WACSgen v1.0
User guide and description of the model

D. Allard
Biostatistiques et Processus Spatiaux, INRA, Avignon, France

January 13, 2016

Abstract

This document presents the stochastic weather generator WACSgen with its associated models, methods and algorithms.

WACSgen is a unisite stationary multivariate weather generator for daily climate variables based on weather-states that uses a Markov chain for modeling the succession of (an unlimited number of) weather states. Conditionally to the weather states, the multivariate variables are modeled using the family of Complete skew-normal distributions. It is described in Flecher *et al.* (2010). Since this first paper, the model has slightly changed, in particular as regards the modeling of the temporal autocorrelation.

This document provides is organized in three, relatively independent, parts. The first part is a classical user guide. It provides a workflow description of WACSgen. The second part describes in details the model, the methods, the associated algorithms and the mathematics involved. The third part is a description of the main functions and variables in WACSgen, thus providing a more detailed account of the implementation as a set of `R` functions.

Part I
User guide

1 Introduction and general description

WACSgen is a single site multivariate weather generator for daily climate variables, based on weather-states. The general features of WACSgen v1.0 are:

1. Annual cycles for means and variances are removed, and residuals are computed for all variables, except precipitation. Daily rainfalls are modeled for each season independently, according to parametric models (Gamma distribution or transformed truncated Gaussian distribution).

2. The modeling of the residuals is seasonal. Seasons are provided by the user; they are not estimated from the data.

3. There is a hidden discrete variable describing the succession of weather states. A distinction between dry days (precip = 0) and wet days (precip > 0) is made. Then, clusters are estimated on residuals for dry days (resp. wet days) with the help of the `Mclust` package. The transition matrix between weather days is then estimated.

4. For each season and each weather state a closed skew-normal (CSN) distribution is fitted. It models the vector of residuals of two consecutive days belonging to the same weather state. In this model, the following statistical properties of the residuals are estimated:

   (a) the multivariate distribution of the variables;

   (b) the temporal correlation for each variable.

   How the residuals are precisely modeled and how their parameters are estimated will be explained in details in Section 9.

The general workflow fur running WACSgen is:

Data preparation → Estimation of the parameters → Simulation of new series → Validation

There are 6 main functions to perform a complete workflow:

1. Preparing the data: function `WACSdata`

2. Estimating the parameters: function `WACSestim`

3. Simulating new series: function `WACSsimul`
4. Validating the simulations: function `WACSvalid`
5. Comparing two datasets or two simulation series: function `WACScompare`
6. Plotting figures: `WACSplot`

2 Preparing the data

Data needs to be prepared for the estimation step of the workflow. This is done by calling the function `WACSdata`

```
myWACSdata = WACSdata(mydata, mapping=NULL, bounds=NULL, from=NULL, to=NULL,
                      skip=NULL, Trange=FALSE, seasons=season.limits)
```

This function creates a structure belonging to the class `'WACSdata'`. The following organization of the data must be verified in the dataframe `mydata`:

- There is one record per line. Lines correspond to consecutive daily measurements.
- The three first columns must indicate: year, month, day.
- After these, the first variable must always be precipitation, which is thus mandatory. The other variables can be any variable, such as temperatures, radiation, wind speed, etc. In principle, there is no limitation to the number of variables.
- Data are organized in columns. It is left to the user to verify that the data are in a format that can be read by the function `WACSdata`, e.g.

```
> mydata[1,]
  year month day rain  V1  V2  V3 V4
1 1972     1   1    0 0.8 4.1 161  2
>
```

- WACSgen makes use of certain variable names for internal usage. This is the case for `"year","month","day","rain","tmin","tmax","RG","V","ETPP"`, where `"RG", "V"` and `"ETPP"` indicate respectively "Daily radiation", "wind speed" and "daily humidity". It is recommended to use these names whenever these variables are recorded. These variables are not mandatory for WACSgen. The only mandatory variable is `rain`.

Several options are possible when calling the function `WACSdata`:

- A temporal window of `mydata` can be selected using the date variables `from` and `to`, which can be useful for comparing subsets of very long series. The variable `skip` allows to skip useless columns.
- The variable `bounds` allows to indicate absolute bounds for some variables (e.g. positive variables), which will be enforced during the simulation step. It must be provided as a list of lists, e.g.

```
> mybounds = list( rain=list(min=0, max=7), tmax=list(-5, 45) )
```

If `bounds=NULL`, bounds are computed from the data. Some variables will have minimal values set automatically to 0 (`trange,V,RG,ETPP`) and maximal values to 100 (`ETPP`). Other minimum (resp. maximum) values are computed by adding (resp. subtracting) to the maximum (resp. minimum value) its difference to the 10th largest (resp. lowest) value.

- The variable `skip` allows to skip some variables in the dataframe `mydata`.
- If minimum and maximum temperature are recorded, WACSgen offers the choice of modeling the variables `(tmin,tmax)` or `(tmin,trange = tmax-tmin)`. In this case, `tmin` must be in the second column and `tmax` in the third one. This choice is specified by the Boolean variable `Trange`. Default is `Trange = FALSE`, which makes no transformation. In the simulation step, the condition `tmax > tmin` will always be imposed. If there is only a mean temperature, or no temperature at all, one must set `Trange = FALSE`.
- The variable `seasons` is a vector of days indicating the first day of each season. For example, for the 4 seasons MAM, JJA, SON, DJF, one must define

  ```
  > season.limits=c("03-01", "06-01", "09-01", "12-01")
  ```

  There can be any number of seasons. The length of the vector `season.limits` defines the number of season. One can impose a single season by setting a vector of length 1.

The function `WACSdata` returns a structure belonging to the class `'WACSdata'`. It does the following tasks:

1. Suppressing 29th of February;
2. Selecting the period of time;
3. Creating a vector of seasons;
4. If `Trange = True`, transforming `(tmin,tmax)` into `(tmin,trange = tmax-tmin)`.
5. Creating the bounds for any variable for which bounds are not provided.

The structure of the class `'WACSdata'` is detailed in Section 10.

3 Estimating the parameters of the model

The function `WACSestim` does the estimation of the parameters. Data must have been previously prepared by `WACSdata`.

```
myParam = WACSestim(myWACSdata,spar=0.7,trend.norm="L2",rain.model="Gamma",
                    method="MLE",Vsel=NULL,Nclusters=NULL,clustering="soft",
                    plot.it=FALSE,DIR="./")
```

The estimation of the parameters involves several steps calling dedicated internal functions (see Section 11).

1. The seasonal cycle is removed on all variables, except precipitation. A smoothed version of the central tendency and a smoothed version of a measure of deviation are computed. Two options are possible: mean + standard deviation (`trend.norm = "L2"`) or median + absolute deviation (`trend.norm = "L1"`). Figure 1 (left panel) illustrates how the trend is removed for minimum temperature. The parameter `spar` controls the amount of smoothing when estimating the trends. Larger values of `spar` produce smoother estimates. Smaller values produce less smooth estimates. The recommended compromise is `spar=0.7`.

2. Daily rainfalls, `rain`, are modeled for each season independently. Two models are proposed for `rain`: the Gamma distribution (`rain.model ="Gamma"`) and the model proposed in Allard and Bourotte (2013) (`rain.model ="AB"`; not yet implemented). Parameters can be estimated by Maximum likelihood (`method= "MLE"`) or by a method of moment. (`method= "MOM"`). Figure 1 (right panel) shows the precipitation modeling using a Gamma distribution. There is also an option `rain.model ="None"`, which makes no transformation on the rain data. This option allows to use its own precipitation transformation outside `WACSgen`. In this case, simulated values are Gaussian and need to be transformed during a post-processing step, outside `WACSgen`.

3. Clusters of residuals define weather states. The clustering itself is obtained by calling the function `Mclust` of the package `mclust` (Fraley and Raftery, 2003; Fraley et al., 2012).

   - The number of clusters can either be specified with the parameter `Nclusters` or left unspecified (with `Nclusters=NULL`), in which case the optimal number of clusters less or equal to 4 is estimated according to a BIC criterion.
   - The variables used for estimating the clusters can be specified using the parameter `Vsel`. By default (when `Vsel=NULL`) all variables are taken into account.
   - Clustering is run independently and separately on dry and wet days. Obviously, `rain` is not considered for dry days.
   - Classification of weather states can either be "hard" or "soft". This is controlled by the parameter `clustering`. When `clustering=soft`, one given day belongs to each weather state according to a probability distribution. When `clustering=hard`, the probability is set to 1 to the most likely weather state and 0 to all others.

