Intensity-Duration-Frequency Curve

Code available on GitHUB:
https://github.com/FRM5/TimeSeriesAnalysisToolbox/commit/5676d2ae963eece09401a9251a56ed0599c5b06a

This set of codes requires a number of built-in packages in order to run.

```
library(lubridate)
library(xts)
library(timeDate)
library(extRemes)
library(ggplot2)
```

R reads the data in .txt file format. Time is reformatted up to the minute accuracy using *as.POSIXct* (another function that works similarly, *strptime*, does not work for *xts*). This is necessary due to some data has repeated seconds during high intensity rainfall, such that the tipping bucket rain-gauge clicks more than once within a second. This also allows for easier data aggregation by the minute later. The date and time is combined in the same column to create a time series, where time has to be unique.

```
mydata=read.table("filename.txt",header=TRUE,col.names=c("Date","Time"))
mydata$dt <- as.POSIXct(paste(mydata$Date,mydata$Time),format = "%m/%d/%Y %H:%M")
mydata$depth <- 0.1
mydata$Date <- NULL
mydata$Time <- NULL
```

Function *tapply* sums (or aggregates) the data by the minute and returns a single column variable with the date and time as rownames. *xts* creates a time series. In order to use the moving average function in the *zoo* package, time series has to be uniform (i.e. 1 minute interval), as the function does not specify the window unit. In this case if the time series is in 1 minute interval, if window length of 5 is specified, it means 5 minutes window.

A dummy time series of time interval of 1 minute with depth 0 mm (indication no rain) is created. The dummy matrix is merged with the time series created previously. The argument *all.x = TRUE* means, the merging keeps all the data in *x* and only keep the *y* that is matched. This is necessary to be defined if using *xts* instead of *zoo*.

```
#Create uniform time series with 1 min intervals
grpdata <- tapply(mydata$depth, mydata$dt, sum)
mydata_ts <- xts(grpdata,as.POSIXct(rownames(grpdata)))
bymin <- seq(start(mydata_ts), end(mydata_ts), "min")
dummycell <- xts(rep(0,length(bymin)), order.by=bymin)
new_TS <- merge(dummycell, mydata_ts, fill=0, all.x = TRUE)[,2]
```

Due to the huge file size created, the unused variables in the R environment have to be cleared to allow more memory space to proceed the run. Otherwise, the memory could be increase by using *memory.limit()*

```
rm(mydata,grpdata,mydata_ts,bymin,dummycell) #Remove to free-up memory space
```

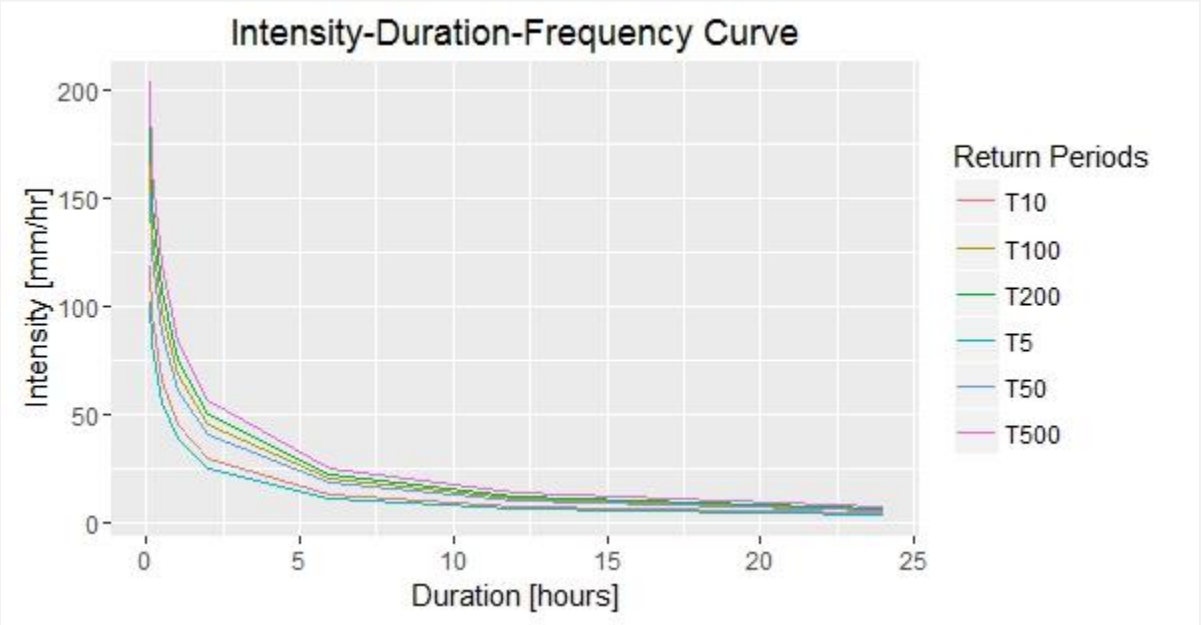
The *rollapply* function sums the data by the minute-window specified in each *for-loop* run. The moving averaged data is aggregated by the year using the *aggregate* function within the *if-loop*. The *if-loop* is only used for the purpose of tabulating the results in an array where the *if*-condition is only used to set the first and second column (of year and first duration) and the *else*-condition is used for the remaining durations with the year column removed.

```
#Apply moving average for user-defined durations and find annual maximum
tdur <- c(5,10,30,60,120,360,720,1440) #Duration in minutes
t <- tdur/60
dma <- data.frame(date = index(new_TS))
ctr=0
for (i in tdur){
  dma$depth <- rollapply(coredata(new_TS),i,sum,fill=0)
  #assign(paste("ma", i, sep = ""), dma) #Activate to display new variables
  dma$depth <- dma$depth/t[ctr+1] #Convert to intensity mm/hr
  if (ctr < 1){
    maxd <- aggregate(dma$depth ~ year(dma$date),dma, FUN=max)}
  else{
    maxd[[ctr+2]] <- aggregate(dma$depth ~ year(dma$date),dma, FUN=max)[,-1]}
  ctr=ctr+1
}
colnames(maxd) <- c("Year",paste(round(t,digits = 1), "hour"))
```

