MultiHazard Exemplar

Robert Jane

January 1, 2020

1. Introduction

The `MultiHazard` package provides tools for stationary multivariate statistical modeling. For instance to model the joint distribution of co-occurring hazards. The package contains functions for pre-processing data including imputing missing values, detrending and declustering time series (Section 2) as well as analyzing pairwise correlations over a range of lags (Section 3). Functionality is also built in to conditionally sample a bivariate dataset (given one of the variables is above a predetermined threshold) and select the best fitting amongst an array of parametric (extreme and non-extreme, truncated and non-truncated) marginal distributions or copulas (Section 4). The latter allow the dependence structure between the set of variables to be modelled independenly of marginal distributions. Estimation of joint probability contours using the method of overlaying (conditional) contours given in Bender et al. (2016) and subsequently for a given return period extracting design events assuming full dependence, as well as the "most likely" or an ensemble of possible design events once accounting for dependence is possible. The package also provides the capability of fitting and simulating synthetic records from three higher dimensional approaches - standard (elliptic/Archimedean) copulas, Pair Copula Constructions (PCCs) and the conditional threshold exceedance approach of Heffernan and Tawn (2004) (Section 5). The package provides the code and data used in Jane et al. (2020), consequently applications in this vignette center around assessing the potential for compound flooding in South Florida.

2. Pre-processing

Imputation

Well G_3356 represents the groundwater level at Site S20, however, it contains missing values. Lets impute missing values in the record at Well G_3356 using recordings at nearby Well G_3355. Firstly, lets ensure the columns of the dataframes containing the two time series are of the necessary class.

```
#Viewing the first few rows of the two time series
head(G_3356)
```

```
##         Date Value
## 1 1985-10-23  2.46
## 2 1985-10-24  2.47
## 3 1985-10-25  2.41
## 4 1985-10-26  2.37
## 5 1985-10-27  2.63
## 6 1985-10-28  2.54
```

```
head(G_3355)
```

```
##         Date Value
## 1 1985-08-20  2.53
## 2 1985-08-21  2.50
## 3 1985-08-22  2.46
## 4 1985-08-23  2.43
## 5 1985-08-24  2.40
## 6 1985-08-25  2.37
```

```
#Converting Date column to "Date"" object
G_3356$Date<-seq(as.Date("1985-10-23"), as.Date("2019-05-29"), by="day")
G_3355$Date<-seq(as.Date("1985-08-20"), as.Date("2019-06-02"), by="day")
```

```
#Converting column containing the readings to a "numeric"" object
G_3356$Value<-as.numeric(as.character(G_3356$Value))
G_3355$Value<-as.numeric(as.character(G_3355$Value))
```

Warning message confirms there are NAs in the record at Well G_3356. Before carrying out the imputation the two data frames need to be merged.

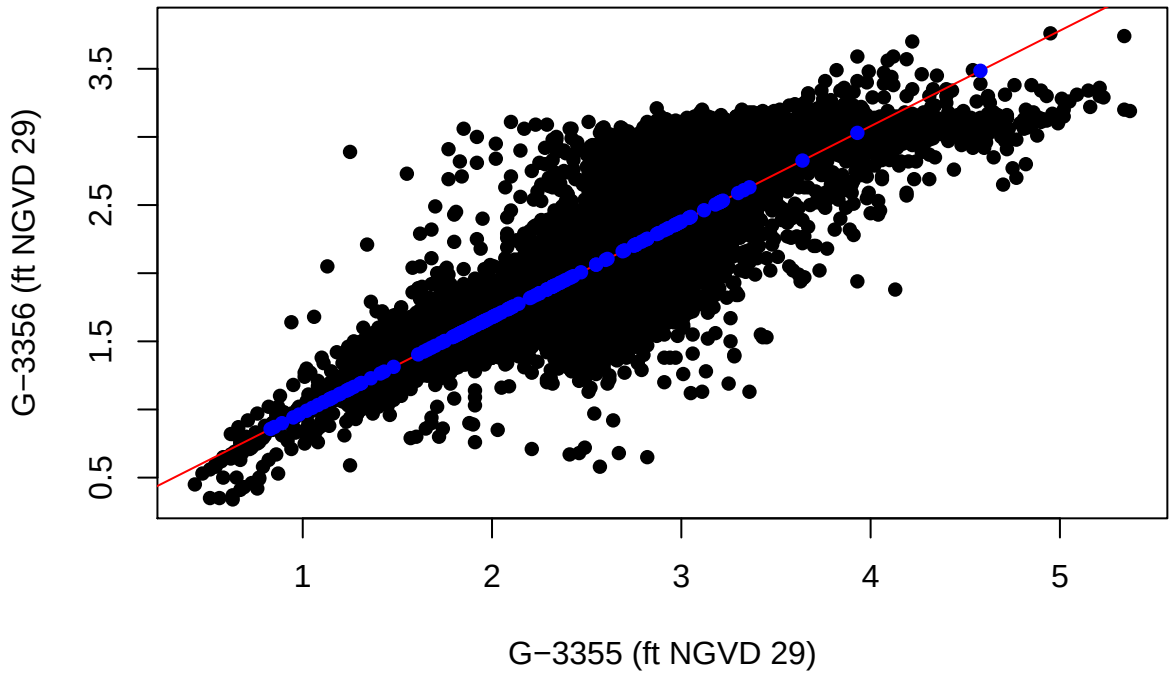
```
#Merge the two dataframes by date
library("dplyr")
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
GW_S20<-left_join(G_3356,G_3355,by="Date")
colnames(GW_S20)<-c("Date","G3356","G3355")
#Carrying out imputation
Imp<-Imputation(Data=GW_S20,Variable="G3356",x_lab="G-3355 (ft NGVD 29)",
                y_lab="G-3356 (ft NGVD 29)")
```



The completed record is given in the `ValuesFilled` column of the data frame outputted as the `Data` object while the linear regression model including its coefficient of determinant are given by the `model` output

argument.

```
head(Imp$Data)
```

```
##         Date G3356 G3355 ValuesFilled
## 1 1985-10-23  2.46  2.87         2.46
## 2 1985-10-24  2.47  2.85         2.47
## 3 1985-10-25  2.41  2.82         2.41
## 4 1985-10-26  2.37  2.79         2.37
## 5 1985-10-27  2.63  2.96         2.63
## 6 1985-10-28  2.54  2.96         2.54
```

```
Imp$Model
```

```
##
## Call:
## lm(formula = data[, variable] ~ data[, Other.variable])
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.60190 -0.16654  0.00525  0.16858  1.73824
##
## Coefficients:
##                         Estimate Std. Error t value Pr(>|t|)
## (Intercept)             0.275858   0.012366   22.31   <2e-16 ***
## data[, Other.variable] 0.700724   0.004459  157.15   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2995 on 11825 degrees of freedom
##   (445 observations deleted due to missingness)
## Multiple R-squared:  0.6762, Adjusted R-squared:  0.6762
## F-statistic: 2.47e+04 on 1 and 11825 DF,  p-value: < 2.2e-16
```

Are any values still NAs?

```
G_3356_ValueFilled_NA<-which(is.na(Imp$Data$ValuesFilled)==TRUE)
length(G_3356_ValueFilled_NA)
```

```
## [1] 3
```

Linear interpolating the three remaining NAs.

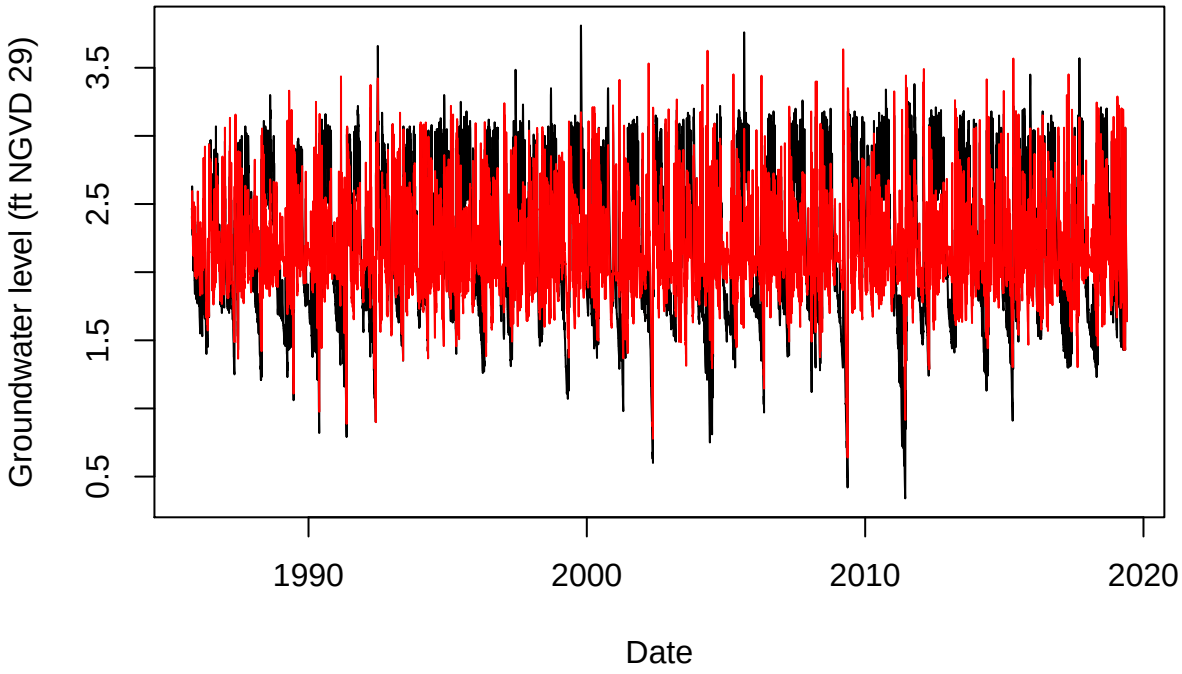
```
G3356_approx<-approx(seq(1,length(Imp$Data$ValuesFilled),1),Imp$Data$ValuesFilled,
                     xout=seq(1,length(Imp$Data$ValuesFilled),1))
Imp$Data$ValuesFilled[which(is.na(Imp$Data$ValuesFilled)==TRUE)]<-
  G3356_approx$y[which(is.na(Imp$Data$ValuesFilled)==TRUE)]
```

Detrending

In the analysis completed O-sWL (Ocean-side Water Level) and groundwater level series are subsequently detrended. The `Detrend()` function uses either a linear fit covering the entire data (`Method=linear`) or moving averge window (`Method=window`) of a specified length (`Window_Width`) to remove trends from a time series. The residuals are added to the final `End_Length` observations. The default Detrend() parameters specify a moving average (`Method=window`) three month window (`Window_Width=89`), to remove any seasonality from the time series. The deafult is then to add the residuals to the average of the final five years of observations (`End_Length=1826`) to bring the record to the present day level, accounting for the Perigian tide in the case of O-sWL. The mean of the observations over the first three months were subtracted from the values during

this period before the present day (5 year) average was added. The following R code detrends the record at Well G_3356. Note the function requires a Date object and the completed series.

```
#Cresaring a data from with the imputed series alongside the corresponding dates
G_3356_Imp<-data.frame(Imp$Data$Date,Imp$Data$ValuesFilled)
colnames(G_3356_Imp)<-c("Date","ValuesFilled")
#Detrending
G_3356_Detrend<-Detrend(Data=G_3356_Imp,PLOT=TRUE,x_lab="Date",
                        y_lab="Groundwater level (ft NGVD 29)")
```



Output of the`Detrend()` function is simply the detrended time series.

```
head(G_3356_Detrend)
```

```
## [1] 2.394700 2.411588 2.360033 2.327588 2.595588 2.520255
```

Creating a data frame containing the detrended groundwater series at site S20 i.e. G_3356_Detrend and their corresponding dates

```
S20.Groundwater.Detrend.df<-data.frame(as.Date(GW_S20$Date),G_3356_Detrend)
colnames(S20.Groundwater.Detrend.df)<-c("Date","Groundwater")
```