     The estimation of the transition matrix between weather states and the estimation of the parameters of the multivariate density in each weather state will take into account all data, weighted by their probabilities of belonging to the given weather state. Estimates are found to be more robust when `clustering="soft"`.

4. The transition matrix between weather states and the parameters of the temporal multivariate density are then estimated. Densities are assumed to belong to the Complete Skew-Normal class of densities. No control parameters are needed for this step. Details regarding the model and the estimation method are given in Section 9.

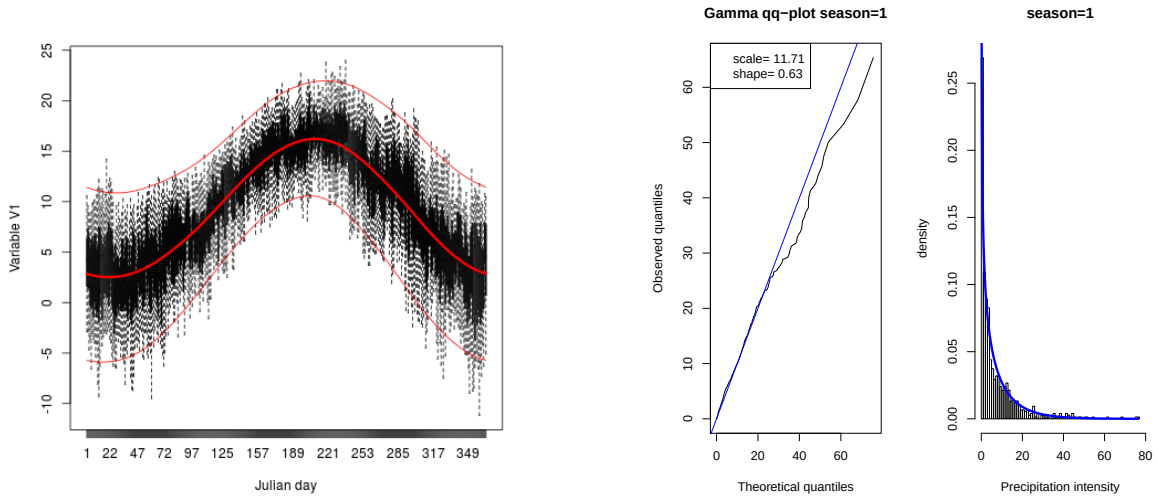

Figure 1: *Left panel: Removing the seasonal trend for minimum temperature. In red: smoothed average and the lower and upper bounds (average ±2 standard deviation. In black: daily box-plots. Right panel: Modeling positive precipitation with a Gamma distribution. Left: qq-plot. and experimental and theoretical distribution.*

All parameters are stored in a structure belonging to the class `'WACSpar'`, which is a list of lists. At the upper level, it contains 7 items + one list per season. For example, the call

```
myParam = WACSestim(myWACSdata,spar=0.7,trend.norm="L2",rain.model="Gamma",
                    method="MLE",Vsel=NULL,Nclusters=NULL,clustering="soft",
                    plot.it=FALSE,DIR="./")
```

returns a list with 11 items, since 4 seasons are defined in `myWACSdata`. The object `myParam` contains everything that is needed to perform simulations and to draw figures. Its structure is detailed in Section 3.

4 Performing the simulations

Simulations are performed by calling the function `WACSsimul`

```
mySim = WACSsimul(myParam, from, to, first.day, REJECT=FALSE)
```

This function requires the parameters `myParam`, which must belong to the class `'WACSpar'` as returned by the function `WACSestim` (see Section 3). The variables `from` and `to` indicate the dates of the beginning and the end of the time window for the simulation. They can be equal to or different than those used for the estimation. The variable `first.day` can be used to force the weather variables to som given values. This can be useful in some cases, e.g. to perform non stationary simulations by calling successive runs of WACSgen with different parameters. `first.day` must be a vector with the same structure as the weather variables values in `myWACSdata` (i.e. from column 5 to column $4 + Nv$, where Nv is the number of weather variables.

The simulation is done sequentially: day `d` is simulated conditionally on the values at day `(d-1)`. The variable `REJECT` is a Boolean. When `REJECT = TRUE` a rejection technique is used to guarantee that the variables stay within the bounds returned by the function `WACSdata`. Default is `FALSE`. In this case, values that have been simulated outside the bounds are forced to the bounds. The rejection technique tends to produce biases. Setting `REJECT=FALSE` is thus recommended.

5 Validating and comparing the simulations

5.1 Validation

Simulations can be validated, by calling the function `WACSvalid`. Some statistics are computed on the simulated data, and compared with the same statistics computed on the recorded data.

```
myValid = WACSvalid(what="Sim",wacsdata=myWACSdata,wacspar=myParam,
                    wacssimul=mySim,varname=myvar,varname2=NULL,
                    base = 0,above=TRUE,months=1:12)
```

This functions requires data, parameters and simulated series belonging, respectively, to the class `'WACSdata'`, `'WACSpar'` and `'WACSsimul'`. Several comparison analysis are proposed. The analysis is specified by choosing one of the following values for the variable `what`:

`what = "Sim"`	Simulations are compared to the data (see Figure 2).
`what = "Rain"`	QQ-plots of simulated vs. recorded precipitation are displayed for each seasons.
`what = "MeanSd"`	Monthly means and standard deviations are compared (see Figure 3).
`what = "BiVar"`	Monthly bivariate correlations coefficients are compared.
`what = "CorTemp"`	Monthly temporal correlations are compared.
`what = "SumBase"`	The sums of the variable above a given threshold are compared.
`what = "Persistence"`	The distributions of the number of consecutive days above a given threshold are compared (see Figure 4).

The comparison is done for the variable `varname`. The option `what = "BiVar"` requires a second variable, `varname2`. The options `what = "SumBase"` and `what = "Persistence"` require a threshold, `base`, as well as a Boolean, `above`, indicating whether sums and persistence are computed above (`above=TRUE`) or below (`above=FALSE`) the threshold. For these two options, the analysis can be restricted during a set of months described in the variable `months`. Defaults are `varname2=NULL`, `base=0`, `above=TRUE` and `months=1:12`.

The object `myValid` will be used later to produce plots with `WACSplot`.

5.2 Comparison

Two series belonging to the same class (either `'WACSdata'` or `'WACSsimul'`) can be compared by calling the function `WACScompare`. This function has a syntax very similar to the function `WACSvalid`

```
myComp = WACScompare(what="Sim",wacs1=mySim1,wacspar=myParam,wacs2=mySim2,
```

```
                    varname=myvar,varname2=NULL,base = 0,above=TRUE,
                    months=1:12)
```

Here, the function is used to compare two simulated series (e.g. obtained with different parameters). It can also be used to compare two recorded series (at different locations, or at different period of time)

```
myComp = WACScompare(what="Sim",wacs1=myWACSdata1,wacspar=myParam,
                    wacs2=myWACSdata2, varname=myvar,varname2=NULL,
                    base = 0,above=TRUE,months=1:12)
```

The object `myComp` will be used later to produce plots with `WACSplot`.

5.3 Producing plots

There are two functions for producing plots. The function `WACSplot` is applied to objects returned by `WACSvalid` and `WACScompare` to produce a PDF plot with file name `file`. It does not return anything.

```
WACSplot(myValid,file="myFile.pdf")
```

The function `WACSplotdensity` produces a bivariate plot of the residuals, with the fitted density superimposed. The user must specify the chosen variables (through the vector `dimens`), the season (variable `season`), and whether dry or wet weather states must be considered (respectively, `dry = TRUE` or `dry = FALSE`). The function produces a PDF plot which will be stored in the directory `DIR`.

```
WACSplotdensity(wacsdata = myWACSdata,wacspar = myParam,season=2,dimens=c(1,2),
                dry=TRUE,DIR="./")
```

The vector `dimens` specifies the index of the variable(s) with the convention that the first index corresponds always to `rain`. During dry days, the associated values are equal to 0. If `length(dimens)=1`, the bivariate density of the variable at days `(d,d+1)` is plotted. If `length(dimens)=2`, the same-day bivariate density of the pair of variables is plotted. Figure 5 depicts the bivariate plot of the residuals corresponding to `(tmin,tmax)` during the second season, here equal to MAM.

