Precipitation data analysis

Creating IDF curves from tipping bucket precipitation data

Gaby Gründemann

UPC

Debris Flow and Flash Floods

9 November 2016

Introduction

A common tool for analysing precipitation data are intensity-duration-frequency (IDF) relationships. The relationships are usually displayed as curves and are used in water resources management, for instance for the design of stormwater retention tanks. Historical rainfall time series data are used to create IDF curves. The annual maximum rainfall of your time series is fitted to a probability distribution.

A summary of the steps to be taken is given below. First, the raw data in text format will be loaded into R. The data will be converted into a time series format, whereafter the moving averages of different durations will be computed. The moving averages are then aggregated for the different years, resulting in a table with the maximum precipitation depth per duration period. The maximum intensity, in millimeters per hour, is then computed. With build-in packages there are two different distributions fitted to the data; GEV and Gumbel. The one with the best fit will be used to compute the IDF curves.

The aim of this report is to explain a script in the free open source program R to compute the IDF curves. The script is called “Script_Grundemann.R” and could be accessed through the following link: https://github.com/FRM5/TimeSeriesAnalysisToolbox/blob/Gaby/Script_Grundemann.R

The script is set up in an automised way using loops. The only things which have to be determined beforehand are the chosen duration in minutes, the return periods for computing the IDF curves, the working directory and the file with the precipitation data. The file with the precipitation data consists of two columns, one with the dates and one with the times the tipping bucket provided.

Libraries

There are a number of libraries loaded into R in order to for the script to function. These are:

- xts
  - For handeling the time-series, extending zoo
- plyr
  - For handeling data, in particular used for the function “rename”
- hydroTSM
  - For time series management related to hydraulics, in particular for computing the moving averages
- lubridate
  - For advanced working with dates, in particular used for the function “year”
- extRemes
  - For performing the extreme value analysis and the sensitivity analysis of the distribution
- reshape2
  - For flexible reshaping data table as input for the graph of the IDF curves
- ggplot2
  - For plotting the data tables into getting the IDF curves

Settings

The only settings which needs to be done in order for the whole script to run are the duration and the return period. The duration, in minutes, is the length of the intervals needed for computing the moving averages. The return periods are given in years. The different settings used are displayed below:

```
Duration=c(5, 15, 30, 60, 120, 360, 720, 1440)    # duration in minutes
Return_Period=c(2, 5, 10, 50, 100, 200)           # return periods in years
```

Loading the data

First the working directory will be specified. The data file used for the script is located will be set as the working directory. The data file itself is in a .txt format and imported in R with the *read.table* function. There are two columns, one for the dates "date" and one for the corresponding times "time". These two columns will be merged into one column with a date format recognised by R. In order to decrease the computational time of the whole script, the seconds are neglected in the new "DateTime" column. There will be a new column added to the data with the depth corresponding to one click of the tipping bucket, which is 0.1 mm.

```
setwd("~/Documents/UPC/Assignment")
My_Data=read.table("21.txt", header=TRUE)
My_Data <- data.frame(as.POSIXct(paste(My_Data$date, My_Data$time), format="%m/%d/%Y %H:%M"))
colnames(My_Data) <- c("DateTime")
My_Data$Depth <- 0.1
```

Creating the time series

The data is converted into XTS format (*xts*), which allows R to understand that the data is a time series. During heavy storms, the precipitation will be intense so within one minute the tipping bucket will register multiple clicks of 0.1 mm. These clicks need be summed per minute to determine the minutely maximum intensities. This is done using the *aggregate* function.

