‘NamRain’ — A generator for stochastic multimodal annual time series of daily precipitation at single locations

Martin Köchy, Universität Potsdam, Am Neuen Palais 10, 14469 Potsdam
office@martinkoechy.de

Date of online publication: 2010-11-22, corrected 2020-01-27 (columns in input files)

Abstract

I describe the console program ‘NamRain’ that produces stochastic annual time series of daily precipitation at single locations using examples from Namibia. The input required for each location are long-term monthly records of precipitation. First, the frequency distribution of monthly rainfall, daily amounts, and rainy days is determined using an *R* script. The distribution is imported into the program. The program randomly selects precipitation events from an exponential distribution and rejects the event if it does not fit the specified statistics. The simulated time series have the same annual and monthly distribution as the original data within 100 mm categories. Daily rain amounts match long-term statistics but have not been the focus of the algorithm. NamRain should be applied in situations where the emphasis is on seasonal variation.

Introduction

Studies simulating the effect of climate on natural ecosystems depend on the use of many time series representing the current climate and its daily, monthly, and annual variability. The nature of simulations with stochastic models requires the use of more time series than can usually provided by historic measurements from climate stations. For many parts of the world algorithms to produce synthetic time series have been described (e.g., Zucchini et al. 1992, Srikanthan and McMahon 2001, Köchy 2006). Such an algorithm has not yet been described for Namibia.

Here I describe an algorithm that produces stochastic time series of daily precipitation. As an example I use stations in Namibia of interest for the BIOTA project and for which data were freely available. I also describe the algorithm to derive the parameters for other stations. The focus of the algorithm is on providing a solution with few parameters that can be easily adjusted for other stations, climate change scenarios, and used as a module in computer programs.

Methods

Climate data for Namibia was downloaded from the National Climatic Data Center of the United States of America (www7.ncdc.noaa.gov). The database comprises 31 Namibian stations but only seven stations (Fig. 1) have continuous records ranging from 9 to 18 years. One station (Gobabis) with only two complete years was included because of its proximity to ongoing research projects. I assumed that all no-data entries corresponded to zero rainfall on a given day and converted the values from inches to mm. I screened the precipitation data for obvious errors. Six entries showed daily precipitation amounts of >150 mm up to 415 mm. In reality, rainfall events of this magnitude are extremely rare and have been observed only twice while climate was recorded (249 mm on 1986-02-02 in Ombalantu and 208.5 mm on 1952-02-18 in Kamanjab, Olszewski 2007). None of the days corresponded with historically known extreme rainstorms, therefore the data for these days was excluded.

The monthly distribution of rainy days in Namibia shows a skewed single or double-peaked distribution. The mean daily rain amount per month shows no recognizable distribution. This prevented using the same algorithm as in the ReGen simulator (Köchy 2006) developed for symmetric rain distributions in the eastern Mediterranean. Instead, I constructed a two-part model (Srikanthan and McMahon 2001). The first part determines the whether a day is a rainy day (amount > 0.5 mm) by comparing the average historic probability of monthly rainy day occurrence with a uniformly distributed pseudo-random number. The second part determines the daily rain amount drawn from a negative exponential distribution. The drawn daily rain amount is limited by the maximum daily amount observed historically in the same month, or, if less than five data have been observed in the same months, the maximum daily amount of all months with fewer than five data (105 mm). This prevents the drawing of very high amounts from the unbounded negative exponential distribution. This maximum threshold is very arbitrary and the user could change the value to 30 mm, which is a more frequently observed amount in dry months. I also tested using the lognormal and the skew normal distributions. Both performed poorer than the exponential distribution. The parameters for the negative exponential distribution were calculated from complete months of daily historic records.