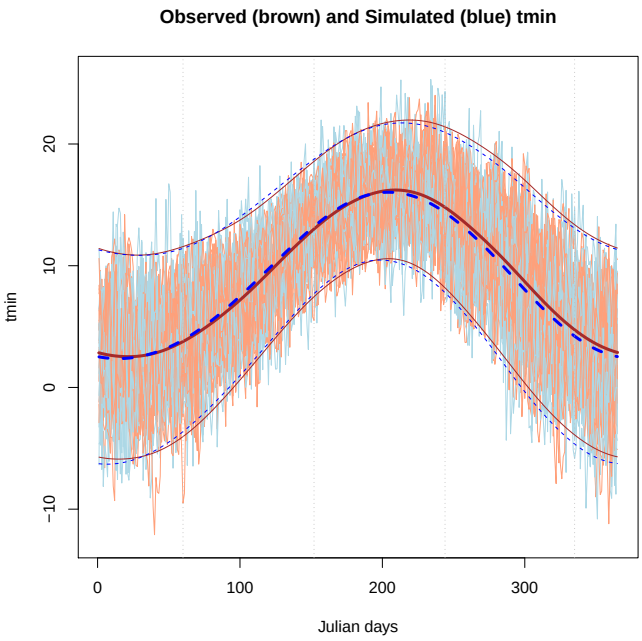

Figure 2: *Variable* `tmin`*: observed (brown) and simulated (blue) simulations. Thick lines: smoothed mean. Dashed lines: smoothed means standard deviations.*

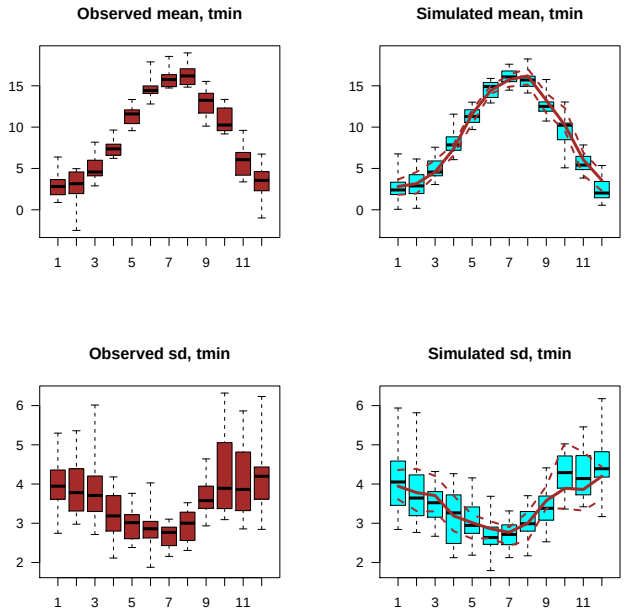

Figure 3: *Variable* `tmin`*: observed (brown) and simulated (blue) monthly boxplots of means (top row) and standard deviations (bottom row). Solid lines: monthly observed medians (equal to the black lines of the observed boxplots). Dashed lines: monthly Q1 and Q3 (equal to the boxes of the observed boxplots).*

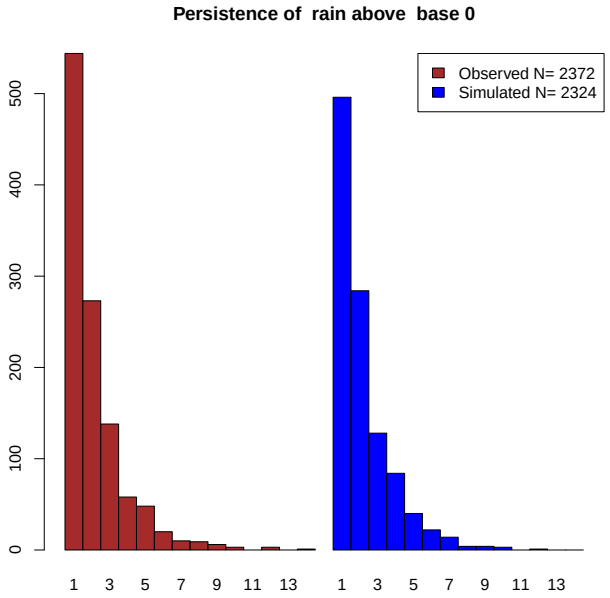

Figure 4: *Distribution wetspells for the observed (brown) and simulated (blue) data.*

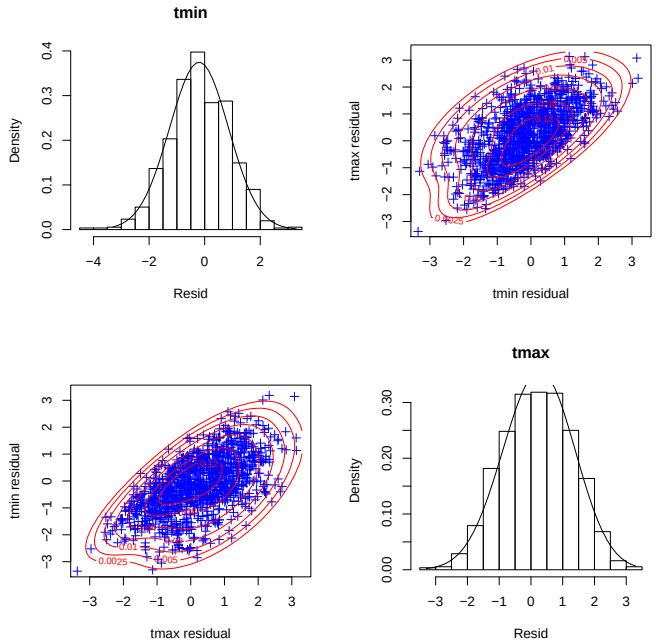

Figure 5: *Bivariate plot of the residuals corresponding to* `(tmin,tmax)` *during dry days of the second season (here equal to MAM).*

6 A simple example

Below is the R script that was used to produce all Figures shown in this documents.

```
library(WACS)
data(ClimateSeries)

ThisData = WACSdata(ClimateSeries, from="1995-01-01", to="2012-12-31",
           plot.it=TRUE, seasons=c("03-01", "06-01", "09-01", "12-01"))
ThisPar  = WACSestim(ThisData, plot.it=T, Nclusters=1:2)
ThisSim  = WACSsimul(ThisPar, from="1995-01-01", to="2012-12-31")

Comp = WACSvalid(what="Sim", wacsdata = ThisData, wacspar = ThisPar,
                 wacssimul = ThisSim, varname="tmin")
pdf("Sim_tmin.pdf")
WACSplot(Comp)
dev.off()

Comp = WACSvalid(what="MeanSd", wacsdata = ThisData, wacspar = ThisPar,
                 wacssimul = ThisSim, varname="tmin")
pdf("Mean_Sd.pdf")
WACSplot(Comp)
dev.off()

Comp = WACSvalid(what="Persistence", wacsdata = ThisData, wacspar = ThisPar,
                 wacssimul = ThisSim, varname="rain", base=0, above=TRUE)
pdf("Wet_spell.pdf")
WACSplot(Comp)
dev.off()

WACSplotdensity(ThisData,ThisPar,season=2,dimens=c(2,3),dry=TRUE)
```

Part II
Models and methods in WACSgen

7 Modeling precipitation

Daily rainfalls (denoted P) are modeled for each season independently. Two models are proposed for P: the Gamma distribution (`rain.model = "Gamma"`) and the model proposed in Allard and Bourotte (2013), hereafter referred to as the AB model (`rain.model = "AB"`); not yet implemented). The gamma distribution was chosen for its flexibility to model distributions of precipitation. It is widely used in the hydrology literature. Once the parameters have been estimated, data are transformed with the use of the fitted cumulative distribution function (cdf): $\tilde{P} = \Phi^{-1}(G_{\hat{\theta}}(P))$, where G represents the fit by a Gamma cdf and Φ^{-1} corresponds to the inverse of the standardized Normal cdf. $\tilde{P}$ is thus modeled as a Gaussian random variable; for a given season and a given weather state, $\tilde{P}$ will be considered as a CSN (see below) in order to account for possible asymmetries within clusters. Parameters can be estimated by Maximum likelihood (`method= "MLE"`) or by a method of moment. (`method= "MOM"`). Considering the discretization of rainfall data, it is commonly observed that small values are well fitted with MLE, but that occurrences of large value are underestimated, whereas the converse is observed with MOM.