The built-in *fevd* (fit to extreme value distribution) function is used to fit the data of yearly maximum intensity over the different durations. Two distribution types are compared using the Akaike Information Criterion (AIC, Akaike, 1974) and Bayesian Information Criterion (BIC, Schwarz, 1978) values, such that the smaller AIC and/or BIC is the better fit. Both BIC and AIC resolve the problem of overfitting by introducing a penalty term for the number of parameters in the model. The penalty term is larger in BIC than in AIC. In this code, it looks for both smaller AIC and BIC, however it can also be modified for looking at AIC only or BIC only.

```
# GEV or Gumbel method
rp <- c(5,10,50,100,200,500) # Specify return period
outrp <- data.frame(matrix(ncol = length(tdur), nrow = length(rp)))
for (i in 1:length(tdur)){
  val = as.numeric(maxd[,1+i])
  fitgev <- fevd(val, maxd, type="GEV")
  fitgmb <- fevd(val, maxd, type="Gumbel")
  fit1 <- summary(fitgev)
  fit2 <- summary(fitgmb)
  #Choose the best fit using AIC and BIC
  if (fit1$AIC & fit1$BIC < fit2$AIC & fit2$BIC){
    mlefit <- fitgev}
  else {mlefit <- fitgmb}
  rlmle <- return.level(mlefit, conf = 0.05, return.period = rp)
  outrp[i] <- as.numeric(rlmle)
}
```

The *ggplot* function was used to plot the graph below;



For this assignment, the length of data received (18.txt) was for a period of 8 years which took approximately 50 minutes to run. It is expected that much longer computational time required for longer period of data.

Another set of code (available on GitHUB: https://github.com/FRM5/TimeSeriesAnalysisToolbox/commit/5fde6629ad411601b7254ed718313dc5e4a751da) does not utilizes the *cut* function which returns the frequency of clicks in each duration, has a shorter computational time. However, moving average is a preferred method.

Design rainfall

(The generation of design rainfall is optional for this assignment.)

There are two approach to create the design rainfall – i) S-shape graph and, ii) alternating block.

Here, the alternating block method will be discussed.

Method: A function is fitted to a specific return period for example T100 i.e. using the IDF function $I = \frac{c}{Td^e + f}$, where e, c, and f is a constant. Suggestion to use the *nls* function and specify a start value for the constants to start the iteration.

We can obtain the duration and intensity of a specific duration, usually approximates the time of concentration of the catchment. In this case, since no details of the catchment were known, 24-hour duration can be used. The cumulative depth and incremental depth are calculated. A hyetograph is created such that the maximum rainfall will be at the center, the next largest block is placed to the right of the maximum, third largest block is placed to the left of the maximum, and so on. This is done with the *if-loop*.

```
#Creating synthetic rainfall
idfdata <- data.frame(t, outrp$`100`)
colnames(idfdata) = c("Td","i")
idffit <- nls(i ~ c/(f+Td^e), start=list(c=1, f=1, e=1), data=idfdata)
fitdat <-predict(idffit, newdata = data.frame(Td=c(seq(1,24))))
designR <- data.frame(c(seq(1,24)))
designR[2] <- fitdat
designR[3] <- designR[1] * designR[2]
designR[4] <- c(designR[1,3],diff(as.matrix(designR[3])))

rain <- as.matrix(unlist(designR[4]))
med <- max(rain)
num <- which.max(rain)
Q <- rain[-num]

if (length(rain) %% 2 != 0){ #check if function is odd
  tl <- sort(which(index(Q) %% 2 == 0),decreasing = TRUE)
  hd <- sort(which(index(Q) %% 2 != 0),decreasing = FALSE)
} else {
  hd <- sort(which(index(Q) %% 2 == 0),decreasing = TRUE)
  tl <- sort(which(index(Q) %% 2 != 0),decreasing = FALSE)
}
designR[5] <- c(Q[hd],med,Q[tl])
colnames(designR) = c("Duration","Intensity","Cum.Depth","Inc.Depth","Precipitation")
```

The resulting bar plot is as per below.