Hereafter, there is an empty timeframe created (*zoo*) over the whole length of the precipitation tipping bucket data per minute. This empty frame is needed to create a complete minutely data series over the whole time span. Then the aggregated time series is merged (*merge*) with the empty frame, whereafter the missing "NA" values are replaced by 0 values (*[is.na() <- 0]*). This is needed, because there cannot be missing values in the data. The data is then converted to a data frame format (*data.frame)*, because this is required for the function *rollapply* to compute the moving averages for the different durations. This will be further explained in the next part.

```
TS <- xts(My_Data[,-1], order.by=My_Data[,1])
TS_Agg <- aggregate(TS, identity, sum)
TS_Empty <- zoo(, seq(start(TS_Agg), end(TS_Agg), by="1 min"))
TS_Merged <- merge(TS_Agg, TS_Empty)
TS_Merged[is.na(TS_Merged)] <- 0
TS_DF <- data.frame(date=index(TS_Merged), coredata(TS_Merged))
```

Moving averages

There are two ways explained to compute the moving averages. The first one is by using the moving average *ma* function, which works with a zoo class. The second one is by using the *rollapply* function, which requires a data frame class. However, in this script they work according to the same principle because the same loop is used. The loop runs for the different durations specified before. For the first run (i=1), the moving average is computed by aggregating over the first duration period (5 minutes), moving 1 minute every time (*ma* or *rollapply*). Then the moving average is converted to a data frame format (*data.frame*) with two columns (one with the minutely time steps and one with the aggregated

values for those time steps). Then a new data frame is created with the annual maximum values for the given duration. The same is done for the durations afterwards, but then the annual maximum values are added to the previous table.

```
# Moving average with ma
for(i in 1:length(Duration)){
  Mov_Avg <- ma(TS_Merged, win.len = Duration[i], FUN = sum)
  Mov_Avg <- data.frame(DateTime=index(Mov_Avg), "TS_Agg" = coredata(Mov_Avg))
  if (i==1){
    Max_Year <- aggregate(Mov_Avg$TS_Agg ~ year(Mov_Avg$DateTime), FUN = max)
  }else{
    Mov_Avg<- aggregate(Mov_Avg$TS_Agg ~ year(Mov_Avg$DateTime), FUN = max)
    Max_Year<-merge(Max_Year, Mov_Avg, by = "year(Mov_Avg$DateTime)")
  }
  if(Duration[i]<60){
    Colname=paste0(Duration[i], "min")
  }else{
    Colname=paste0(Duration[i]/60, "hr")
  }
  Max_Year<-rename(Max_Year, replace = c("Mov_Avg$TS_Agg" = Colname))
}
Max_Year<-rename(Max_Year, replace = c("year(Mov_Avg$DateTime)" = "Year"))
Max_Year <- Max_Year[-c(1), ] # remove the year 1994
```

```
# Moving average with rollapply
for (i in 1:length(Duration)) {
  Mov_Avg <- TS_DF[1]
  Mov_Avg$Precipitation <- rollapply(TS_DF[2], as.numeric(Duration[i]), sum, fill = NA, align='left')
  if (i==1){
    Max_Year <- aggregate(Mov_Avg$Precipitation ~ year(Mov_Avg$date), FUN = max)
  }else{
    Mov_Avg <- aggregate(Mov_Avg$Precipitation ~ year(Mov_Avg$date), FUN = max)
    Max_Year<-merge(Max_Year, Mov_Avg, by="year(Mov_Avg$date)")
  }
  if(Duration[i]<60){
    Colname=paste0(Duration[i], "min")
  }else{
    Colname=paste0(Duration[i]/60, "hr")
  }
  Max_Year<-rename(Max_Year, replace = c(TS_Agg = Colname))
}
Max_Year<-rename(Max_Year, replace = c("year(Mov_Avg$date)" = "Year"))
Max_Year <- Max_Year[-c(1), ] # remove the year 1994
```

In order to give correct column names, the last part of the code is added. It will change the name of the column of the table with the yearly values to the corresponding durations. If the durations are below the hour, it will add the word “min” to it. If the durations are more than one hour, it will divide the duration by 60 to get the number of hours and add the word “hr” to it.