Model AB will be coded later. This model is expected to give more flexibility and better fit.

There is also an option `rain.model = "None"`, which makes no transformation on the rain data. This option allows to use its own precipitation transformation outside `WACSgen`. In this case, simulated values are Gaussian and need to be transformed during a post-processing step, outside `WACSgen`.

8 Modeling the weather states

8.1 Estimating the weather states

Clusters of residuals define weather states. The clustering itself is obtained by calling the function `Mclust` of the package `mclust` (Fraley and Raftery, 2003; Fraley et al., 2012). The number of clusters can either be specified or left unspecified, in which case the optimal number of clusters is estimated according to a BIC criterion. The variables on which the clusters are estimated can also be specified. By default, all variables are taken into account for determining the clusters. Clustering are run independently and separately on dry and wet days.

Classification of weather states can either be "hard" or "soft". With a "soft" classification, each day d belongs to weather states with a probability z_w, with $w = 1, \ldots, W$ and $\sum_{w=1}^{W} z_w = 1$, where W is the number of weather states. With "hard" classification, probabilities belong to $\{0, 1\}$, i.e. one and only one weather state is assigned to each day d.

The estimation of the parameters of a weather state involves the computation of empirical moments: mean, covariance, weighted moment, etc. Their computation take into account the clustering variable z_w. In case of "hard" clustering, they are equal to the usual moments. In case of "soft" clustering these computations take into account all data, weighted by their probabilities of belonging to the given weather state. The estimation of the transition matrix between weather states presented in the next section provides an illustration of how "hard" or "soft" classification can be used.

8.2 Estimating the transition matrix

The weather state transition probabilities are simply estimated by

$$\hat{p}_{w,w'} = \frac{\sum_d I[W(d) = w, W(d+1) = w'] z_w(d) z_{w'}(d+1)}{\sum_t z_w(d) z_{w'}(d+1)}, \tag{1}$$

in which $z_w(d)$ is the probability of the weather state being w at day d and $I[A]$ is the indicator function equal to 1 if condition A is satisfied and equal to 0 otherwise. The probabilities $z_w(d)$ are by-products of the clustering step. They can be transformed into 0-1 values if they are set to 1 for the most likely weather-state and to 0 to all other ones.

9 Temporal complete skew-normal models for residuals

9.1 Some reminders on complete skew-normal distributions

For each season and each weather state, we assume a Closed Skew-normal (CSN) distribution for the $k = N_v$ (dry weather state) or $k = N_v + 1$ (wet weather states) variables. Skew-normal distributions are extensions of the normal distribution which admit skewness whilst retaining most of the interesting properties of the Gaussian distribution. An overview of theoretical and applied developments related to skewed distributions is provided in Genton (2004). Most theoretical results about closed skew-normal distributions can be found in Gonzàlez-Farías, Domínguez-Molina and Gupta (2004), Domínguez-Molina, Gonzàlez-Farías and Gupta (2003) and Gupta and Aziz (2012).
A k-dimensional random vector $\mathbf{Y}$ is said to have a multivariate closed skew-normal distribution, denoted by $\text{CSN}_{k,l}(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{D}, \boldsymbol{\nu}, \boldsymbol{\Delta})$, if its density function is of the form

$$f_{k,l}(\mathbf{y}) = c_l \; \phi_k(\mathbf{y}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \; \Phi_l(\mathbf{D}(\mathbf{y} - \boldsymbol{\mu}); \boldsymbol{\nu}, \boldsymbol{\Delta}), \text{ with } c_l^{-1} = \Phi_l(\mathbf{0}; \boldsymbol{\nu}, \boldsymbol{\Delta} + \mathbf{D}\boldsymbol{\Sigma}\mathbf{D}'), \tag{2}$$

where $\boldsymbol{\mu} \in \mathbb{R}^k$ and $\boldsymbol{\nu} \in \mathbb{R}^l$ are both location vectors, $\boldsymbol{\Sigma} \in \mathbb{R}^{k \times k}$ and $\boldsymbol{\Delta} \in \mathbb{R}^{l \times l}$ are both covariance matrices, $\mathbf{D} \in \mathbb{R}^{l \times k}$, $\phi_k(\mathbf{y}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ and $\Phi_k(\mathbf{y}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ are the probability distribution function (pdf) and cumulative distribution function (cdf), respectively, of the k-dimensional normal distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$, and $\mathbf{D}'$ is the transpose of the matrix $\mathbf{D}$. In the particular case $\mathbf{D} = \mathbf{0}$, $\mathbf{Y}$ is the usual k-dimensional normal distribution with mean $\boldsymbol{\mu}$ and variance covariance matrix $\boldsymbol{\Sigma}$. CSN distributions defined by (2) are over-parametrized (Gonzàlez-Farías, Domínguez-Molina and Gupta, 2004). Without loss of generality $\boldsymbol{\nu}$ is thus set equal to $\mathbf{0}$. In practice, the normalizing constant c_l^{-1} defined in (2) can be difficult to compute. The moment generating function of the CSN distribution (2) is:

$$M_Y(\mathbf{t}) = \frac{\Phi_l(\mathbf{D}\boldsymbol{\Sigma}\mathbf{t}; \boldsymbol{\nu}, \boldsymbol{\Delta} + \mathbf{D}\boldsymbol{\Sigma}\mathbf{D}')}{\Phi_l(\mathbf{0}; \boldsymbol{\nu}, \boldsymbol{\Delta} + \mathbf{D}\boldsymbol{\Sigma}\mathbf{D}')} \exp\{\mathbf{t}'\boldsymbol{\mu} + 1/2\mathbf{t}'\boldsymbol{\Sigma}\mathbf{t}\}$$

To simplify its expression, we assume, without loosing the skew-normal flexibility, that $k = l$, $\mathbf{D} = \mathbf{S}\boldsymbol{\Sigma}^{-\frac{1}{2}}$ and $\boldsymbol{\Delta} = \mathbf{I}_k - \mathbf{S}^2$ where $\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\Sigma}^{-\frac{1}{2}} = \boldsymbol{\Sigma}^{-1}$, $\mathbf{I}_k$ is the k-dimensional identity matrix and $\mathbf{S}$ is a diagonal matrix with elements in $[-1, 1]$ parametrizing the skewness for each variable. With this parametrization, equation (2) becomes

$$f_{k,k}(\mathbf{y}) = 2^k \; \phi_k(\mathbf{y}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \; \Phi_k(\mathbf{S}\boldsymbol{\Sigma}^{-\frac{1}{2}}(\mathbf{y} - \boldsymbol{\mu}); \mathbf{0}, \mathbf{I}_k - \mathbf{S}^2),$$

and the moment generating function simplifies to

$$M_Y(\mathbf{t}) = 2^k \Phi_k(\mathbf{S}\boldsymbol{\Sigma}^{-\frac{1}{2}}\mathbf{t}; \mathbf{0}, \mathbf{I}_k) \exp\{\mathbf{t}'\boldsymbol{\mu} + 1/2\mathbf{t}'\boldsymbol{\Sigma}\mathbf{t}\},$$

which is a much easier quantity to compute. Below, such a distribution which will be denoted

$$\mathrm{CSN}_k(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{S}\boldsymbol{\Sigma}^{-\frac{1}{2}}, \mathbf{0}, \mathbf{I}_k - \mathbf{S}^2) = \mathrm{CSN}^*_k(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{S}).$$

The transformed vector

$$\tilde{\mathbf{Y}}_k = \boldsymbol{\Sigma}^{-1/2}(\mathbf{Y} - \boldsymbol{\mu})$$

can be easily shown to be distributed as

$$\tilde{\mathbf{Y}}_k \sim \mathrm{CSN}^*(\mathbf{0}, \mathbf{I}, \mathbf{S}), \tag{3}$$

whose moment generating function is simply

$$M_{\tilde{Y}}(\mathbf{t}) = 2^k \Phi_k(\mathbf{S}\mathbf{t}; \mathbf{0}, \mathbf{I}_k) \exp\{1/2\mathbf{t}'\mathbf{t}\},$$

with $\Phi_k(\mathbf{S}\mathbf{t}; \mathbf{0}, \mathbf{I}_k) = \prod_k \Phi(S_k t_k; 0, 1)$.