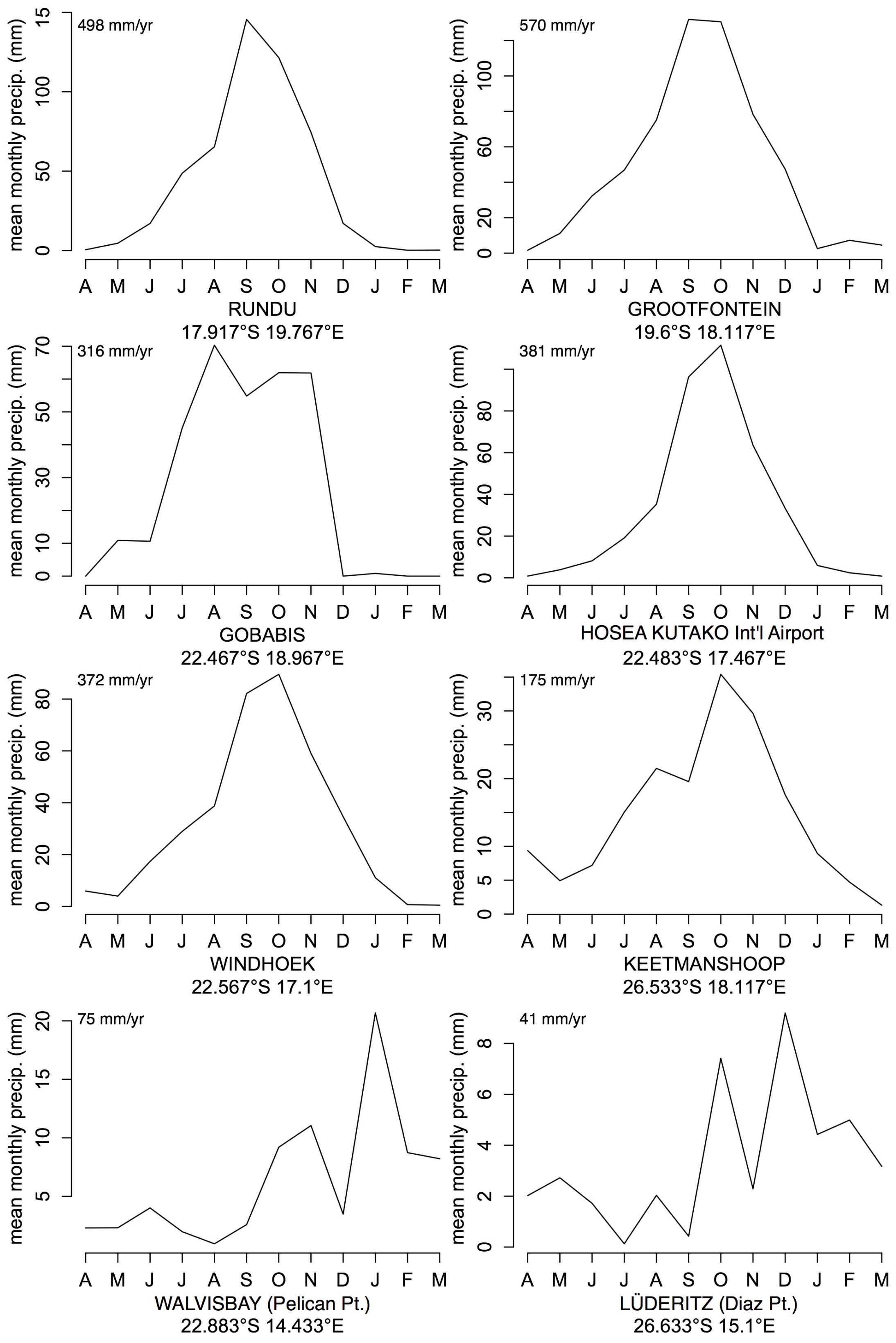

Fig. 1. Mean monthly and annual precipitation amounts at eight selected stations in Namibia. Data calculated from complete months within records of World Climate Data Center.

For each station with >9 yr records I compared the distribution of annual precipitation amounts of stochastic time series of 20 years with those of historic records (Fig. 2). The stochastic time series generally have s smaller range and in three out of eight stations have a median that is higher than the historic median. The medians of the stochastic series of all long-term stations are within the historic inter-quartile range. Means and confidence intervals of historic and synthetic daily data matched closely (Fig. 2).