The main difference between the *ma* and the *rollapply* function is the way it determines the moving average. This can be shown on the tables below. The first table shows the first two rows from the annual maximums using the *ma* function and the second table shows the *rollapply* function. The data from 1994 is just one precipitation event that took place on the 19th of December in 1994. The first precipitation event of 1995 takes place on the first of January in 1995. The precipitation event on the first of January is a lot more intense than the recorded event of 1994. Therefore, the high values for the longer durations are actually capturing the precipitation that took place in 1995 instead of 1994.

Additionally, there is only one event in 1994 included in the original dataset. This is not representative for a whole year and this will increase the error of the IDF curves. This is shown by the AIC and BIC values, which are measures for the error and to determine the sensitivity of the model. AIC is the Akaike Information Criterion. BIC is the Bayesian Information Criterion. Both parameters are used for sensitivity analysis later on in this script, in the section "Gumbel or GEV distribution", to test which type of distribution should be used. For both the AIC and BIC is the lowest value preferred, because they both assign a penalty for overfitting.

This sensitivity analysis has been performed both with and without the year 1994. It has shown that when the year 1994 is removed from the data the values of AIC and BIC decrease significantly. For instance the values of AIC for the Gumbel distribution decrease from 80.3 to 71.2 after removal of the year 1994. Therefore, there has been chosen to remove the year 1994 from the dataset in the final version script. This is done by the line *Max_Year <- Max_Year[-c(1),]*.

Year	5min	15min	30min	1hr	2hr	6hr	12hr	24hr
1994	0.5	1.1	1.5	2.2	3.1	3.2	3.2	7.8

Table 1. Annual maximums (Max_Year) when the "ma" function is used

Year	5min	15min	30min	1hr	2hr	6hr	12hr	24hr
1994	0.5	1.1	1.5	2.2	3.1	3.2	7.8	9.7

Table 2. Annual maximums (Max_Year) when the "rollapply" function is used

Maximum Intensities

The maximum intensities are then computed, by converting the table with the annual maximum values (Max_Year) to the amount of precipitation per hour. The code is displayed below.

```
# The maximum intensities are in [mm/hr]
Max_Intensity <- Max_Year
for(i in 1:length(Duration)){
  Max_Intensity[,i+1]=Max_Intensity[,i+1]*(60/Duration[i])
}
```

Gumbel or GEV distribution

In order to fit either the Gumbel or the GEV to the data, the function *fevd* from the "extRemes" library is used. Fevd stands for Fit an Extreme Value Distribution to the data. The script for the Gumbel distribution is shown below. First an empty matrix is created. Thereafter, a loop is run for the different durations. The maximum intensity table is used to fit both the Gumbel and the GEV distribution to the data. With the *return.level* function, the values of the durations for the different return periods are assigned to a new table. The same applies for this loop as for the previous one with the moving averages, namely that for the first duration the table is created and for the other durations they are merged.

Thereafter, the sensitivity analysis is performed, to determine if the Gumbel or the GEV distribution should be used. The *fevd* function automatically computes statistics, among which are the AIC and BIC values. They have already been introduced in the paragraph "moving averages". The distribution which has the lowest AIC and BIC values is chosen for the final plot of the IDF curves. That is determined with a small *if-else* loop.

```
# Apply the Gumbel distribution to the data
Gumbel <- matrix(data=NA, ncol=length(Return_Period))
for(i in 1:length(Duration)){
  value=as.numeric(Max_Intensity[,1+i])
  Fit_Gum <- fevd(value, Max_Intensity, scale=1, type="Gumbel")
  RetLev <- return.level(Fit_Gum, conf=0.05, return.period=(Return_Period))
  if (Duration[i]==1) {
    Gumbel <- t(data.frame(as.numeric(RetLev)))
  }else {
    Gumbel2 <- t(data.frame(as.numeric(RetLev)))
    Gumbel <- rbind(Gumbel, Gumbel2)
  }
}
Gumbel<-Gumbel[-which(apply(Gumbel,1,function(x)all(is.na(x)))),]
row.names(Gumbel) <- Duration


# Sensitivity
AICBIC_GEV <- summary(Fit_GEV, silent=TRUE)
AICBIC_Gum <- summary(Fit_Gum, silen=TRUE)
if (AICBIC_Gum$AIC & AICBIC_Gum$BIC < AICBIC_GEV$AIC & AICBIC_GEV$BIC) {
  FINAL <- Gumbel
} else {
  FINAL <- GEV
}
```