9.2 Multivariate model of residuals with temporal autocorrelation

Let d and $d + 1$ denote two successive days. The augmented vector of residuals for two successive days is assumed to be distributed as a CSN*:

$$\begin{pmatrix} \mathbf{Y}_d \\ \mathbf{Y}_{d+1} \end{pmatrix} \sim \mathrm{CSN}^* \left(\begin{pmatrix} \boldsymbol{\mu}_d \\ \boldsymbol{\mu}_{d+1} \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Sigma}_{d,d} & \boldsymbol{\Sigma}_{d,d+1} \\ \boldsymbol{\Sigma}_{d+1,d} & \boldsymbol{\Sigma}_{d+1,d+1} \end{pmatrix}, \begin{pmatrix} \mathbf{S}_d \\ \mathbf{S}_{d+1} \end{pmatrix} \right). \tag{4}$$

In principle, each vector or matrix of parameter depends on the weather state observed at days d and $d + 1$. In order to maintain the number of parameters in a reasonable limit, we shall make a simplifying option: the matrix $\boldsymbol{\Sigma}_{d,d+1}$ depends only on $\boldsymbol{\Sigma}_d$, $\boldsymbol{\Sigma}_{d+1}$ and a vector of correlation coefficients written in a diagonal matrix $\mathbf{R}$:

$$\boldsymbol{\Sigma}_{d,d+1} = \boldsymbol{\Sigma}_d^{1/2} \mathbf{R} \boldsymbol{\Sigma}_{d+1}^{1/2}.$$

This model corresponds to separability between cross-correlations and temporal correlations. It is equivalent to the following conditional independence assumption: $Y_{i,d} \perp Y_{j,d+1} \mid Y_{i,d+1}$, for $i \neq j$. For two consecutive days, the model is thus[1]:

$$\begin{pmatrix} \mathbf{Y}_d \\ \mathbf{Y}_{d+1} \end{pmatrix} \sim \mathrm{CSN}^* \left(\begin{pmatrix} \boldsymbol{\mu}_d \\ \boldsymbol{\mu}_{d+1} \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Sigma}_d & \boldsymbol{\Sigma}_d^{1/2} \mathbf{R} \boldsymbol{\Sigma}_{d+1}^{1/2} \\ \boldsymbol{\Sigma}_{d+1}^{1/2} \mathbf{R} \boldsymbol{\Sigma}_d^{1/2} & \boldsymbol{\Sigma}_{d+1} \end{pmatrix}, \begin{pmatrix} \mathbf{S}_d \\ \mathbf{S}_{d+1} \end{pmatrix} \right), \tag{5}$$

where $\boldsymbol{\Sigma}_d^{1/2}$ is the only symmetric matrix such that $\boldsymbol{\Sigma}_d^{1/2}\boldsymbol{\Sigma}_d^{1/2} = \boldsymbol{\Sigma}_d$.

[1] Note that the lag-1 autocorrelation model in Eq. (5) is different that the one in Flecher et al. (2010) which was ill-defined.

- If the weather states are different in days d and $d+1$, the matrix $\mathbf{R}$ above is defined as the term by term maximum between the two correlation matrices:

$$\mathbf{R}[i] = \max\{\mathbf{R}_d[i], \mathbf{R}_{d+1}[i]\},\ i = 1, \ldots, k.$$

- If the number of variables is different in the weather states corresponding to d and $d+1$, the matrix $\mathbf{R}$ is rectangular, with a first column of 0s if d is dry and $d+1$ is wet, and a first line of 0s if d is wet and $d+1$ is dry.

This model is always well defined, i.e. the covariance matrix in (5) is always definite positive.

9.3 Two consecutive days in the same weather state

In what follows, we will assume that days d and $d+1$ are within the same weather state, with $k = N_v$. The model becomes

$$\begin{pmatrix} \mathbf{Y}_d \\ \mathbf{Y}_{d+1} \end{pmatrix} \sim \text{CSN}^* \left(\begin{pmatrix} \boldsymbol{\mu}_d \\ \boldsymbol{\mu}_d \end{pmatrix}, \begin{pmatrix} \boldsymbol{\Sigma}_d & \boldsymbol{\Sigma}_d^{1/2}\mathbf{R}\boldsymbol{\Sigma}_d^{1/2} \\ \boldsymbol{\Sigma}_d^{1/2}\mathbf{R}\boldsymbol{\Sigma}_d^{1/2} & \boldsymbol{\Sigma}_d \end{pmatrix}, \begin{pmatrix} \mathbf{S}_d \\ \mathbf{S}_d \end{pmatrix} \right). \tag{6}$$

Marginal distribution

Within this model, from the result in Appendix A, the marginal distribution of $\mathbf{Y}_d$ is:

$$\mathbf{Y}_d \sim \text{CSN}_{k,2k}(\boldsymbol{\mu}_d, \boldsymbol{\Sigma}_d, \mathbf{D}_{\text{marg}}, \mathbf{0}, \boldsymbol{\Delta}_{\text{marg}}), \tag{7}$$

with

$$\mathbf{D}_{\text{marg}} = \begin{pmatrix} \mathbf{S}_d & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_d \end{pmatrix} \begin{pmatrix} \mathbf{B} + \mathbf{C}\boldsymbol{\Sigma}_d^{1/2}\mathbf{R}\boldsymbol{\Sigma}_d^{-1/2} \\ \mathbf{C} + \mathbf{B}\boldsymbol{\Sigma}_d^{1/2}\mathbf{R}\boldsymbol{\Sigma}_d^{-1/2} \end{pmatrix}$$

and

$$\boldsymbol{\Delta}_{\text{marg}} = \begin{pmatrix} \mathbf{I} - \mathbf{S}_d^2 & \mathbf{0} \\ \mathbf{0} & \mathbf{I} - \mathbf{S}_d^2 \end{pmatrix} + \begin{pmatrix} \mathbf{S}_d\mathbf{C} \\ \mathbf{S}_d\mathbf{B} \end{pmatrix} \left(\boldsymbol{\Sigma}_d - \boldsymbol{\Sigma}_d^{1/2}\mathbf{R}_d^2\boldsymbol{\Sigma}_d^{1/2} \right) \begin{pmatrix} \mathbf{C}\mathbf{S}_d & \mathbf{B}\mathbf{S}_d \end{pmatrix},$$

where $\mathbf{B}$ and $\mathbf{C}$ are such that

$$\begin{pmatrix} \mathbf{B} & \mathbf{C} \\ \mathbf{C} & \mathbf{B} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\Sigma}_d & \boldsymbol{\Sigma}_d^{1/2}\mathbf{R}\boldsymbol{\Sigma}_d^{1/2} \\ \boldsymbol{\Sigma}_d^{1/2}\mathbf{R}\boldsymbol{\Sigma}_d^{1/2} & \boldsymbol{\Sigma}_d \end{pmatrix}^{-1/2}.$$

This law is slightly different than a $\text{CSN}^*(\boldsymbol{\mu}_d, \boldsymbol{\Sigma}_d, \mathbf{S}_d)$, in particular for high absolute values in $\mathbf{R}$ and in the off-diagonal elements of $\boldsymbol{\Sigma}_d$.