Declustering

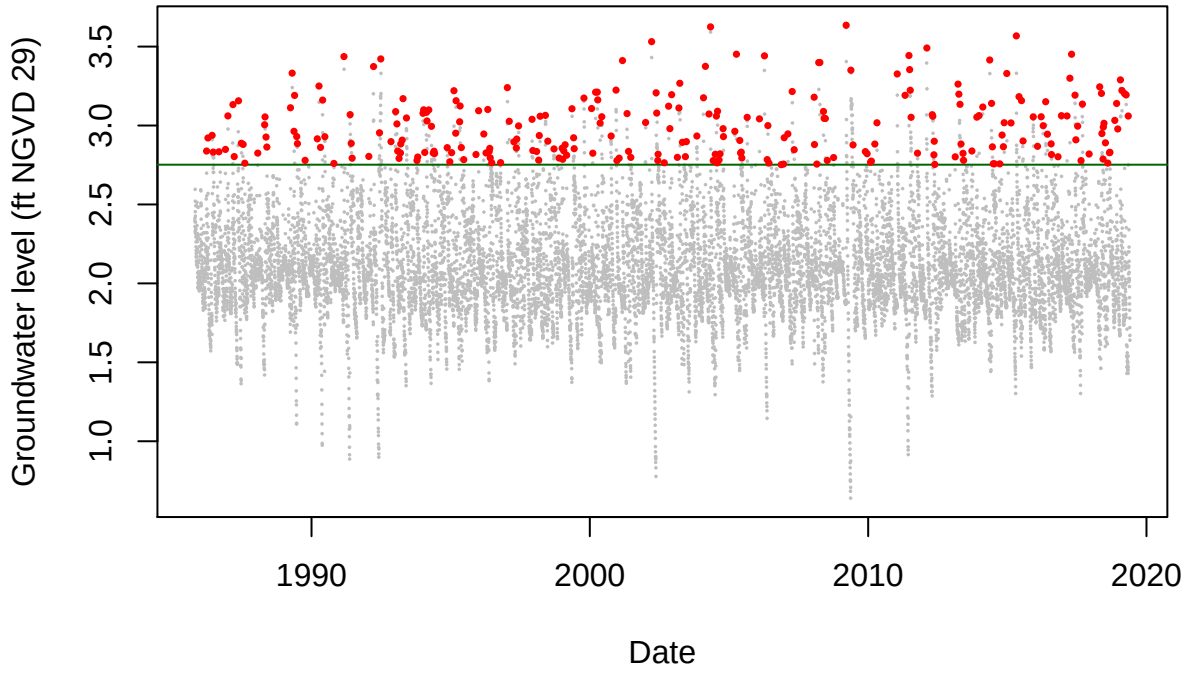
The `Decluster()` function declusters a time series using a threshold `u` specified as a quantile of the completed series and separation criterion `SepCrit` to ensure independent events. If `mu=365.25` then `SepCrit` denotes the minimum number of days readings must remain below the threshold before a new event is defined.

```
G_3356.Declustered<-Decluster(Data=G_3356_Detrend,u=0.95,SepCrit=3,mu=365.25)
```

```
## Warning in x.exceed.max.position[i] <- (x.exceed.lower.bound) +
## which(Data[(x.exceed.lower.bound + : number of items to replace is not a
## multiple of replacement length
```

Plot showing the completed, detrended record at Well G-3356 (grey circles) along with cluster maxima (red circles) identified using a 95% threshold (green line) and three day separation criterion.

```
G_3356_Imp$Detrend<-G_3356_Detrend
plot(as.Date(G_3356_Imp$Date),G_3356_Imp$Detrend,col="Grey",pch=16,
     cex=0.25,xlab="Date",ylab="Groundwater level (ft NGVD 29)")
abline(h=G_3356.Declustered$Threshold,col="Dark Green")
points(as.Date(G_3356_Imp$Date[G_3356.Declustered$EventsMax]),
       G_3356.Declustered$Declustered[G_3356.Declustered$EventsMax],
       col="Red",pch=16,cex=0.5)
```



Other outputs from the `Decluster()` function include the threshold on the original scale

```
G_3356.Declustered$Threshold
```

```
## [1] 2.751744
```

and the number of events per year

```
G_3356.Declustered$EventsPerYear
```

```
## [1] 7.857399
```

In preparation for later work, lets assign the detrended and declustered groundwater series at site S20 a name.

```
S20.Groundwater.Detrend.Declustered<-G_3356.Declustered$Declustered
```

Reading in the other rainfall and O-sWL series at Site S20

```
S20.Rainfall.df<-Perrine_df
S20.Rainfall.df$Date<-as.Date(S20.Rainfall.df$Date)
S20.Rainfall.df$Na.Removed<-S20.Rainfall.df$Value
S20.Rainfall.df$Na.Removed[which(is.na(S20.Rainfall.df$Value)==TRUE)]<-0
S20.OsWL.df<-S20_T_MAX_Daily_Completed_Detrend_Declustered[,c(2,4)]
S20.OsWL.df$Date<-as.Date(S20.OsWL.df$Date)
```

Detrending and declustering the rainfall and O-sWL series at Site S20

```
S20.OsWL.Detrend<-Detrend(Data=S20.OsWL.df,Method = "window",PLOT=FALSE,
                          x_lab="Date",y_lab="O-sWL (ft NGVD 29)")
```

Creating a dataframe with the date alongside the detrended OsWL series

```
S20.OsWL.Detrend.df<-data.frame(as.Date(S20.OsWL.df$Date),S20.OsWL.Detrend)
colnames(S20.OsWL.Detrend.df)<-c("Date","OsWL")
```

Declustering rainfall and O-sWL series at site S20,

```
#Setting missing values to zero
S20.Rainfall.Declustered<-Decluster(Data=S20.Rainfall.df$Na.Removed,
                                    u=0.95,SepCrit=3)$Declustered
S20.Rainfall.Declustered[which(is.na(S20.Rainfall.df$Value)==TRUE)]<-NA
#S20.O-sWL does not contain any missing values and so can be declustered without
S20.OsWL.Detrend.Declustered<-Decluster(Data=S20.OsWL.Detrend,
                                        u=0.95,SepCrit=3,mu=365.25)$Declustered
```

Creating data frames with the date alongside declustered series

```
S20.OsWL.Detrend.Declustered.df<-data.frame(S20.OsWL.df$Date,S20.OsWL.Detrend.Declustered)
colnames(S20.OsWL.Detrend.Declustered.df)<-c("Date","OsWL")
S20.Rainfall.Declustered.df<-data.frame(S20.Rainfall.df$Date,S20.Rainfall.Declustered)
colnames(S20.Rainfall.Declustered.df)<-c("Date","Rainfall")
S20.Groundwater.Detrend.Declustered.df<-data.frame(G_3356$Date,S20.Groundwater.Detrend.Declustered)
colnames(S20.Groundwater.Detrend.Declustered.df)<-c("Date","OsWL")
```

Use the Dataframe_Combine function to create data frames containing all observations of the original, detrended if necessary, and declustered time series. On dates where not all variables are observed, missing values are assigned NA.

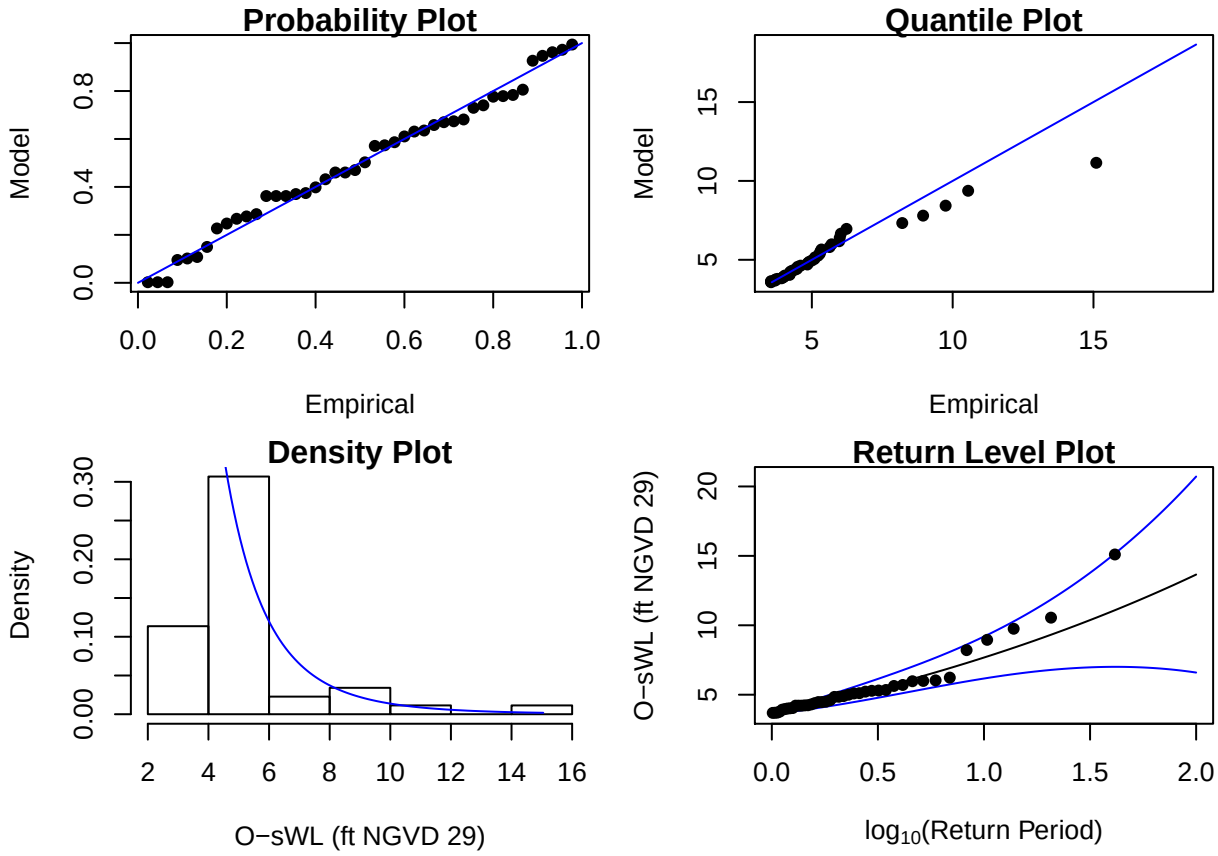
```
S20.Detrend.df<-Dataframe_Combine(data.1<-S20.Rainfall.df,
                                  data.2<-S20.OsWL.Detrend.df,
                                  data.3<-S20.Groundwater.Detrend.df,
                                  names=c("Rainfall","OsWL","Groundwater"))
S20.Detrend.Declustered.df<-Dataframe_Combine(data.1<-S20.Rainfall.Declustered.df,
                                              data.2<-S20.OsWL.Detrend.Declustered.df,
                                              data.3<-S20.Groundwater.Detrend.Declustered.df,
                                              names=c("Rainfall","OsWL","Groundwater"))
```

Fit GPD

The `GPD_Fit()` function fits a generalized Pareto distribution (GPD) to observations above a threshold `u`, specified as a quantile of the completed time series. To fit the distribution the `GPD_Fit()` function requires the declustered series as its Data argument and the entire completed series, detrended if necessary, as its

Data.Full argument. The completed series is required to calcuate the value on the original scale corresponding to u. If PLOT=="TRUE" then diagnostic plots are produced to allow an assessment of the fit.

```
GPD_Fit(Data=S20.Detrend.Declustered.df$Rainfall,Data_Full=na.omit(S20.Detrend.df$Rainfall),
        u=0.997,PLOT="TRUE",xlab_hist="O-sWL (ft NGVD 29)",y_lab="O-sWL (ft NGVD 29)")
```