Plotting the IDF curve

The final part is plotting the IDF curves. First, the final table with either the Gumbel or GEV distribution is copied and converted to a dataframe format using *data.frame*. Then a new table will be created with the different return period, which were determined in the settings at the beginning, as the column names. First an empty matrix is computed. This matrix will hold the names of the return periods plus the word “years”, needed for the correct display of the legend. This is written by a small loop. The column names are then used in the final IDF table.

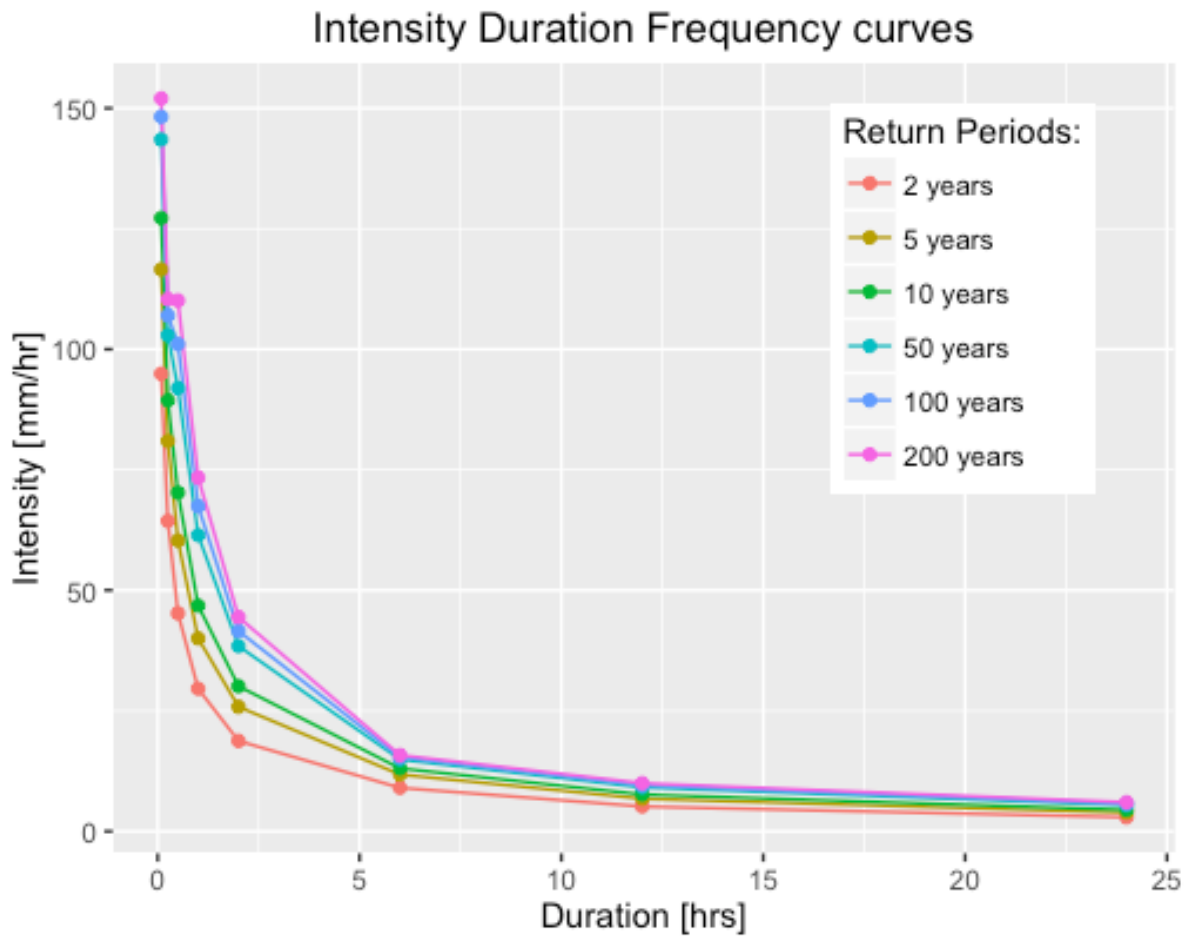
The next step is to convert the minutely values to hourly values, so the X-axis will be displayed in hourly values. This is done using the following function:
Hours <- as.data.frame(c(as.numeric(as.character(t(Duration))))/60)
This column is then added to the final IDF table (*cbind*), in order to display it correctly. The next step is to order the table by the different durations, using the function *melt*. With the function *ggplot* from

```
IDF_Table <- data.frame(FINAL)
Col_Names_RP <- matrix(data=NA) # Create an empty data frame to host the names of the return periods
# That data frame will contain the return periods, needed for the legend
for (i in 1:length(Return_Period)){
  Col_Names_RP[i] <- paste0(Return_Period[i], " years")
}
colnames(IDF_Table) <- c(Col_Names_RP) # Copy the names to the IDF table
Hours <- as.data.frame(c(as.numeric(as.character(t(Duration))))/60)
# This converts the final durations to hours instead of minutes for the X axis
colnames(Hours) <-"Duration [hrs]" # change column name
IDF_Table <- cbind(Hours, IDF_Table) # add the duration table to the final IDF table.
IDF_Plot<- melt(IDF_Table, id.vars="Duration [hrs]") # order the table by duration, needed for the plotting
IDF_Plot <- ggplot(IDF_Plot, aes(x=`Duration [hrs]`, y=value, color=variable)) + geom_line() + geom_point() +
  ylab("Intensity [mm/hr]") + labs(title="Intensity Duration Frequency curves") + labs(color="Return Periods:") +
  theme(legend.position = c(.8, .7))
plot(IDF_Plot)
```