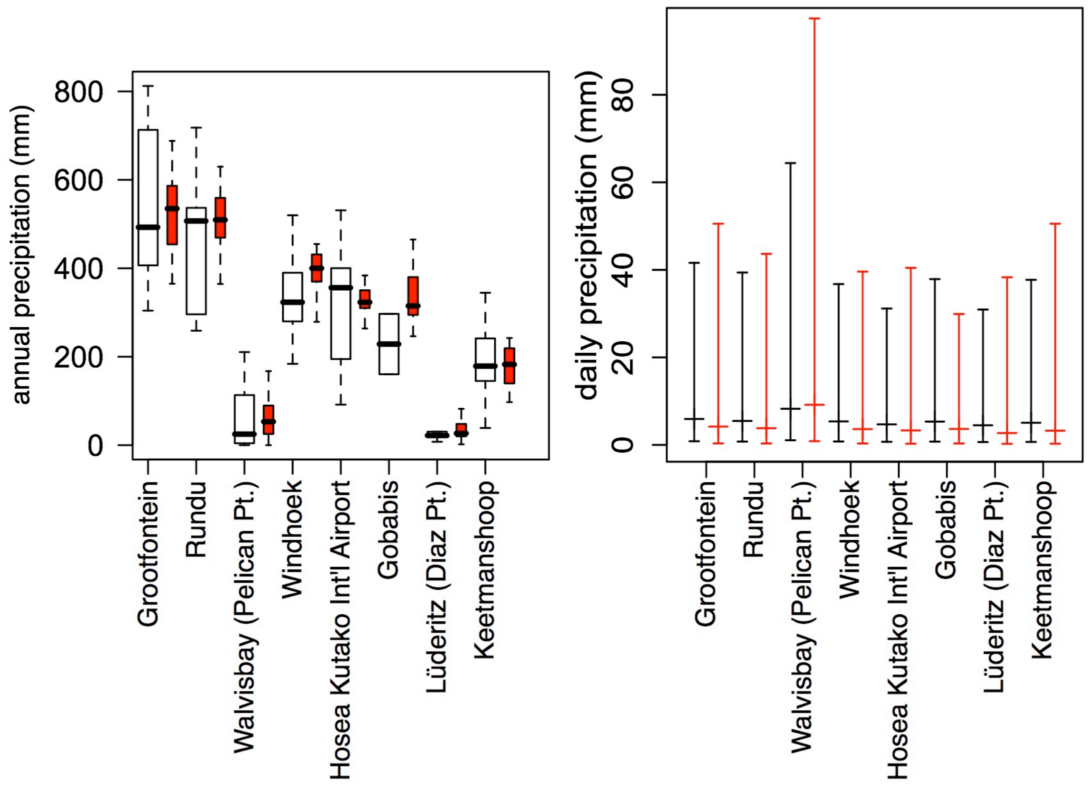

Fig. 2. Comparison of synthetic (red) with historic (black) time series of daily precipitation. In the left panel, the boxplot whiskers indicate the inter-quartile range times 1.5, the boxplot box indicates the 25%, 50%, and 75% quantiles. In the right panel, the ends of the bars indicate the retransformed 95%-confidence interval on the natural logarithm of the daily data and the centre of the bar indicates the retransformed mean of the logarithm of the daily data.

To further improve the distribution of annual precipitation at each simulated station, NamRain versions >1.0.6 check whether a simulated time series fits to the empirical distribution of annual precipitation. If this is not the case, the time series is discarded. In addition, an empirical correction factor is applied to the simulated daily rain to obtain the same mean annual precipitation as observed historically.

Results

Preparation of the data was carried out in JMP 7.0 (SAS, Cary, NC, U.S.A.) and *R* (R Development Core Team, 2008) and is presented in Appendix 1. The algorithm is coded in C++ (Appendix 2). It can be used as a standalone application or as a function. The input files must have six columns, separated by tabulator codes ("\t"): station, month, probability of a rainy day, rate (=1/mean) of the exponential distribution, standard deviation of the rate, and maximum observed amount in the month. Each row represents one calendar month. In the stand alone application the user can enter the station, the length of the time series and the number of time series to be produced. The output file(s) contain the daily records in rows for each year. Data are separated by tabulators.