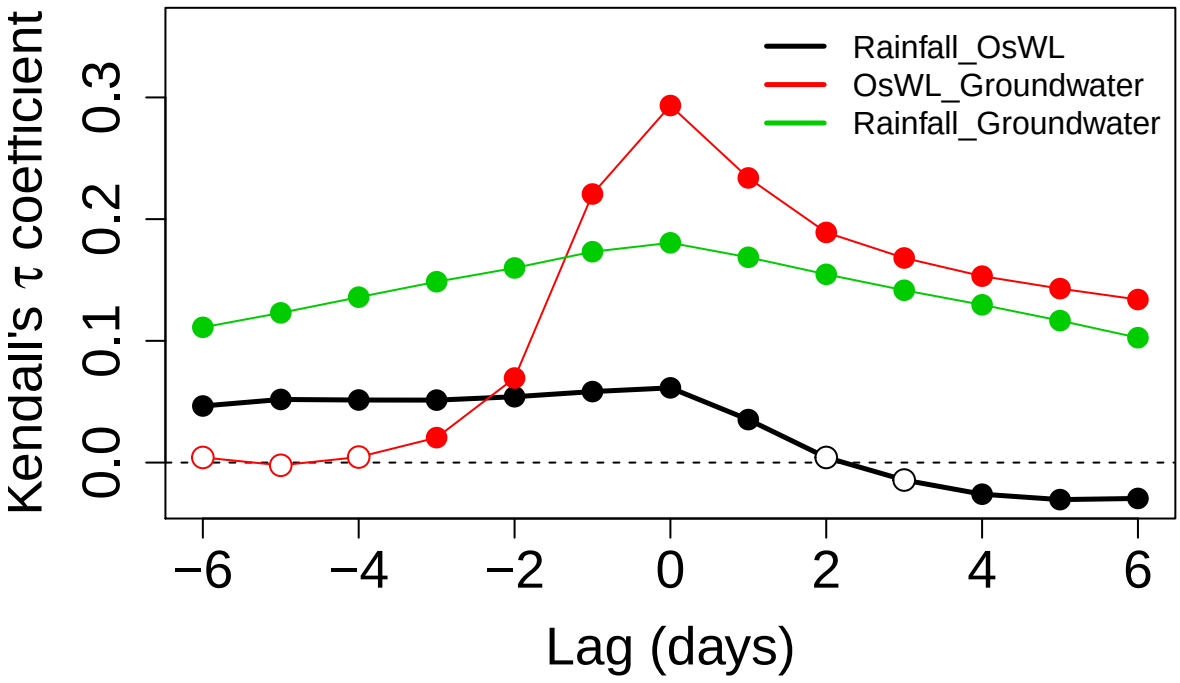



```
## $Threshold
##  99.7%
## 3.5465
##
## $sigma
## [1] 1.421037
##
## $xi
## [1] 0.1696554
##
## $sigma.SE
## [1] 0.2365755
##
## $xi.SE
## [1] 0.1781031
```

3. Correlation analysis

We can use the Kendall_Lag function to view the Kendall's rank correlations coefficient τ between the time seres over a range of lags

```
S20.Kendall.Results<-Kendall_Lag(Data=S20.Detrend.df,GAP=0.2)
```



Lets pull out the Kendall correlation coefficient values between rainfall and O-sWL for lags of $-5, ..., 0, .., 5$ applied to the latter quantity

```
S20.Kendall.Results$Value$Rainfall_OsWL
```

```
##  [1]  0.046483308  0.051860955  0.051392298  0.051311970  0.054097316
##  [6]  0.058316831  0.061388245  0.035305812  0.004206059 -0.014356749
## [11] -0.025993095 -0.030431776 -0.029481162
```

and the corresponding p-values testing the null hypothesis $\tau = 0$

```
S20.Kendall.Results$Test$Rainfall_OsWL_Test
```

```
##  [1] 5.819748e-13 1.014698e-15 8.196547e-16 1.030482e-15 5.160274e-17
##  [6] 1.887733e-19 1.987221e-20 2.232548e-07 4.033739e-01 7.669577e-02
## [11] 1.186248e-03 6.352872e-05 3.864403e-05
```

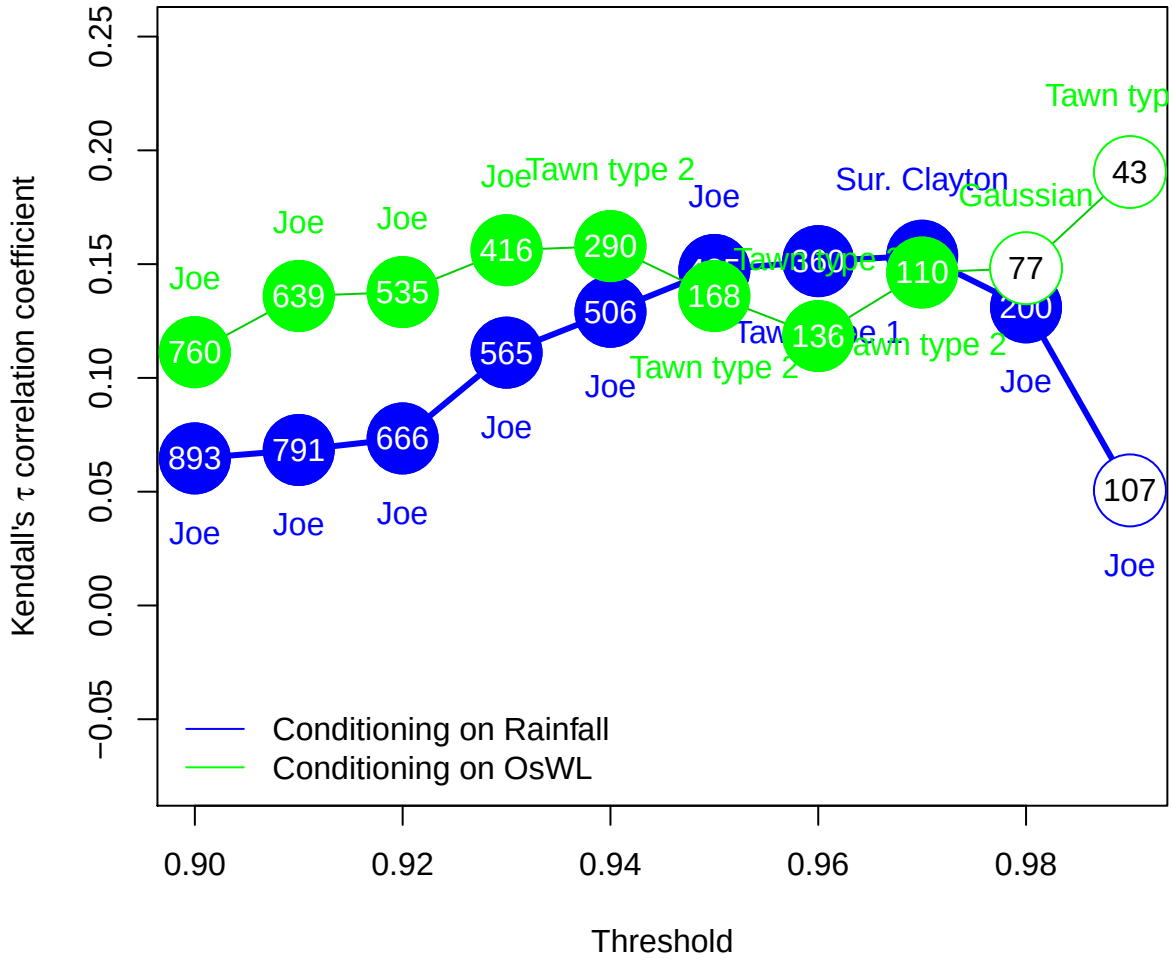
4. Bivariate Analysis

In the report the bivariate analysis considers the two forcings currently accounted for in Flood Protection Level of Service assessments undertaken by SFWMD: rainfall and O-sWL. The bivariate analysis commences with the well-established two-sided conditional sampling approach (approach in Heffernan and Tawn (2004)), where excesses of a conditioning variable are paired with co-occurring values of another variable to create two samples. For each sample the marginals (one extreme, one non-extreme) and joint distribution are then modeled.

The two (conditional) joint distributions are modeled independently of the marginals by using a copula. To al-

low the practitioner to make an informed choice with regards threshold selection, the `Copula_Threshold_2D()` function explores the sensitivity of the best fitting copula, in terms of Akaike Information Criterion (AIC), to the choice of threshold. It undertakes the conditional sampling described above and reports the best fitting bivariate copula. The procedure is carried out for a single or range of thresholds specified by the `Thres` argument and the procedure is automatically repeated with the variables reversed.

```
Copula_Threshold_2D(Data_Detrend=S20.Detrend.df[,-c(1,4)],
Data_Declust=S20.Detrend.Declustered.df[,-c(1,4)],
y_lim_min=-0.075, y_lim_max =0.25,
Upper=c(6,8), Lower=c(6,8),GAP=0.1)
```

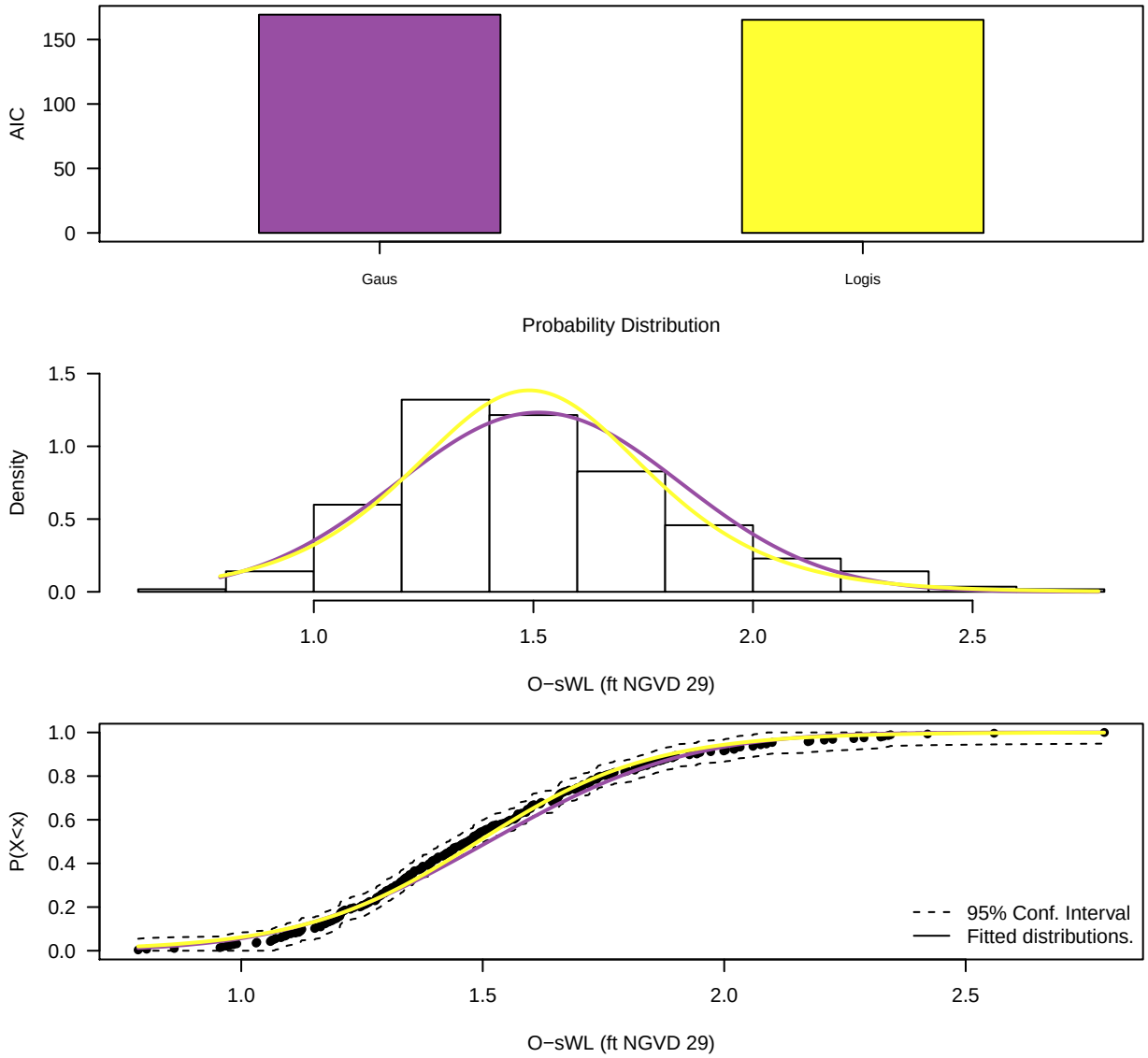



```
## $Kendalls_Tau_Var1
##  [1] 0.06462446 0.06831834 0.07340181 0.11110660 0.12906919 0.14744597
##  [7] 0.15130948 0.15369700 0.13101587 0.05056147
##
## $p_value_Var1
##  [1] 3.953054e-03 4.193644e-03 4.603933e-03 8.693758e-05 1.786936e-05
##  [6] 6.766536e-06 2.045544e-05 1.347405e-04 7.367252e-03 5.236705e-01
```

```
##
## $N_Va1
##  [1] 893 791 666 565 506 425 360 284 200 107
##
## $Copula_Family_Var1
##  [1]   6   6   6   6   6   6 104  13   6   6
##
## $Kendalls_Tau_Var2
##  [1] 0.1113049 0.1359921 0.1377104 0.1561184 0.1579352 0.1359861 0.1183870
##  [8] 0.1463056 0.1482198 0.1904729
##
## $p_value_Var2
##  [1] 3.111295e-05 3.130393e-06 1.201990e-05 1.226532e-05 2.030287e-04
##  [6] 1.218006e-02 4.758068e-02 3.021842e-02 6.691436e-02 7.073874e-02
##
## $N_Var2
##  [1] 760 639 535 416 290 168 136 110  77  43
##
## $Copula_Family_Var2
##  [1]   6   6   6   6 204 204 204 204   1 204
```

