsline_meanValue”)

The first baseline estimation is the simplest one, that consists in taking the mean value of the whole sample and use it as baseline.

The function takes the intensity vector and returns a vector containing in the first column the time and in the second one the subtraction of the intensity to the baseline computed.

3.2 Separation in time intervals (”Moving_window_static”)

A first step to the moving window algorithm is to split the total time into intervals according to the time. One can then choose the size (in time) of an interval and choose a quantity to be computed on that interval (mean, standard deviation, etc).
The function takes as parameters :

1. ”attenuation”, as usual
2. ”width”, that is the size in time of the interval
3. ”f”, that is the mathematical quantity that has to be computed on an interval

The function then divides the size of the data set by the width of an interval to get the total number of intervals. Then, using the different interval boundary values, it filters the data to get the ones inside each interval and apply the mathematical function on each interval.

The function returns an array with in the two first columns the input intensity and in the third column the quantity computed according to ”f” on each interval.

3.3 Time based moving windows (”Moving_window_dyn”)

We can now add few improvements to the previous function. We want to allow the window to move in time, which means that we fix the width of an interval but the position of this interval will change along the data set in order to compute for each point a value given by the mathematical function chosen. It is possible to make the window symmetric (half of the width before the point and half after) or asymmetric (all the width before). It is also possible to generalise the amount of functions to be computed.
The function takes now as arguments :

1. ”attenuation”
2. ”width”
3. ”fcts”, a list containing the different mathematical functions that we want to compute (e.g. ”list(function(x) mean(x), function(x) sd(x))”)
4. ”dir_window”, a string to choose between symmetric (”symm”) and asymmetric (”asymm”)

For every point of the data set, the algorithm filters the points that are inside the interval (symm or asymm), computes the different quantities that are present in the list of functions and moves to the next point.

It returns a vector with the two original columns and an added number of columns equal to the size of the list of functions to be computed.

4 Conversion into rainfall rate and constants (”Convert_rain”)

The baseline is a tool to separate the rainy and dry events, but once this is done, one needs to convert the attenuation into rainfall rate. The formula to go from the attenuation to the rainfall rate is simply $R = {}^{\alpha}\frac{\gamma}{k}$, where γ is the attenuation and α, k constants that have to be determined.

4.1 Constants and frequencies (”Antenna_freq”)

The way to determine the constants presented above is to use an international reference table named ”ITU-R”. One has to choose the polarisation and frequency of the antenna in order to find the constants.
The function implemented for that takes as arguments the frequency (”freq”) and polarisation (”polari”). It looks into the table at the corresponding polarisation what is the closest frequency and returns the corresponding constants.

4.2 Make it rain ! (”Convert_into_rain”)

Once we have the constants and the attenuation, we can simply compute the rainfall rate estimation. Thus, the function only needs the attenuation and the constants in order to return the rainfall rate.

5 Algorithms to predict rainfall rate (”Algorithms”)

Once we presented different technics to organise the data and estimate the baseline, we can use different algorithms that combine the previous functions.

5.1 Schleiss Decision rule 1 (”Schleiss_S1”)