Discussion

On the annual level, the synthetic time series provide acceptable input for stochastic simulations. One has to keep in mind that their inter-quartile ranges are smaller than those of historic time series. This may be partly due to the shorter length of the historic series, where single extreme years would affect the position of the quantiles. If a more similar range is desired the user must draw an appropriate sample from a pool of synthetic time series. So far, I have not yet validated the performance of the synthetic time series at the monthly or daily level.

Acknowledgements

This study is part of the BIOTA Southern Africa project financed by the German Federal Ministry of Science and Education (BMBF). The author alone is responsible for the content of this publication.

References

Köchy, M. 2006. Stochastic time series of daily precipitation for the interior of Israel. Israel Journal of Earth Sciences **55**:103-109.

Olszewski, J. 2007. Wet Days Part 2: extreme events in the regular picture 05 Oct 07. Namibia Economist Online Edition http://www.economist.com.na/content/view/1972/57/, viewed 2008-12-19.

R Development Core Team. 2008. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna (Austria).

Srikanthan, R., and T. A. McMahon. 2001. Stochastic generation of annual, monthly and daily climate data: A review. Hydrology and Earth System Sciences **5**:653-670.

Zucchini, W., P. Adamson, and L. McNeill. 1992. A model of southern African rainfall. South African Journal of Science **88**:103-109.