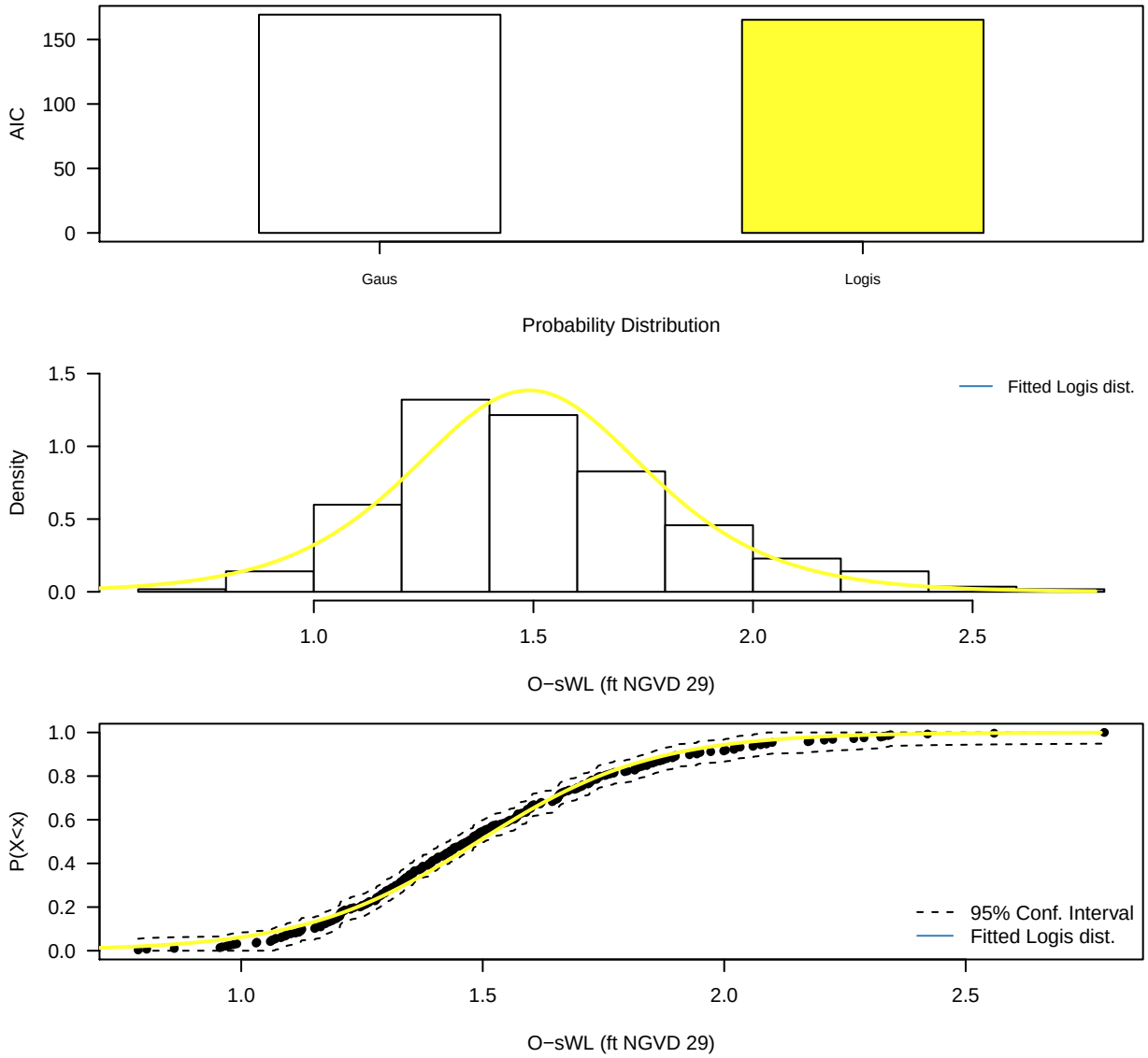
The `Diag_Non_Con()` function is designed to aid in the selection of the appropriate (non-extreme) unbounded marginal distribution for the non-conditioned variable.

```
S20.Rainfall<-Con_Sampling_2D(Data_Detrend=S20.Detrend.df[,-c(1,4)],
Data_Declust=S20.Detrend.Declustered.df[,-c(1,4)],
Con_Variable="Rainfall",Thres=0.97)
Diag_Non_Con(Data=S20.Rainfall$Data$OsWL,x_lab="O-sWL (ft NGVD 29)",
             y_lim_min=0,y_lim_max=1.5)
```



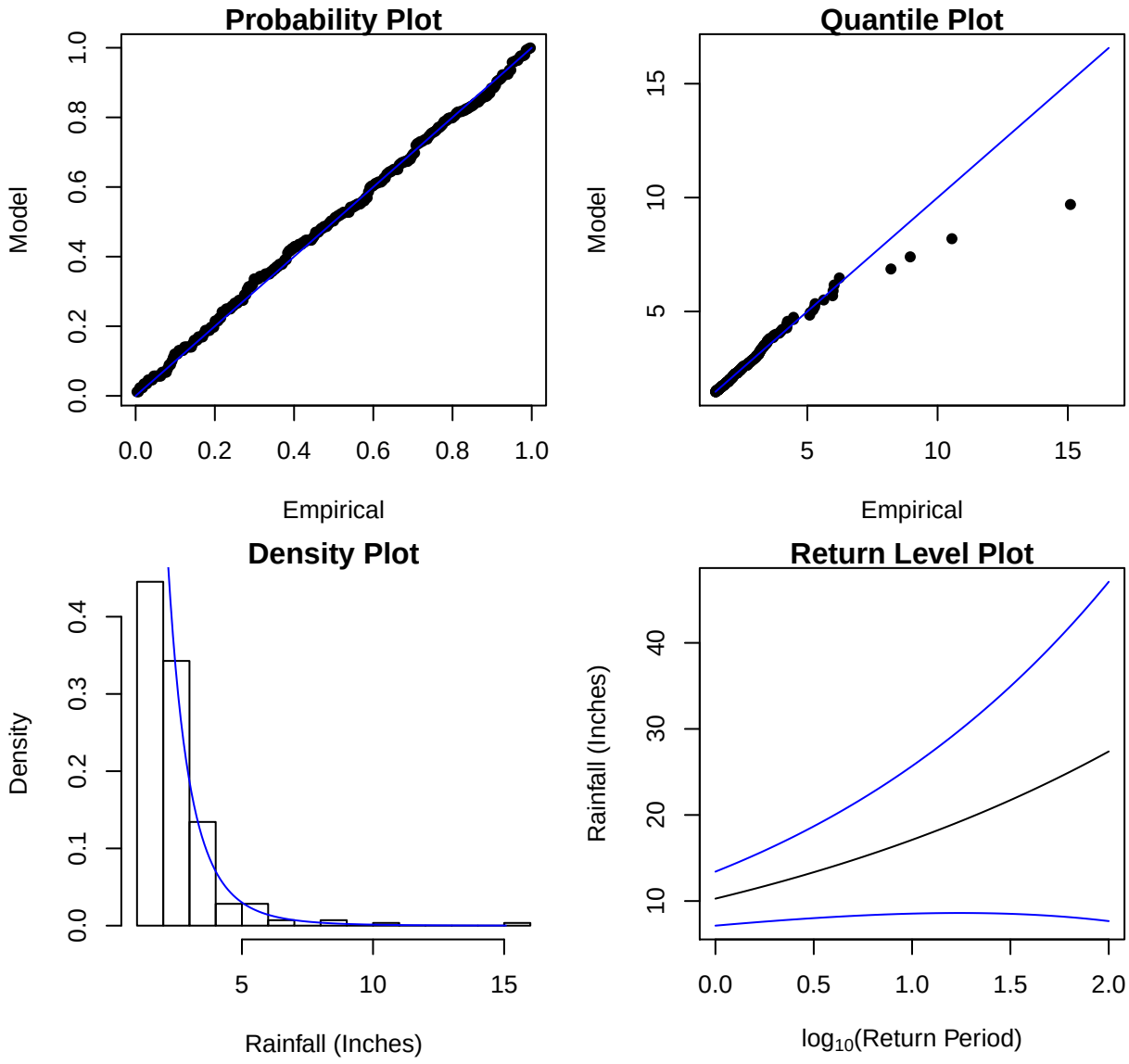
The `Diag_Non_Con_Sel()` function, is similar to the `Diag_Non_Con()` command, but only plots the probability density function and cumulative distribution function of a (single) selected univariate distribution in order to more clearly demonstrate the goodness of fit of a particular distribution. The options are the Gaussian (`Gaus`) and logistic (`Logis`) distributions.

```
Diag_Non_Con_Sel(Data=S20.Rainfall$Data$OsWL,x_lab="O-sWL (ft NGVD 29)",
y_lim_min=0,y_lim_max=1.5,Selected="Logis")
```



A GPD is fitted to the marginal distribution of the conditioning variable i.e. the declustered excesses identified using `Con_Sampling_2D()`. Use `GPD_Fit()` to model the OsWL excesses in the `S22.Rainfall$Data` sample

```
GPD_Fit(Data=S20.Rainfall$Data$Rainfall,Data_Full=S20.Rainfall$Data$Rainfall,u=0,
PLOT="TRUE",xlab_hist="Rainfall (Inches)",y_lab="Rainfall (Inches)")
```




```
## $Threshold
##   0%
## 1.47
##
## $sigma
## [1] 0.8481353
##
## $xi
## [1] 0.176777
##
## $sigma.SE
## [1] 0.08700822
##
## $xi.SE
## [1] 0.06406792
```