Conditional distribution

Let us consider two consecutive days in the same weather state. Applying the result in Appendix B yields

$$\mathbf{Y}_{d+1} \mid \mathbf{Y}_d \sim \text{CSN}_{k,2k}(\boldsymbol{\mu}_{\text{cond}}, \boldsymbol{\Sigma}_{\text{cond}}, \mathbf{D}_{\text{cond}}, \boldsymbol{\nu}_{\text{cond}}, \boldsymbol{\Delta}), \tag{8}$$

with

$$\boldsymbol{\mu}_{\text{cond}} = \boldsymbol{\mu}_{d+1} + \boldsymbol{\Sigma}_{d+1}^{1/2}\mathbf{R}\boldsymbol{\Sigma}_{d}^{-1/2}(\mathbf{Y}_d - \boldsymbol{\mu}_d), \quad \boldsymbol{\Sigma}_{\text{cond}} = \boldsymbol{\Sigma}_{d+1} - \boldsymbol{\Sigma}_{d+1}^{1/2}(\mathbf{I}_k - \mathbf{R}^2)\boldsymbol{\Sigma}_{d+1}^{1/2},$$

$$\mathbf{D}_{\text{cond}} = \begin{pmatrix} \mathbf{S}_d\mathbf{C} \\ \mathbf{S}_{d+1}\mathbf{B} \end{pmatrix}, \quad \boldsymbol{\nu}_{\text{cond}} = \begin{pmatrix} \mathbf{S}_d(\mathbf{B} + \mathbf{C}\boldsymbol{\Sigma}_{d+1}^{1/2}\mathbf{R}\boldsymbol{\Sigma}_{d}^{-1/2}) \\ \mathbf{S}_{d+1}(\mathbf{C} + \mathbf{B}\boldsymbol{\Sigma}_{d+1}^{1/2}\mathbf{R}\boldsymbol{\Sigma}_{d}^{-1/2}) \end{pmatrix} (\boldsymbol{\mu}_d - \mathbf{Y}_d),$$

and

$$\boldsymbol{\Delta} = \begin{pmatrix} \mathbf{I} - \mathbf{S}_d^2 & \mathbf{0} \\ \mathbf{0} & \mathbf{I} - \mathbf{S}_{d+1}^2 \end{pmatrix},$$

where $\mathbf{B}$ and $\mathbf{C}$ are such that

$$\begin{pmatrix} \mathbf{B} & \mathbf{C} \\ \mathbf{C} & \mathbf{B} \end{pmatrix} = \begin{pmatrix} \boldsymbol{\Sigma}_d & \boldsymbol{\Sigma}_d^{1/2}\mathbf{R}\boldsymbol{\Sigma}_{d+1}^{1/2} \\ \boldsymbol{\Sigma}_{d+1}^{1/2}\mathbf{R}\boldsymbol{\Sigma}_{d}^{1/2} & \boldsymbol{\Sigma}_{d+1} \end{pmatrix}^{-1/2}.$$

9.4 Estimation of the parameters

Estimation of single day marginal distribution

Concerning the inference of the marginal CSN parameters, the weighted moment method approach proposed in Flecher *et al.* (2009) is used. The first, second and weighted moments of a random variable $\mathbf{Y} \sim \text{CSN}_k^*(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{S})$ are:

$$\mathbb{E}[\mathbf{Y}] = \boldsymbol{\mu} + \frac{2}{\sqrt{2\pi}}\mathbf{S}\boldsymbol{\Sigma}^{1/2}\mathbf{1}_k, \quad \mathbb{V}ar(\mathbf{Y}) = \boldsymbol{\Sigma} - \frac{2}{\pi}\boldsymbol{\Sigma}^{1/2}\mathbf{S}^2\boldsymbol{\Sigma}^{1/2} \tag{9}$$

and

$$\mathbb{E}[\Phi_k(\mathbf{Y}, \mathbf{0}, \mathbf{I}_k)] = 2^k\Phi_{2k}\left(\mathbf{0}; \begin{bmatrix} -\boldsymbol{\mu} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \boldsymbol{\Sigma} + \mathbf{I}_k & \mathbf{S}\boldsymbol{\Sigma}^{1/2} \\ \boldsymbol{\Sigma}^{1/2}\mathbf{S} & \mathbf{I}_k \end{bmatrix}, \right). \tag{10}$$

Since we wish to estimate the parameters of the whole temporal model, we shall consider the full model described in Eq. (5), i.e. $\mathbf{Y}$ is the concatenated vector $(\mathbf{Y}_d', \mathbf{Y}_{d+1}')'$, $\boldsymbol{\Sigma}$ is the block matrix described in Eq. (5), and $\mathbf{S}$ is the concatenated block diagonal matrix.

For each weather state w, the corresponding experimental moments are computed according to

$$\mathbf{M}_1 = \bar{\mathbf{Y}}_w = \frac{\sum_d \mathbf{Y}(d)z_w(d)}{\sum_d z_w(d)}; \quad \mathbf{M}_2 = \frac{\sum_d (\mathbf{Y}(d) - \bar{\mathbf{Y}}_w)(\mathbf{Y}(d) - \bar{\mathbf{Y}}_w)'z(d)}{\sum_d z(d)}$$

and

$$M_0 = \frac{\sum_d \Phi_h(\mathbf{Y}(d); \mathbf{0}, \mathbf{I}_k)z_w(d)}{\sum_d z_w(d)},$$

whether the clustering is soft ($z \in]0,1[$) or hard ($z \in \{0,1\}$). For the temporal correlation, one computes

$$\mathbf{M}_2^{d,d+1} = \frac{\sum_d (\mathbf{Y}(d) - \bar{\mathbf{Y}})(\mathbf{Y}(d+1) - \bar{\mathbf{Y}})'z_w(d)z_w(d+1)}{\sum_d z_w(d)z_w(d+1)}$$

with

$$\bar{\mathbf{Y}} = \frac{\sum_d \mathbf{Y}(d)z_w(d)z_w(d+1)}{\sum_d z_w(d)z_w(d+1)}.$$

The estimation procedure coded in the function `estim.csnstar` uses a profile likelihood approach. For a given skewness matrix $\mathbf{S}$, the method of moment (MOM) estimates of $\boldsymbol{\Sigma}$ and $\boldsymbol{\mu}$ are

$$\hat{\boldsymbol{\Sigma}}^{1/2} = \mathbf{M}_2^{1/2}\left(\mathbf{I} - \frac{2}{\pi}\mathbf{S}^2\right)^{1/2} \quad ; \quad \hat{\boldsymbol{\mu}} = \mathbf{M}_1 - \sqrt{\frac{2}{\pi}}\mathbf{S}\hat{\boldsymbol{\Sigma}}^{1/2}\mathbf{1}_k. \tag{11}$$

The strategy is thus to find the elements of $\mathbf{S}$ minimizing the quantity

$$\left| M_0 - 2^k \Phi_{2k}\left(\mathbf{0}; \begin{bmatrix} -\hat{\boldsymbol{\mu}} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \hat{\boldsymbol{\Sigma}} + \mathbf{I}_k & \mathbf{S}\hat{\boldsymbol{\Sigma}}^{1/2} \\ \hat{\boldsymbol{\Sigma}}^{1/2}\mathbf{S} & \mathbf{I}_k \end{bmatrix}\right)\right|.$$

Estimation of lag-1 temporal correlation

The lag-1 temporal correlation coefficients in the diagonal matrix $\mathbf{R}$ are estimated conditional on the estimated values $\hat{\mathbf{S}}$ and $\hat{\boldsymbol{\Sigma}}$ obtained above. Again, a method of moment is used. The temporal second moments are computed as follows. For a given weather state w, we compute

$$\mathbf{M}_2^{d,d+1} = \frac{\sum_d (\mathbf{Y}(d) - \bar{\mathbf{Y}})(\mathbf{Y}(d+1) - \bar{\mathbf{Y}})' z_w(d) z_w(d+1)}{\sum_d z_w(d) z_w(d+1)}$$

with

$$\bar{\mathbf{Y}} = \frac{\sum_d \mathbf{Y}(d) z_w(d) z_w(d+1)}{\sum_d z_w(d) z_w(d+1)}.$$

9.5 Simulation

Non conditional simulation of two consecutive days

Direct application of Appendix C yields the following algorithm

1. Simulate $\mathbf{U}$ and $\mathbf{V}$ two i.i.d. $(0,1)$ Gaussian random vectors.
2. Compute

$$\mathbf{Y} = \boldsymbol{\mu} + \begin{pmatrix} \boldsymbol{\Sigma}_d & \boldsymbol{\Sigma}_d^{1/2}\mathbf{R}\boldsymbol{\Sigma}_{d+1}^{1/2} \\ \boldsymbol{\Sigma}_{d+1}^{1/2}\mathbf{R}\boldsymbol{\Sigma}_d^{1/2} & \boldsymbol{\Sigma}_{d+1} \end{pmatrix}^{1/2} \cdot \left[\begin{pmatrix} \mathbf{S}_d & \mathbf{0} \\ \mathbf{0} & \mathbf{S}_{d+1} \end{pmatrix} |\mathbf{U}| + \begin{pmatrix} \mathbf{I}_k - \mathbf{S}_d^2 & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_k - \mathbf{S}_{d+1}^2 \end{pmatrix}^{1/2} \mathbf{V} \right],$$

where $|\mathbf{U}|$ indicates the component-wise absolute value of $\mathbf{U}$.