Appendix 1

```
# R-Script for analyzing daily precipitation data from Namibia.
# Data downloaded from the National Climatic Data Center
#  original data file ("CDO5152001650765") converted to tab-separated text

show.graphs=FALSE

d=read.table("CDO5152001650765.txt", sep="\t", na.strings=c("99.99", "999.9"), header=TRUE,
as.is=TRUE)
my.d=data.frame(station=d$STATION)
my.d$yr=d$YEARMODA%/%10000
my.d$mo=(d$YEARMODA-my.d$Year*10000)%/%100
my.d$day=d$YEARMODA%%100
my.d$q=ifelse(is.character(type.convert(substr(d$PRCP,nchar(d$PRCP),nchar(d$PRCP)),
as.is=TRUE)),substr(d$PRCP,nchar(d$PRCP),nchar(d$PRCP)),NA) # check whether a code is attached
my.d$pr.in=as.numeric(substr(d$PRCP,1,nchar(d$PRCP)-ifelse(is.na(my.d$q),0,1))) # remove
letter code
my.d$precip=ifelse(my.d$pr.in<99,my.d$pr.in*0.254,NA) # convert from inches to mm

### OR ###
# use subset of data with only complete years
my.d=read.table("nm complete months.csv", sep="\t", na.strings=".", header=TRUE)
colnames(my.d)
colnames(my.d)<-c("station", "yr", "mo", "day", "precip")


#### some simple stats
# select years
historic=my.d[,my.d$yr>1960 && my.d$yr<=2008] # be liberal, use as much data as possible
historic$mo = (historic$mo+4)%%12
# monthly rain amount each year
historic.monthly.rain=aggregate(historic$precip, by=list(mo= historic$mo, yr= historic$yr,
station= historic$station), function(x){sum(x, na.rm=TRUE)})[,c(3,2,1,4)]
# mean monthly rain amount
historic.mean.monthly.rain=aggregate(historic.monthly.rain$x, by=list(mo=
historic.monthly.rain$mo, station = historic.monthly.rain$station), mean)


# annual rain each year
historic.annual.rain=read.delim("nm complete years.txt")
colnames(historic.annual.rain)<-c("station", "year", "N", "x")
historic.mean.annual.rain=aggregate(historic.annual.rain$x, by=list(station=
historic.annual.rain$station), mean)

stations.names=read.table("nm.stations", header=TRUE, sep="\t", encoding="UTF-8")
stations=unique(historic.mean.monthly.rain$station)

# mean daily volume
historic.daily = historic[historic$precip>0.5,]
historic.daily.mean=aggregate(log(historic.daily$precip),
by=list(station=historic.daily$station), mean)
historic.daily.mean$logsd=aggregate(log(historic.daily$precip),
by=list(station=historic.daily$station), sd)$x
historic.daily.mean$n=aggregate(historic.daily$precip,
by=list(station=historic.daily$station), length)$x
historic.daily.mean$lc=exp(historic.daily.mean$x-1.96*historic.daily.mean$logsd)
historic.daily.mean$uc=exp(historic.daily.mean$x+1.96*historic.daily.mean$logsd)
#write.table(historic.daily.mean, "nm complete years, daily stats.txt", quote=FALSE, sep="\t",
row.names=FALSE)

##### show graphs
if(show.graphs) {
      for(i in 1:length(stations.names$X.ID)) {
```

```
	quartz(width=4, height=3)
	par(mar=c(5,4,0.2,0.2), mgp=c(2.5, 1, 0), mex=0.8)
	m=historic.mean.monthly.rain$mo[historic.mean.monthly.rain$station==stations.names[i,1]]