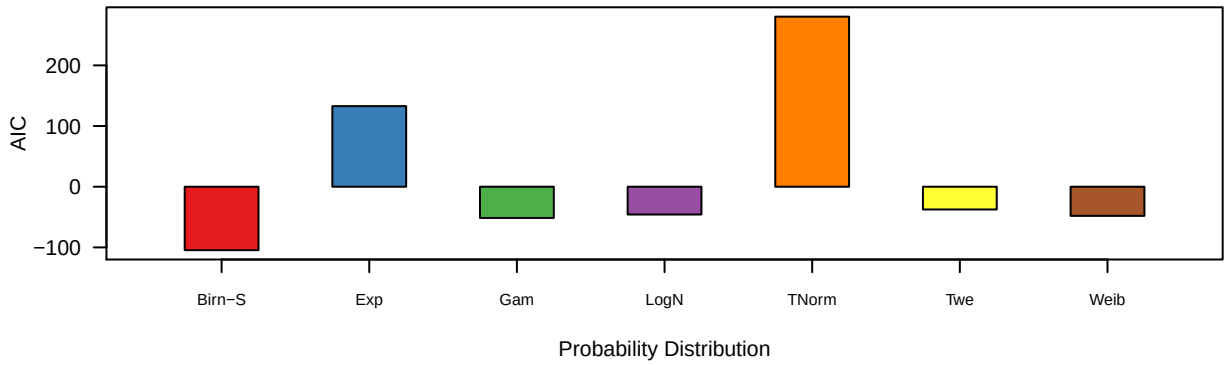
The process of selecting a conditional sample and fitting marginal distributions is repeated but instead conditioning on O-sWL. The non-conditional variable in this case is (total daily) rainfall, which has a

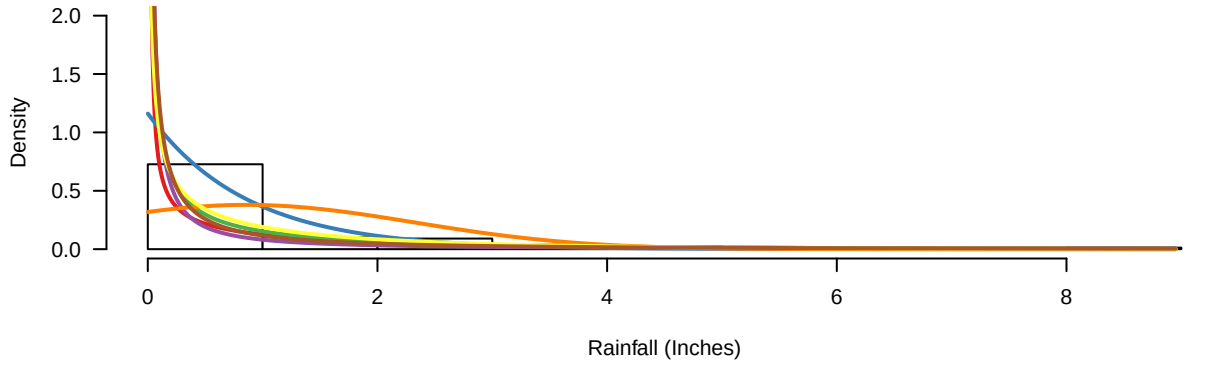
lower bound at zero, and thus requires a suitably truncated distribution. The `Diag_Non_Con_Trunc()` fits a selection of truncated distributions to a vector of data. The `Diag_Non_Con_Trunc_Sel()` function is analogous to the `Diag_Non_Con_Sel()` function, avalibale distributions are the Birnbaum-Saunders (`BS`), exponential (`Exp`), gamma (`Gam`), lognormal (`LogN`), Tweedie (`Twe`) and Weibull (`Weib`).

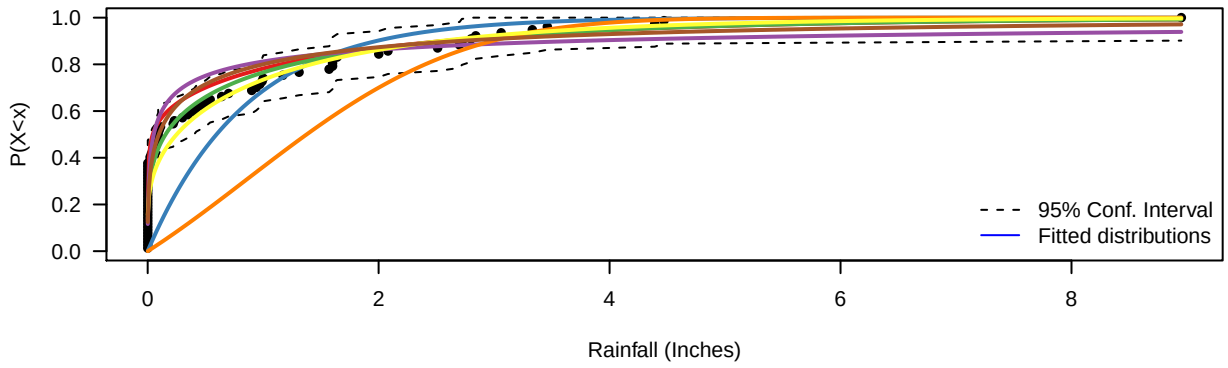
```
S20.OsWL<-Con_Sampling_2D(Data_Detrend=S20.Detrend.df[,-c(1,4)],
Data_Declust=S20.Detrend.Declustered.df[,-c(1,4)],
Con_Variable="OsWL",Thres=0.98)
Diag_Non_Con_Trunc(Data=S20.OsWL$Data$Rainfall+0.001,x_lab="Rainfall (Inches)",
y_lim_min=0,y_lim_max=2)
```

```
## 1.5 1.7 1.9 2.1 2.3 2.5
## ......Done.
```

```
## 1.5 1.7 1.9 2.1 2.3 2.5
## ......Done.
```



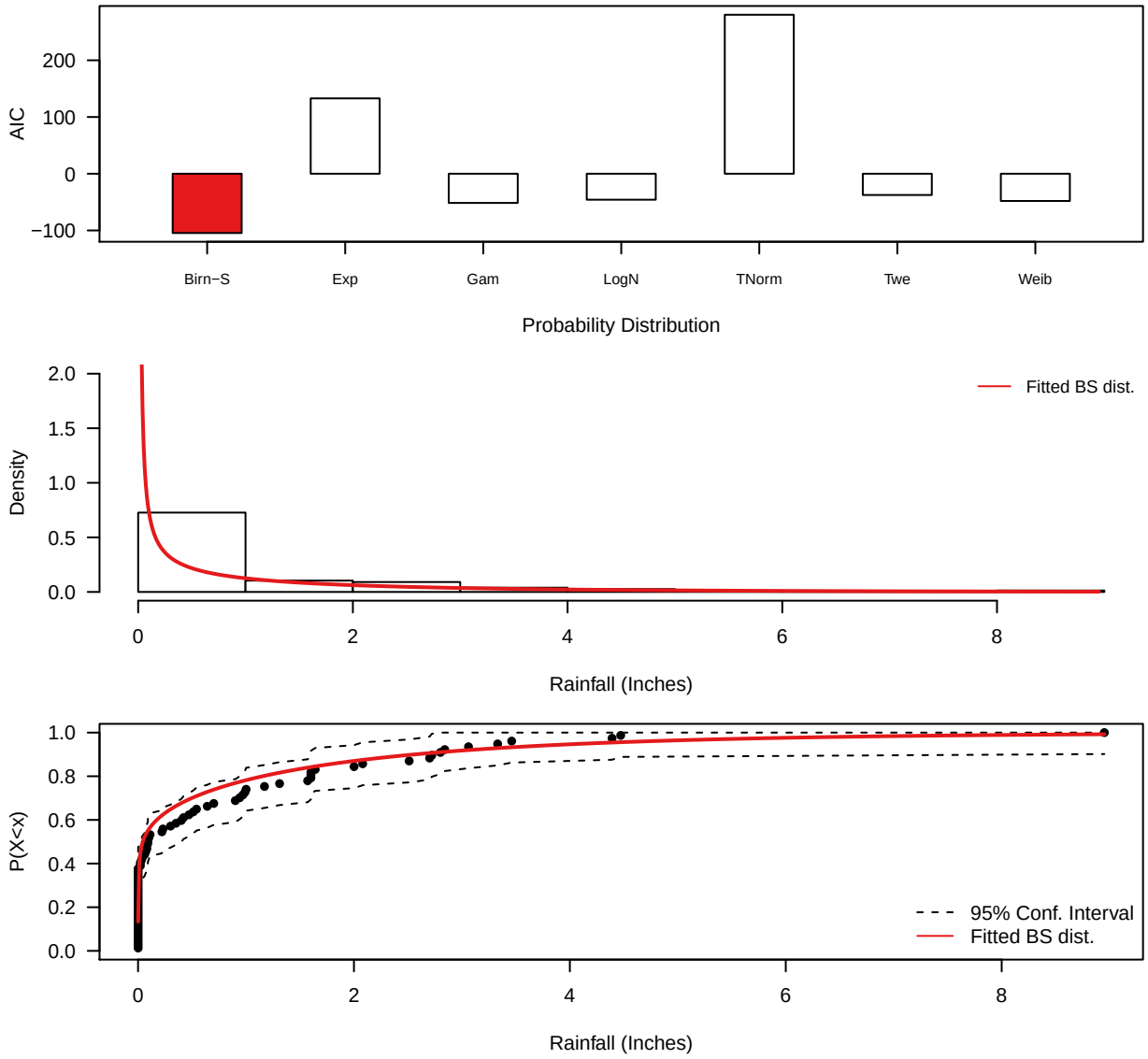





```
## 1.5 1.7 1.9 2.1 2.3 2.5
## ......Done.
```

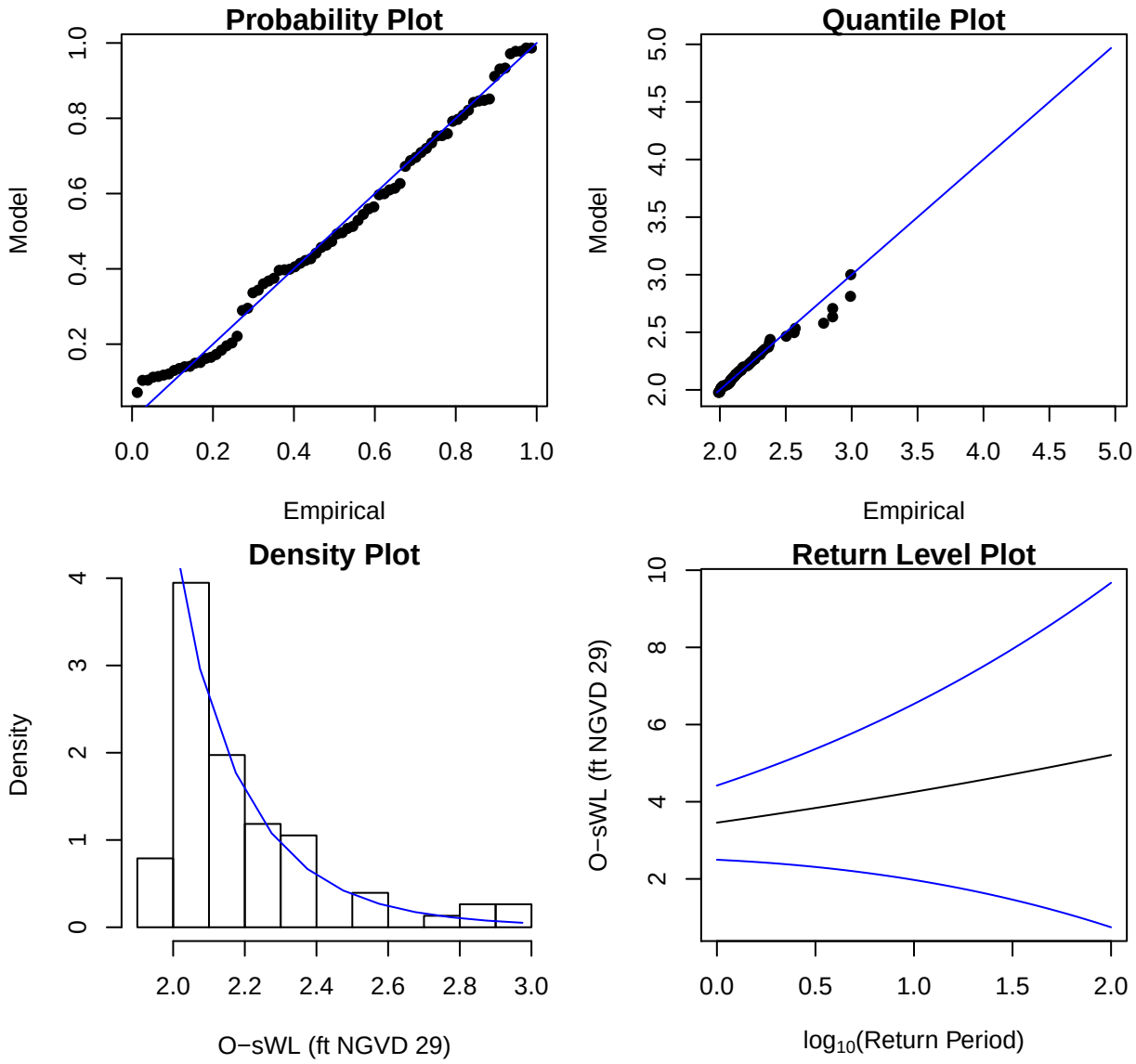
```
Diag_Non_Con_Trunc_Sel(Data=S20.OsWL$Data$Rainfall+0.001,x_lab="Rainfall (Inches)",
y_lim_min=0,y_lim_max=2,Selected="BS")
```

```
## 1.5 1.7 1.9 2.1 2.3 2.5
## ......Done.
```