Conditional simulation of $\mathbf{Y}_{d+1} \mid \mathbf{Y}_d$

The conditional distribution of $\mathbf{Y}_{d+1}$ given $\mid \mathbf{Y}_d$ is given in Eq. (8). Its simulation is performed by calling the general algorithm described in Appendix C. Care must be taken when the number of variables is different in $\mathbf{Y}_{d+1}$ and $\mid \mathbf{Y}_d$.

Part III
Implementation details

10 Preparing the data

10.1 The function `WACSdata`

Data are prepared by calling

```
myWACSdata = WACSdata(mydata, mapping=NULL, bounds=NULL, from="1960-01-01",
                      to="1979-12-31", skip=NULL, Trange=FALSE,
                      seasons=season.limits)
```

The arguments of this function are:

- `mydata` is the array containing the data organized in columns (see Section 2).
- `mapping`
- `bounds` is a list of of lists indicating the bounds for some, or all variables. If `bounds=NULL`, the bounds are set automatically according to the data. Some variables will have minimal values set automatically to 0 (`trange, V, RG, ETPP`) and maximum values at 100 (`ETPP`). Other minimum (resp. maximum) values are computed by adding (resp. subtracting) to the maximum (resp. minimum value) its difference to the 10th largest (resp. lowest) value.
- `from` and `to`: beginning and end of the selected period.
- `skip`: any variables to be skipped
- `Trange` is a Boolean variable. If `Trange = True`, the variables `(tmin,tmax)` are transformed into `(tmin,trange = tmax-tmin)`. `Trange = FALSE`, there is no transformation. Default is `Tminmax = False`.
- `seasons`, which contains the first day of each season (there can be any number of seasons).

10.2 The class `'WACSdata'`

The call

```
myWACSdata = WACSdata(mydata, mapping=NULL, bounds=NULL, from="1960-01-01",
                      to="1979-12-31", skip=NULL, Trange=FALSE,
                      seasons=season.limits)
```

produces a structure belonging to the class 'WACSdata', which is a list of 5 elements. The first element, myWACSdata$data is the data array. It is organized as follows:

```
> str(myWACSdata)
$ data   :'data.frame': 7300 obs. of  10 variables:
  ..$ year  : int [1:7300] 1960 1960 1960 1960 1960 1960 1960 1960 1960 1960 ...
  ..$ month : int [1:7300] 1 1 1 1 1 1 1 1 1 1 ...
  ..$ day   : int [1:7300] 1 2 3 4 5 6 7 8 9 10 ...
  ..$ season: num [1:7300] 1 1 1 1 1 1 1 1 1 1 ...
  ..$ rain  : num [1:7300] 0.243 1.105 2.097 0 0 ...
  ..$ tmin  : num [1:7300] 0.76263 2.6656 2.5839 3.3493 0.00564 ...
  ..$ tmax  : num [1:7300] 7.5 5.27 8.08 7.7 6.6 ...
  ..$ RG    : num [1:7300] 4.48 1.91 2.47 7.62 7.25 ...
  ..$ V     : num [1:7300] 3.93 3.7 2.74 5.98 3.27 ...
  ..$ ETPP  : num [1:7300] 85.1 86.2 89.3 74.6 77.1 ...
 $ mapping:'data.frame': 9 obs. of  2 variables:
  ..$ names_data: Factor w/ 9 levels "day","ETPP","month",..: 9 3 1 4 7 6 5 8 2
  ..$ wacs_names: Factor w/ 9 levels "day","ETPP","month",..: 9 3 1 4 7 6 5 8 2
 $ bounds :'data.frame': 2 obs. of  6 variables:
  ..$ rain: num [1:2] 0 108
  ..$ tmin: num [1:2] -18.9 20.5
  ..$ tmax: num [1:2] -12.1 32.7
  ..$ RG  : num [1:2] 0 35.6
  ..$ V   : num [1:2] 0 14.6
  ..$ ETPP: num [1:2] 0 100
 $ seasons:'data.frame': 4 obs. of  2 variables:
  ..$ month: num [1:4] 3 6 9 12
  ..$ day  : num [1:4] 1 1 1 1
 $ Trange : logi FALSE
 - attr(*, "class")= chr "WACSdata"
```

11 Estimating the parameters

11.1 Algorithmic description of WACSestim

The function WACSestim performs the estimation of all parameter. It performs the following tasks:

1. The function wacs.estimcycle estimates a smoothed version of the central tendency and a smoothed version of a measure of deviation by use of the function smooth.spline. Two options are possible: mean + standard deviation (trend.norm = "L2") or median + absolute deviation (trend.norm = "L1"). The parameters controlling the smoothing are spar and trend.norm.

2. The function wacs.estimrain estimates the parameters of the precipitation distribution for each season and computes the vector of Gaussian scores. Two models are available: a Gamma distribution (rain.model = "Gamma") and the transformed truncated

Gaussian distribution (`rain.model = "AB"`; not yet implemented). There is also an option `rain.model =''None''`, which makes no transformation on the rain data. For the Gamma distribution only, two estimation methods are possible: Maximum Likelihood (`method="MLE"`) and method of moments (`method="MOM"`).

3. For each season, s:

   (a) The function `wacs.estimWT` does a clustering of dry and wet days, thereby defining weather types and creating the vectors of probabilities `z`. Variables on which the clustering is determined are indicated in the vector `Vsel`. The optimal number of clusters is found within the vector `Nclusters`. Clustering can either be `hard` or `soft`.

   (b) On the basis of this clustering, the function `wacs.estimMarkov` estimates the transition matrix.

   (c) For each weather state, w (k is the number of variables in weather state w):

      i. The parameters of the CSN^*_{2k} (skewness $\mathbf{S}_w$, covariance $\mathbf{\Sigma}_w$ and location parameters $\boldsymbol{\mu}_w$) are estimated by calling the function `wacs.estimCSNstar`. This function calls `csnPWM` and `WMdiff` for performing the optimization.

      ii. The temporal correlation $\mathbf{R}_w$ are estimated.

If `plot.it = TRUE`, plots are produced and saved in the directory `DIR`. Otherwise no plot is produced.

11.2 The function `WACSestim`

A typical call to `WACSestim` is

```
myParam = WACSestim(wacsdata=myWACSdata,spar=0.7,trend.norm="L2",
                    rain.model="Gamma",method="MOM", Vsel=c(1,2,3),
                    Nclusters=c(1,2,3),clustering="hard",plot.it=FALSE,
                    DIR="./WACSdir")
```

The arguments of this function are:

- `wacsdata` is the data on which parameters are estimated, as retutned by the function `WACSdata`. It must belong to the class `'WACSdata'`.
- `spar` is the smoothing parameter for estimating annual cycle. Default is `spar=0.7`.
- `trend.norm` specifies the type of norm used in for computing central tendency and variation. Must be either `"L1"` or `"L2"` (default, recommended).
- `rain.model` specifies the model used for as distribution for precipitation. Must be either `"Gamma"`, `"AB"` (not yet implemented) or `"None"`. The latter makes no transformation.
- `method` specifies the estimation method used for estimating the parameters of the model for precipitation. Must be either `"MLE"` (default, recommended) or `"MOM"`.