	r=historic.mean.monthly.rain$x[historic.mean.monthly.rain$station==stations.names[i,1]]
	plot(m, r, type="l", sub=paste(paste(stations.names[i,2],"\n", sep=""),
paste(stations.names[i,3]/-1000,"°S ", sep=""), paste(stations.names[i,4]/1000, "°E", sep=""),
sep=""), axes=FALSE, xlab="", ylab="mean monthly precip. (mm)")
	axis(2)
	axis(1, labels=c("A", "M","J", "J", "A", "S", "O", "N", "D", "J", "F", "M"), at=0:11)
	}
}

##### calculate daily rain probability for each month
	## exclude tiny rain events
	my.d$precip = ifelse(my.d$pr<0.5, 0, my.d$pr)
	# calculate daily probability of rainfall
	dm=aggregate(my.d$precip, by=list(mo=my.d$mo, station=my.d$station),
function(x){length(na.omit(x))})
	dor=aggregate(my.d$precip, by=list(mo=my.d$mo, station=my.d$station),
function(x){sum(sign(na.omit(x)))})
	dor$p=dor$x/dm$x

##### determine distribution of rainfall volume
library('MASS')
	##### I assume an exponential distribution based on visual inspection
	dor$rate=rep(NA,length(dor$x))
	dor$maxObs=rep(NA,length(dor$x))
	for(i in 1:length(dm$mo)) {
			t.d=my.d[my.d$station==dm$station[i] & my.d$mo==dm$mo[i] & my.d$precip>0.0 &
my.d$precip<150.0,]
			if(length(t.d$precip)>5) {
				lnfit=fitdistr(t.d$precip, "exponential", lower=0.5, upper=150)
				dor$rate[i]=coef(lnfit)[1]
				dor$maxObs[i] = max(na.omit(t.d$precip))
		} else { if (length(t.d$precip)>0) {
				dor$rate[i]=1/9.42 # frequency analysis of any month with < 6 rainfall
events in JMP
					dor$maxObs[i] = 105 # based on two occasions with daily rain >
100, the 'normal max' seems to be ≈30
				} else	{dor$rate[i] = 0
					dor$maxObs[i] = 0
				}
	}
} # end for


# save data for each station for input into a C++ program
for (s in 1:length(stations)) {
	write.table(dor[dor$station==stations[s],c(2,1,4,5,6)], paste(stations[s],"e2.txt",
sep=""), sep="\t", quote=FALSE, row.names=FALSE)
	}

###### validation
old.par=par()
par(mar=c(5,4,0.2,0.2), mgp=c(6,1,0))
	boxplot(historic.annual.rain$x~historic.annual.rain$station, outline=FALSE, las=2,
at=seq(0,8*2-1,2), xlim=c(0, 16), xlab="station ID", ylab="annual precipitation")

	d.stats=data.frame(station=rep(NA,8), logmean=NA, logSD=NA, N=NA)
	vd=read.delim("validation/680140e2_0_rain.txt", header=FALSE)
	vd.y=apply(vd,1,sum)
	boxplot(vd.y, add=TRUE, at=1, col="red", axes=FALSE, outline=FALSE)
	vd.d=as.vector(as.matrix(vd, ncol=1))
	vd.d=vd.d[vd.d>0.5]
	d.stats[1,]$station=680140; d.stats[1,]$N=length(vd.d)
	d.stats[1,]$logmean=mean(log(vd.d)); d.stats[1,]$logSD=sd(log(vd.d))
```

```
      vd=read.delim("validation/680180e2_0_rain.txt", header=FALSE)
      vd.y=apply(vd,1,sum)
      boxplot(vd.y, add=TRUE, at=3, col="red", axes=FALSE, outline=FALSE)
      vd.d=as.vector(as.matrix(vd, ncol=1))
      vd.d=vd.d[vd.d>0.5]
      d.stats[2,]$station=680180; d.stats[2,]$N=length(vd.d)
      d.stats[2,]$logmean=mean(log(vd.d)); d.stats[2,]$logSD=sd(log(vd.d))

      vd=read.delim("validation/681040e2_0_rain.txt", header=FALSE)