Using `GPD_Fit()` to assess the fit of the GPD to the OsWL excesses

```
GPD_Fit(Data=S20.OsWL$Data$OsWL,Data_Full=S20.OsWL$Data$OsWL,u=0,
PLOT="TRUE",xlab_hist="O-sWL (ft NGVD 29)",y_lab="O-sWL (ft NGVD 29)")
```

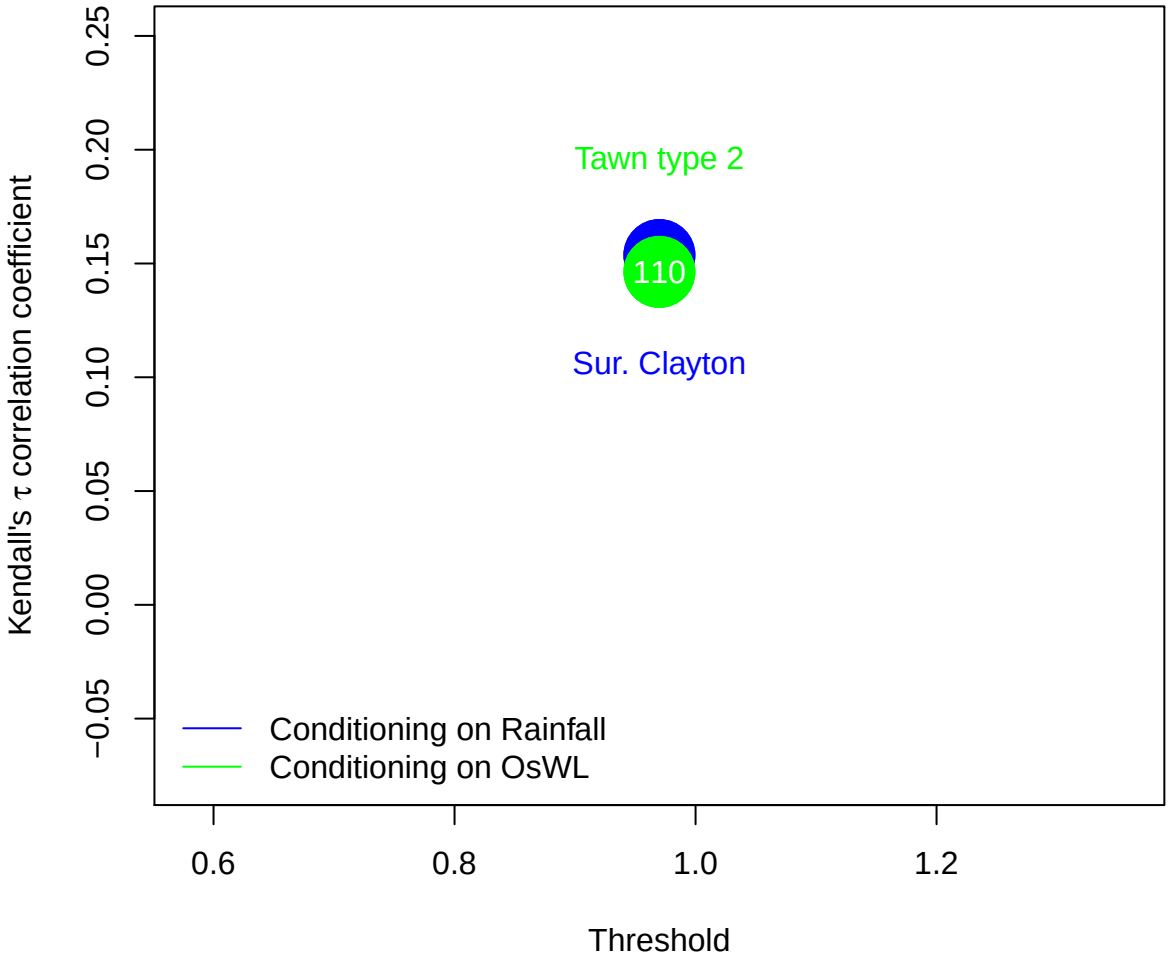



```
## $Threshold
##       0%
## 1.974928
##
## $sigma
## [1] 0.1977586
##
## $xi
## [1] 0.07922455
##
## $sigma.SE
## [1] 0.1803204
##
## $xi.SE
## [1] 0.1381978
```

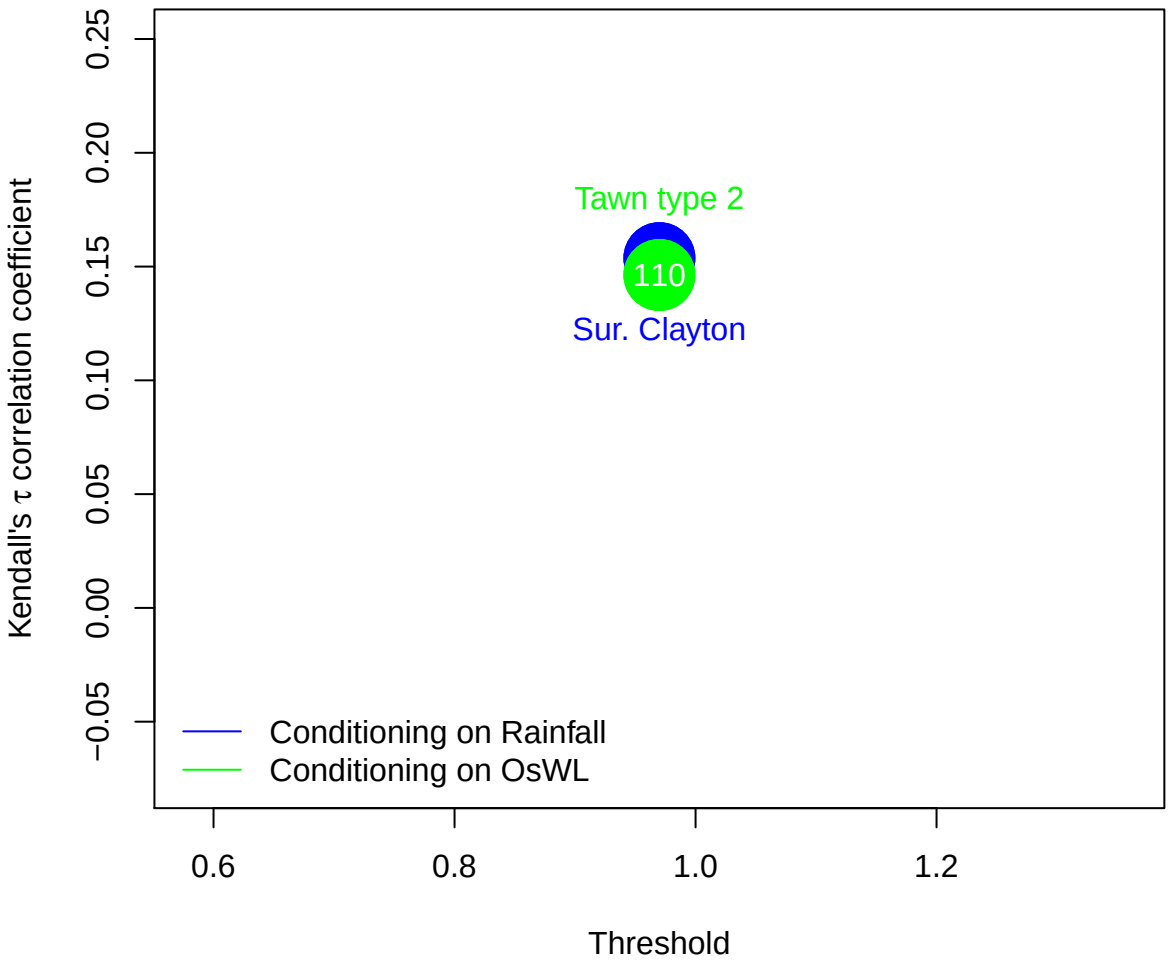
The `Design_Event_2D()` function finds the isoline associated with a particular return period, by overlaying the two corresponding isolines from the joint distributions fitted to the conditional samples using the method

in Bender et al. (2016). `Design_Event_2D()` requires the copulas families chosen to model the dependence structure in the two conditional samples as input.

```
S20.Copula.Rainfall<-Copula_Threshold_2D(Data_Detrend=S20.Detrend.df[,-c(1,4)],
Data_Declust=S20.Detrend.Declustered.df[,-c(1,4)],
Thres =0.97,y_lim_min=-0.075,y_lim_max=0.25,
Upper=c(2,9),Lower=c(2,10),GAP=0.15)$Copula_Family_Var1
```