the library ggplot2 the IDF curves are plotted in a graph. There are some functions added to the standard plot function. The standard function only includes the table, x-axis, y-axes and colour. The x-axis is displayed correctly as Duration [hours] but the y-axis shows value. Therefore, the following is added

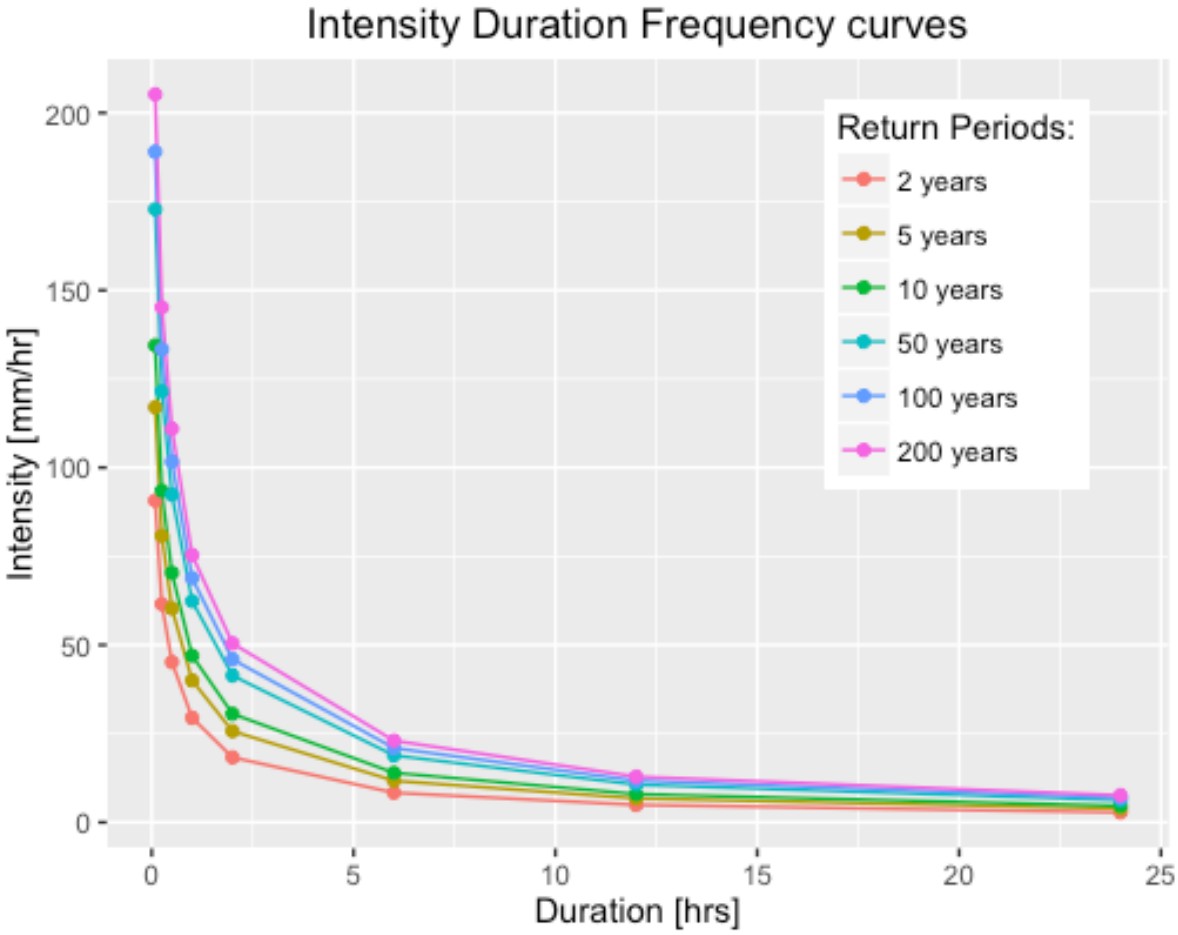
- geom_line() to add the line in the graph
- geom_point() to add the points to the graph
- ylab("Intensity [mm/hr]") to add the correct label for the y-axis
- labs(title="Intensity Duration Frequency curves") to add the title
- labs(color="Return Periods:") to add the title to the legend
- theme(legend.position = c(.8, .7)) to put the legend inside of the frame

The final plot is displayed below.



Discussion

The final plot determined by this script with the provided data is the script displayed above, using the GEV distribution, since that results in lower AIC and BIC values. However, for the 200 year return period, the values for 15 and 30 minute durations are extremely close (110.4 and 110.1 mm/hr) Therefore, it has a strange almost horizontal line at that duration. I also examined the plot for the Gumbel distribution. That plot is displayed on the next page. It can be seen from both the plots and the tables, that the Gumbel distribution results in larger extremes. The maximum intensity for the 200-year return period for the Gumbel distribution is 205 compared to 152 for the GEV distribution. For the lower return periods, however, the values for the Gumbel distribution are a bit lower. What could be observed, however, is the smooth transition of the curves.



The data set this script has been developed with is a data set with 113,834 time steps, from December 1994 until August 2016. Since the year 1994 has been removed from the data set because it is not representable for a yearly maximum due to the lack of sufficient data (see “Moving Averages”) the actual data the IDF curves have been computed with are from January 1995 until August 2016. A way to reduce the error even further (i.e. lower the AIC and BIC values) a dataset with more years will need to be used.