      vd.y=apply(vd,1,sum)
      boxplot(vd.y, add=TRUE, at=5, col="red", axes=FALSE, outline=FALSE)
      vd.d=as.vector(as.matrix(vd, ncol=1))
      vd.d=vd.d[vd.d>0.5]
      d.stats[3,]$station=681040; d.stats[3,]$N=length(vd.d)
      d.stats[3,]$logmean=mean(log(vd.d)); d.stats[3,]$logSD=sd(log(vd.d))

      vd=read.delim("validation/681100e2_0_rain.txt", header=FALSE)
      vd.y=apply(vd,1,sum)
      boxplot(vd.y, add=TRUE, at=7, col="red", axes=FALSE, outline=FALSE)
      vd.d=as.vector(as.matrix(vd, ncol=1))
      vd.d=vd.d[vd.d>0.5]
      d.stats[4,]$station=681100; d.stats[4,]$N=length(vd.d)
      d.stats[4,]$logmean=mean(log(vd.d)); d.stats[4,]$logSD=sd(log(vd.d))

      vd=read.delim("validation/681120e2_0_rain.txt", header=FALSE)
      vd.y=apply(vd,1,sum)
      boxplot(vd.y, add=TRUE, at=9, col="red", axes=FALSE, outline=FALSE)
      vd.d=as.vector(as.matrix(vd, ncol=1))
      vd.d=vd.d[vd.d>0.5]
      d.stats[5,]$station=681120; d.stats[5,]$N=length(vd.d)
      d.stats[5,]$logmean=mean(log(vd.d)); d.stats[5,]$logSD=sd(log(vd.d))

      vd=read.delim("validation/681160e2_0_rain.txt", header=FALSE)
      vd.y=apply(vd,1,sum)
      boxplot(vd.y, add=TRUE, at=11, col="red", axes=FALSE, outline=FALSE)
      vd.d=as.vector(as.matrix(vd, ncol=1))
      vd.d=vd.d[vd.d>0.5]
      d.stats[6,]$station=681160; d.stats[6,]$N=length(vd.d)
      d.stats[6,]$logmean=mean(log(vd.d)); d.stats[6,]$logSD=sd(log(vd.d))

      vd=read.delim("validation/683000e2_0_rain.txt", header=FALSE)
      vd.y=apply(vd,1,sum)
      boxplot(vd.y, add=TRUE, at=13, col="red", axes=FALSE, outline=FALSE)
      vd.d=as.vector(as.matrix(vd, ncol=1))
      vd.d=vd.d[vd.d>0.5]
      d.stats[7,]$station=683000; d.stats[7,]$N=length(vd.d)
      d.stats[7,]$logmean=mean(log(vd.d)); d.stats[7,]$logSD=sd(log(vd.d))

      vd=read.delim("validation/683120e2_0_rain.txt", header=FALSE)
      vd.y=apply(vd,1,sum)
      boxplot(vd.y, add=TRUE, at=15, col="red", axes=FALSE, outline=FALSE)
      vd.d=as.vector(as.matrix(vd, ncol=1))
      vd.d=vd.d[vd.d>0.5]
      d.stats[8,]$station=683120; d.stats[8,]$N=length(vd.d)
      d.stats[8,]$logmean=mean(log(vd.d)); d.stats[8,]$logSD=sd(log(vd.d))

quartz.save("validation exponential2 model.png")

### comparison of daily values ###
d.stats$lc = exp(d.stats$logmean-1.96*d.stats$logSD)
d.stats$uc = exp(d.stats$logmean+1.96*d.stats$logSD)


par(mar=c(3.8,3.8,0.2,0.2), mgp=c(2.3,1,0), mex=0.8)
```

```
plot(0:16, 0:16*6, type="n", xlab="", ylab="daily precip(mm)", axes=FALSE)
axis(1, at=0:7*2+1); axis(2); box()
points(0.5+0:7*2, exp(d.stats$logmean), pch=3)
arrows(0.5+0:7*2, d.stats$lc, 0.5+0:7*2, d.stats$uc, angle=90, length=0.07/2.54, code=3)
points(0:7*2+1.5, exp(historic.daily.mean$x[-c(1,4,10)]), pch=3, col="red")
arrows(0:7*2+1.5, historic.daily.mean$lc[-c(1,4,10)], 0:7*2+1.5, historic.daily.mean$uc[-
c(1,4,6,10)], angle=90, length=0.07/2.54, code=3, col="red")
```