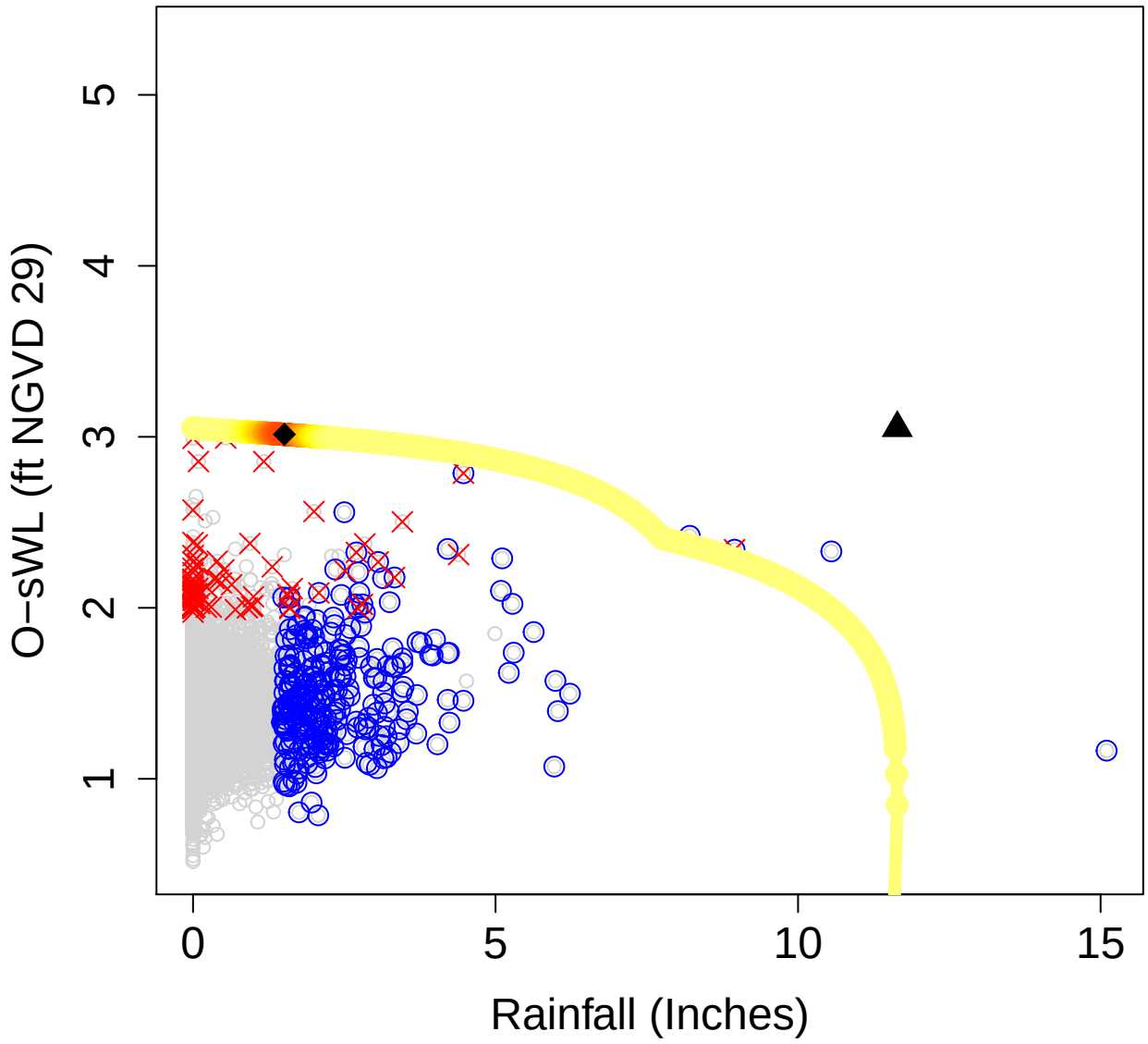

```
S20.Copula.OsWL<-Copula_Threshold_2D(Data_Detrend=S20.Detrend.df[,-c(1,4)],
Data_Declust=S20.Detrend.Declustered.df[,-c(1,4)],
Thres =0.97,y_lim_min=-0.075, y_lim_max =0.25,
Upper=c(6,7,8),Lower=c(6,7,8),GAP=0.1)$Copula_Family_Var2
```



As input the function requires: * Data = Original (detrended) rainfall and O-sWL series * Data_Con1/Data_Con2 = Two conditionally sampled datasets * Thres1/Thres2 = Two thresholds associated with the conditionally sampled datasets * Copula_Family1/Copula_Family2 = Two families of the two fitted copulas * Marginal_Dist1/Marginal_Dist2 = Selected non-extreme margnal distributions * RP = Return Period of interest * N = size of the sample from the fitted joint distributions used to estimate the density along the isoline of interest * N_Ensemble = size of the ensemble of events sampled along the isoline of interest.

```
S20.Bivariate<-Design_Event_2D(Data=S20.Detrend.df[,-c(1,4)], Data_Con1=S20.Rainfall$Data,
Data_Con2=S20.OsWL$Data, Thres1=0.97, Thres2=0.97,
Copula_Family1=S20.Copula.Rainfall, Copula_Family2=S20.Copula.OsWL,
Marginal_Dist1="Logis", Marginal_Dist2="Twe",
x_lab="Rainfall (Inches)",y_lab="O-sWL (ft NGVD 29)",
RP=100,N=10,N_Ensemble=10)
```

```
## When the response variable contains exact zeros, all values of p must be between 1 and 2; other valu
## 1.5 1.7 1.9
## ...Done.
```



Design event according to the "Most likely" event approach (diamond in the plot)

```
S20.Bivariate$MostLikelyEvent
```

```
##   Rainfall     OsWL
## 1     1.51 3.014642
```

Design event under the assuption of full dependence (Triangle in the plot)

```
S20.Bivariate$FullDependence
```

```
##   Rainfall     OsWL
## 1    11.64 3.051776
```

5. Trivariate analysis

In the report three higher dimensional (>3) approaches are implemented to model the joint distribution of rainfall, O-sWL and groundwater level, they are:

- Standard (trivariate) copula

- Pair Copula Construction
- Heffernan and Tawn (2004)

In the package, each approach has a `_Fit` and `_Sim` function. The latter requires a `MIGPD` object as its `Marginals` input argument, in order for the simulations on $[0,1]3$ to be transformed back to the original scale. The `Migpd_Fit` command fits independent GPDs to the data in each row of a dataframe (excluding the first column if it is a "Date" object) creating a `MIGPD` object.

```
S20.Migpd<-Migpd_Fit(Data=S20.Detrend.Declustered.df[,-1],mqu=c(0.975,0.975,0.9676))
summary(S20.Migpd)
```

```
## $d
## [1] 3
##
## $conv
## NULL
##
## $penalty
## [1] "gaussian"
##
## $co
##                     Rainfall       OsWL  Groundwater
## Threshold         1.4200000 1.79170278 2.7067623197
## P(X < threshold) 0.9750000 0.97500000 0.9676000000
## sigma             0.8899232 0.05894553 0.2034300877
## xi                0.1642760 0.62630408 0.0009731279
## Upper end point         Inf        Inf          Inf
##
## attr(,"class")
## [1] "summary.migpd"
```

Standard (trivariate) copula are the most conceptually simple of the copula based models, using a single parametric multivariate probability distribution as the copula. The `Standard_Copula_Fit()` function fits elliptic (specified by `Gaussian` or `tcop`) or Archimedean (specified by `Gumbel`,`Clayton` or `Frank`) copula to a trivariate dataset. Let first fit a Gaussian copula

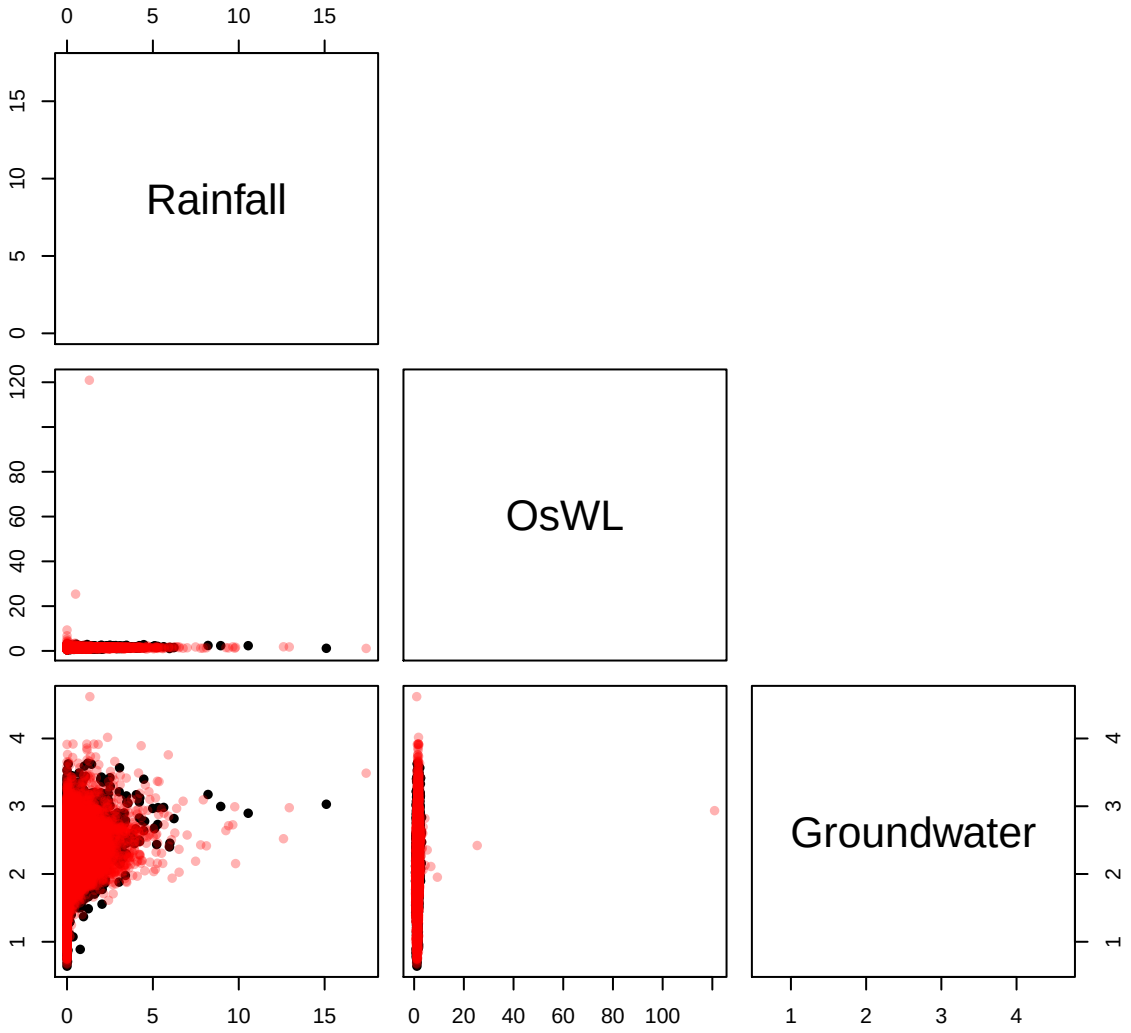
```
S20.Gaussian<-Standard_Copula_Fit(Data=S20.Detrend.df,Copula_Type="Gaussian")
```

From which the `Standard_Copula_Sim()` function can be used to simulate a synthetic record of N years

```
S20.Gaussian.Sim<-Standard_Copula_Sim(Data=S20.Detrend.df,Marginals=S20.Migpd,
Copula=S20.Gaussian,N=100)
```

Plotting the observationed and simulated values

```
S20.Pairs.Plot.Data<-data.frame(rbind(na.omit(S20.Detrend.df[,-1]),S20.Gaussian.Sim$x.Sim),
c(rep("Observation",nrow(na.omit(S20.Detrend.df))),
rep("Simulation",nrow(S20.Gaussian.Sim$x.Sim))))
colnames(S20.Pairs.Plot.Data)<-c(names(S20.Detrend.df)[-1],"Type")
#Using the scale package for transparent colours
library("scales")
pairs(S20.Pairs.Plot.Data[,1:3],
col=ifelse(S20.Pairs.Plot.Data$Type=="Observation","Black",alpha("Red",0.3)),
upper.panel=NULL,pch=16)
```



The `Standard_Copula_Sel()` function can be used to deduce the best fitting in terms of AIC

```
Standard_Copula_Sel(Data=S20.Detrend.df)
```

```
## Warning in var.mpl(copula, u): the covariance matrix of the parameter
## estimates is computed as if 'df.fixed = TRUE' with df = 16.1271469165612
```

```
##     Copula        AIC
## 1 Gaussian -1389.1438
## 2    t-cop -1434.7850
## 3   Gumbel  -876.7537
## 4  Clayton  -565.7595
## 5    Frank  -854.4284
```

Standard trivariate copulas lack flexibility to model joint distributions where heterogeneous dependencies exist between the variable pairs. decomposes an d-dimensional probability distribution into the product of a cascade of bivariate copulas (some of which are conditional) and the marginal densities of each variable. As the dimensionality of the problem increases the number of mathematically equally valid decompositions quickly becomes large. Bedford and Cooke (2001),Bedford and Cooke (2002) introduced the regular vine, a

graphical model which helps to organize the possible decompositions. The Canonical (C-) and D- vine are two commonly utilized sub-categories of regular vines, in the trivariate case a vine copula is simultaneously a C- and D-vine. Lets fit a regular vine copula model

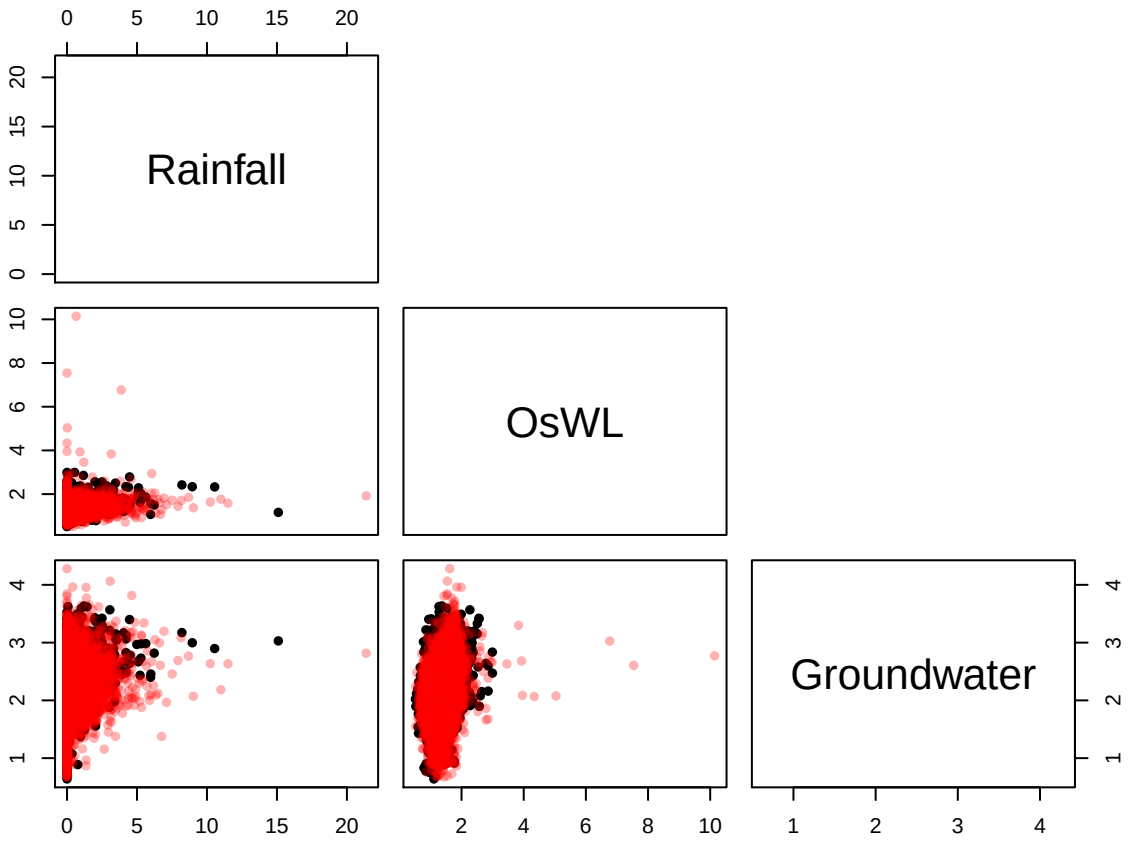
```
S20.Vine<-Vine_Copula_Fit(Data=S20.Detrend.df)
```

From which the `Vine_Copula_Sim()` function can be used to simulate a synthetic record of N years

```
S20.Vine.Sim<-Vine_Copula_Sim(Data=S20.Detrend.df,Marginals=S20.Migpd,
```

Plotting the observationed and simulated values

```
S20.Pairs.Plot.Data<-data.frame(rbind(na.omit(S20.Detrend.df[,-1]),S20.Vine.Sim$x.Sim),
c(rep("Observation",nrow(na.omit(S20.Detrend.df))),
rep("Simulation",nrow(S20.Vine.Sim$x.Sim))))
colnames(S20.Pairs.Plot.Data)<-c(names(S20.Detrend.df)[-1],"Type")
pairs(S20.Pairs.Plot.Data[,1:3],
col=ifelse(S20.Pairs.Plot.Data$Type=="Observation","Black",alpha("Red",0.3)),
upper.panel=NULL,pch=16)
```



Finally, lets implement the Heffernan and Tawn (2004) approach (here on in HT04), where a non-linear regression model is fitted to the (joint) observations where a (conditioning) variable is above a specified threshold. The regression model typically adopted is

$$\mathbf{Y}_{-i} = \mathbf{a}Y_i + Y_i^b\mathbf{Z} \qquad for \qquad Y_i > v$$

where $\mathbf{Y}$ is a set of variables transdformed to a common scale, $\mathbf{Y}_{-i}$ is the set of variables excluding Y_{-i}, $\mathbf{a}$ and $\mathbf{b}$ are vecotrs of regression parameters and $\mathbf{Z}$ is a vecotr of residuals. The dependence structure, when a specified variable is extreme is thus captured by the regression parameters and the joint residuals. The procedure is repeated conditioning on each variable in turn to build up of the joint distribution when at least one variable is in an extreme state. The `HT04` command fits and simulates N years worth of simulations from the model.

```
S20.HT04<-HT04(data_Detrend_Dependence_df=S20.Detrend.df,
data_Detrend_Declustered_df=S20.Detrend.Declustered.df,
u_Dependence=0.995,Migpd=S20.Migpd,mu=365.25,N=1000)
```

Output of the function includes the three conditional `Models`, proportion of occasions where each variable is most extreme given at least one variable is extreme `prop`as well as, the simulations on the transfromed scale `u.Sim` (gumbel by default) and original scale `x.Sim`. Lets view the fitted model when conditioning on rainfall

```
S20.HT04$Model$Rainfall
```

```
## mexDependence(x = Migpd, which = colnames(data_Detrend_Dependence_df)[1],
##     dqu = u_Dependence, margins = Margins, constrain = FALSE,
##     v = V, maxit = Maxit)
##
##
## Marginal models:
##
## Dependence model:
##
## Conditioning on Rainfall variable.
## Thresholding quantiles for transformed data: dqu = 0.995
## Using gumbel margins for dependence estimation.
## Log-likelihood = -113.7848 -88.48197
##
## Dependence structure parameter estimates:
##     OsWL Groundwater
## a 1.0000      0.4026
## b 0.6947     -1.1920
```

```
S20.HT04$Prop
```

```
## NULL
```

and the which the proporiton of the occasions in the original sample that rainfall is the most extreme of the drivers given that at least one driver is extreme.

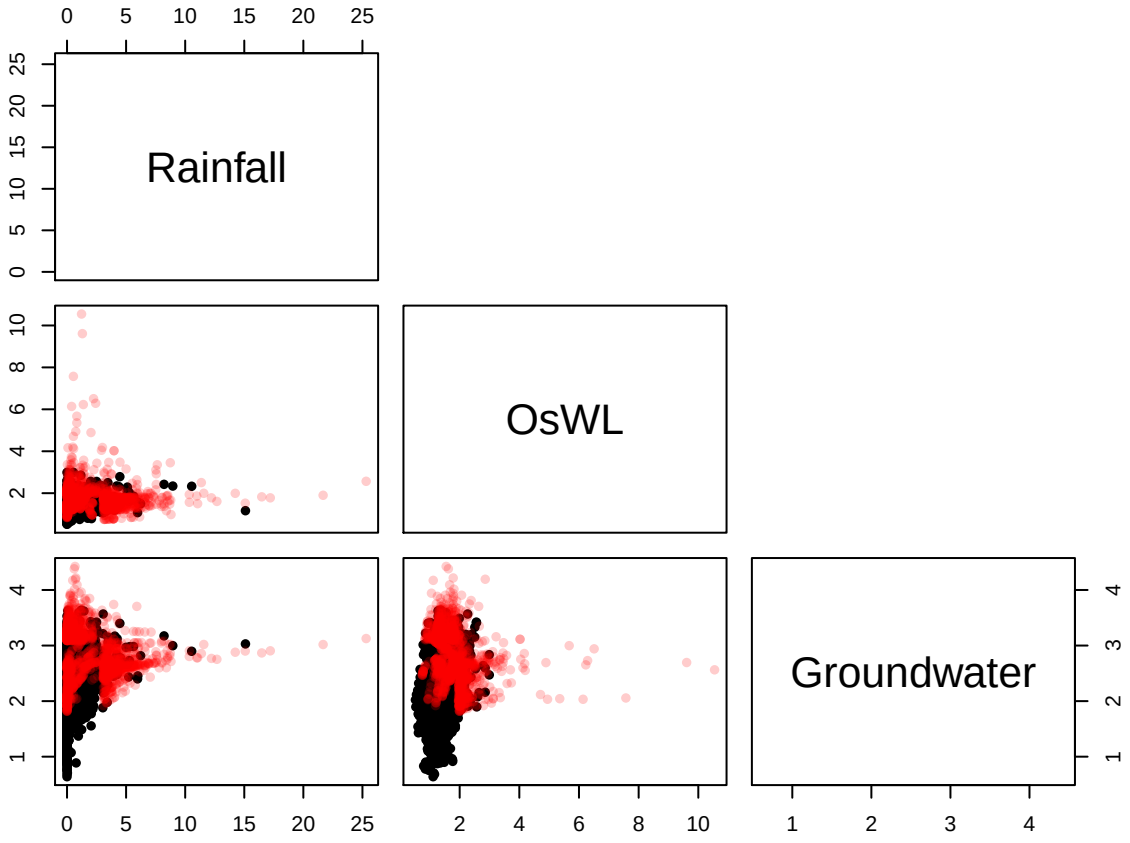
The HT04 approach uses rejection sampling to generate synthetic records. The first step involves sampling a variable, conditioned to exceed the `u_Dependence` threshold. A joint residual associated with the corresponding regression is independently sampled and realizations of the other variables estimated using the fitted regression parameters. If the variable conditioned to be extreme in step one is not the most extreme the sample is rejected. The process is repeated until the relative proportion of simulated events where each variable is a maximum, conditional on being above the threshold, is consistent with the empirical distribution. Labelling the simulations `S20.HT04.Sim`

```
S20.HT04.Sim<-S20.HT04$x.sim
```

and now plotting the simulations from the HT04 model

```
S20.Pairs.Plot.Data<-data.frame(rbind(na.omit(S20.Detrend.df[,-1]),S20.HT04.Sim),
c(rep("Observation",nrow(na.omit(S20.Detrend.df))),
rep("Simulation",nrow(S20.HT04.Sim))))
```

```
colnames(S20.Pairs.Plot.Data)<-c(names(S20.Detrend.df)[-1],"Type")
pairs(S20.Pairs.Plot.Data[,1:3],
col=ifelse(S20.Pairs.Plot.Data$Type=="Observation","Black",alpha("Red",0.2)),
upper.panel=NULL,pch=16)
```



References

Bedford, T. and Cooke, R.: Probability density decomposition for conditionally dependent random variables modeled by vines, Annals of Mathematics and Artificial Intelligence., 32, 245–268, 2001.

Bedford, T. and Cooke, R.: Vines – a new graphical model for dependent random variables, Annals of Statistics, 30, 1031–1068, 2002.

Bender, J., Wahl, T., Müller, A. and Jensen, J.: A multivariate design framework for river confluences, Hydrological Sciences Journal, 61(3), 471–482, 2016.

Heffernan, J. and Tawn, J.: A conditional approach for multivariate extreme values (with discussion), Journal of the Royal Statistical Society: Series B (Statistical Methodology), 66(3), 497–546, 2004.

Jane, R., Wahl, T., Cadavid, L. and Jayantha, O.: Multivariate statistical modelling of the drivers of compound flooding in south florida, Natural Hazards and Earth System Science, In review, 2020.