Appendix 2: C++ source code for the time series generator NamRain, version 1.0.7

```
#include <iostream>
#include <fstream>
#include <cmath>
#include <ctime> // for srand
#include <string>
#include <sstream>

float z01(void) {return rand()/(float(RAND_MAX)+1);} // we use the built in random number generator
                                                                  //use a better one if you like
const int d_in_yr = 365; // days in a year

void generateRain(char* station, int maxyears, int copies, float* pRain_mm) {
     const int years = maxyears; // the length of time series selected by the user

     /* read the parameters from a file into a string */
     std::ifstream infile; // the variable associated with the input file
     std::string FileName; // the name of the input file
     FileName = station; // copy the file name to a temporary variable
     FileName += ".txt";  // copy the file name to a temporary variable
     infile.open(FileName.c_str() , std::ios::binary); // open the file, file name converted to C-
string
     if(!infile)    std::cout << "File '" << FileName << "' not found. No file opened!\n";
     else {

          float skip; // a variable for storing redundant data
          float ProbRain [12]; // an array for the probability of rain on a day for each month
          float rate [12]; // the rate (1/mean) of the exponential distribution
                                 //for the daily rain amount for each month
          float maxObs [12]; // the maximum amount of rain observed for each month, used to prevent
too high values

          infile.ignore(100, '\n'); // skip header

          for (int m=0; m<12; m++) {
               infile >> skip;
               infile >> skip;
               infile >> ProbRain[m]; // probability of rain
               infile >> rate[m]; // 1/mean
               infile >> maxObs[m]; // max amount
          }

          infile.close(); // close the input file


          /* ****** determine the stochastic rain series ******** */
          int d = 0; // variable counting days of year, defined outside day loop for use by annual
summary
          float arain = 0.0; // variable summing the annual rain amount
          std::stringstream outFileName; // the name of the output file
          outFileName << station << "_" << copies << "_rain.txt"; // construct the output file name
          std::cout << "--------------------\nOutput is in file'" << outFileName.str() <<"'.\n";
          std::cout << "--------------------\nannual precipitation summary\n\n";

          for (int y=0; y<years; y++) {
               d = 0; // reset days at start of each year
               arain = 0.0; // reset annual amount at start of each year
               for (int m=0; m<12; m++) { // for each month
                    for(int md=0; md<31; md++) { // for each day in a month
                                                        // determine end of month
                         switch (m) {
                              case  1: if (md==27){md=31;} break;//Feb
```

```
                                case  3: if (md==29){md=31;} break;
                                case  5: if (md==29){md=31;} break;
                                case  8: if (md==29){md=31;} break;
                                case 10: if (md==29){md=31;} break; //Nov
                           } // end switch

                           // determine whether it's a rainy day
                           float Z = z01();
                           if (Z<ProbRain[m]) {
                                pRain_mm[y*d_in_yr+d] = 1;}
                           else {
                                pRain_mm[y*d_in_yr+d] = 0;}

                           // determine volume on a rainy day from an exponential distribution
                           // daily amount is limited by the maximum amount observed in the records
                           if (pRain_mm[y*d_in_yr+d]==1) {
                                Z = z01();
                                pRain_mm[y*d_in_yr+d] = std::min(maxObs[m],float(-log(Z))/rate[m]);
                           }
                           arain += pRain_mm[y*d_in_yr+d]; // sum up for annual amount
                           d++; // keep track of day of year
                      } // end for day in month
                 } // end for month

                 std::cout << y << "\t" << arain << std::endl;

            } // end for year

            std::ofstream outfile; // variable associated with output file
            outfile.open(outFileName.str().c_str(), std::ios::binary); // open output file

            /* ****** write time series to output file *********
                 this algorithm puts one year on one line,
                 so you get a year x day matrix
                 */
            for (int yy = 0; yy<years; yy++) {
                 for (int dd = 0; dd<(d_in_yr-1); dd++) {
                      outfile << pRain_mm[yy*d_in_yr+dd] << "\t";
                 }
                 outfile << pRain_mm[yy*years+(d_in_yr-1)] << "\n";
            }

            /* ****** write time series to output file *********
alternative: this algorithm puts the data in one column
                 */
            /*
             for (int yy = 0; yy<years; yy++) {
                  for (int dd = 0; dd<d_in_yr; dd++) {
                       outfile << pRain_mm[yy*d_in_yr+dd] << "\n";
                  }
             }
             */
            outfile.close(); // close ouput file
       };
}

int stationID = 0; // internal ID of a station
char pStation[7]; // the code of the station in the original climate data files
                    // downloaded from the World Climate Data centeer
int copies = 0; // number of time series requested by the user
int years = 0; // length of each time series in years

int main (int argc, char * const argv[]) {

     /*** the command line input ****/
    std::cout << "Select station:\n";
```

```
     std::cout<<"1. GROOTFONTEIN\n";
     std::cout<<"2. RUNDU\n";
     std::cout<<"3. WALVISBAY(PELICAN)\n";
     std::cout<<"4. WINDHOEK\n";
     std::cout<<"5. HOSEA KUTAKO\n";
     std::cout<<"6. GOBABIS\n";
     std::cout<<"7. LUDERITZ\n";
     std::cout<<"8. KEETMANSHOOP\n";
     std::cin >> stationID;

     std::cout << "Enter number of years in each time series:\n";
     std::cin >> years;

     switch(stationID) {
          case 1: strcpy(pStation, "680140"); break;
          case 2: strcpy(pStation, "680180"); break;
          case 3: strcpy(pStation, "681040"); break;
          case 4: strcpy(pStation, "681100"); break;
          case 5: strcpy(pStation, "681120"); break;
          case 6: strcpy(pStation, "681160"); break;
          case 7: strcpy(pStation, "683000"); break;
          case 8: strcpy(pStation, "683120"); break;
     }

     std::cout << "Enter number of time series (=files) you want to have:\n";
     std::cin >> copies;

     srand(std::time(NULL)); // initializer and seed for random number generator
                                // if you use a different RNG, replace this with the
                                // appropriate initializer


     // finally: the actual algorithm
     // this and the function definition are the parts you need
     // if you want to use generateRain in your own code
     float rain_mm[years*d_in_yr]; // the array for storing the time series

     for (int c=0; c<copies; c++)
          generateRain(pStation, years, c, rain_mm);
     // --------------
    return 0;
}
```