- `Vsel` specifies the variables used for the clustering algorithm. Default is `Vsel=NULL`, in which case all variables are used.
- `Nclusters` specifies the number of clusters to consider in the clustering algorithm. When `Nclusters = NULL` (default), absolute best clustering is sought for wet and dry weather states in each season (up to 4).
- `clustering` indicates whether "hard or "soft" clustering is considered. Must be either `"soft"` (default, recommended) or `"hard"`.
- `plot.it` is a boolean indicating whether plots are produced (`plot.it=TRUE`), or not (`plot.it=FALSE`).
- `DIR` Directory in which storing plots. Default is `DIR =''./''`

11.3 The class `'WACSestim'`

A call to `WACSestim` produces an object, say `myParam` of the class `'WACSestim'`. It is a list containing 7 elements plus 1 list per season. In our example, it is thus a list of 11 elements.

```
> str(myParam)
List of 11
 $ bounds  :'data.frame': 2 obs. of  6 variables:
  ..$ rain: num [1:2] 0 108
  ..$ tmin: num [1:2] -18.9 20.5
  ..$ tmax: num [1:2] -12.1 32.7
  ..$ RG  : num [1:2] 0 35.6
  ..$ V   : num [1:2] 0 14.6
  ..$ ETPP: num [1:2] 0 100
 $ mapping :'data.frame': 9 obs. of  2 variables:
  ..$ names_data: Factor w/ 9 levels "day","ETPP","month",..: 9 3 1 4 7 6 5
  ..$ wacs_names: Factor w/ 9 levels "day","ETPP","month",..: 9 3 1 4 7 6 5
 $ Trend   :List of 3
  ..$ Param    : chr [1:2] "0.7" "L2"
  ..$ Central  : num [1:365, 1:5] -2.32 -2.34 -2.36 -2.38 -2.39 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:5] "tmin" "tmax" "RG" "V" ...
  ..$ Deviation: num [1:365, 1:5] 3.93 3.93 3.93 3.93 3.93 ...
  .. ..- attr(*, "dimnames")=List of 2
  .. .. ..$ : NULL
  .. .. ..$ : chr [1:5] "tmin" "tmax" "RG" "V" ...
 $ Rain    :List of 2
  ..$ RainModel: chr "Gamma"
  ..$ RainPar  : num [1:4, 1:2] 11.706 10.079 11.475 15.686 0.633 ...
 $ seasons :'data.frame': 4 obs. of  2 variables:
  ..$ month: num [1:4] 3 6 9 12
  ..$ day  : num [1:4] 1 1 1 1
```

```
 $ Trange  : logi FALSE
 $ varnames: chr [1:6] "rain" "tmin" "tmax" "RG" ...
 $ Season_1: List of 7
....
```

The list for season #1 contains the following elements. There is one similar list per season.

```
> str(myParam$Season_1)
List of 7
  ..$ NumbDays: num 90
  ..$ NumbWT  : int [1:2] 2 2
  ..$ TransM  : num [1:4, 1:4] 0.517 0.269 0.165 0.1 0.188 ...
  ..$ W1      :List of 4
  .. ..$ loc : num [1:5] -0.516 0.574 0.447 -1.448 -0.111
  .. ..$ cov : num [1:5, 1:5] 0.8586 0.2567 -0.335 0.1875 0.0833 ...
  .. ..$ skew: num [1:5] 0.07026 0.05704 0.01129 0.98 -0.00899
  .. ..$ rho : num [1:5] 0.81 0.689 0.365 0.121 0.54
  ..$ W2      :List of 4
  .. ..$ loc : num [1:5] -0.33 -0.318 0.859 -0.489 -0.883
  .. ..$ cov : num [1:5, 1:5] 1.123 0.98 -0.179 0.26 0.114 ...
  .. ..$ skew: num [1:5] 0.02029 0.00878 0.0043 0.86357 0.00701
  .. ..$ rho : num [1:5] 0.817 0.807 0.272 0.382 0.656
  ..$ W3      :List of 4
  .. ..$ loc : num [1:6] -0.4065 0.1763 -0.0573 -0.6246 -1.3474 ...
  .. ..$ cov : num [1:6, 1:6] 0.6067 0.1688 0.0861 -0.1207 0.0303 ...
  .. ..$ skew: num [1:6] 0.02158 0.00475 0.01381 0.00111 0.98 ...
  .. ..$ rho : num [1:6] 0.211 0.627 0.781 -0.23 0.452 ...
  ..$ W4      :List of 4
  .. ..$ loc : num [1:6] 0.684 0.542 -0.217 -1.321 -0.987 ...
  .. ..$ cov : num [1:6, 1:6] 0.83802 0.03344 -0.00766 0.01025 0.51505 ...
  .. ..$ skew: num [1:6] -0.0149 0.0116 0.0217 0.9045 0.9387 ...
  .. ..$ rho : num [1:6] 0.349 0.545 0.663 -0.95 0.369 ...
```

12 Simulating data with `WACSsimul`

12.1 Algorithmic description of the function `WACSsimul`

The general scheme of `WACSsimul` consists in simulating the succession of weather states according to a Markov Chain with a different transition matrix for each season. Then, conditionally on the weather states and on the residuals of the previous day, new residuals are simulated for the current day, under the condition that the corresponding variables verify the bounds in `bounds`. This can be done either by rejecting any vector of residuals at day `d` not respecting the bounds (option `REJECT = TRUE`), or by simply replacing all values outside the bounds by the minimum or maximum value. The variable `first.day` can be used to force the weather variables to som given values. `first.day` must be a vector with the same structure as the weather variables values in `myWACSdata` (i.e. from column 5 to column $4 + Nv$, where Nv is the number of weather variables. The detailed algorithm is:

1. Actually `WACSsimul` calls an internal function called `wacs.simul_innercall`. If the simulation is longer than 40 years, the inner function is called as many times as necessary by steps of 20 years. At the end of the loop, simulations are concatenated in order to produce the long series.

2. The first day (i.e. `from`)

   (a) Find the corresponding season

   `If (first.day = NULL)`

   i. A random weather state is drawn according to the limiting distribution corresponding to the transition matrix of the current season.

   ii. A random vector of residuals is drawn, from the CSN_k^* distribution corresponding to the simulated weather state, by calling `rmcsnstar`

   iii. The precipitation residual (if any) is transformed by calling `transform.rain`. When `rain.model=''None''`, the function `transform.rain` is called but makes no transformation.

   iv. Other residuals are transformed into climate variables using the trend parameters stored in the parameter file.

   v. Check if the corresponding variables are within the bounds in `bounds` by calling the function `testvariables`. If not and `REJECT=TRUE`, iterate steps (ii.) to (v.) until an appropriate vector is drawn. If not and `REJECT = FALSE`, the residuals outside the bounds are forced to the bounds by calling `boundsvariables`.

   `else`

   i. Set the variables of the first day to `first.day`

   ii. Compute the corresponding residuals, including for rain.

   iii. Set the most likely weather type of the season corresponding to `first.day` by calling `map.wt`

3. For $d = 2, ..., 365 * \texttt{Ny}$:

   (a) Draw a new weather state according to the transition matrix of the current season, by calling `rMarkov`.

   (b) Build the positive definite matrix

$$\mathbf{\Sigma} = \begin{pmatrix} \mathbf{\Sigma}_d & \mathbf{\Sigma}_d^{1/2}\mathbf{R}\mathbf{\Sigma}_{d+1}^{1/2} \\ \mathbf{\Sigma}_d^{1/2}\mathbf{R}\mathbf{\Sigma}_{d+1}^{1/2} & \mathbf{\Sigma}_{d+1} \end{pmatrix}.$$

   (c) Draw a random conditional k-vector of the CSN_{2k}^* in Eq. (5), as explained in Section 9.5, by calling `rmcsn.cond`.

   (d) Check if these random variables are within the bounds in `bounds` by calling the function `testvariables`. If not and `REJECT=TRUE`, iterate steps points (b) to (d) until an appropriate vector is drawn. If not and `REJECT = FALSE`, the residuals outside the bounds are forced to the bounds by calling `boundsvariables`.

   (e) If after 50 iterations a conditional vector cannot be simulated, a marginal random vector satisfying the bounds is drawn.

4. The output belongs to the class `'WACSsimul'`.

Note that when entering a new season, the transition matrix and all parameters of the CSN* distributions change. A weather state corresponding to the parameters of the new season must be assigned to the residual simulated the previous day (with the parameters corresponding to the previous season). This is done by calling `map.wt` which finds the most likely weather state in the new season.

12.2 The function `WACSsimul`

A typical call to `WACSsimul` is

```
mySimul = WACSsimul(wacspar=myParam, from, to, first.day=NULL, REJECT=FALSE)
```

The arguments of this function are:

- `wacspar` is the file containing the parameters of the WACS model, as estimated and returned by `WACSestim`. This file must belong to the class `'WACSestim'`
- `from` and `to`: beginning and end of the simulation period.
- `first.day` is used to force the weather variables to som given values. `first.day` must be a vector with the same structure as the weather variables values in `myWACSdata`. Default is `first.day=NULL`
- `REJECT` is the Boolean indicating whether a rejection technique is used to guarantee variables within bounds. Default is `REJECT=FALSE`. In this case, values outside bounds are forced to the bounds.

12.3 The class `'WACSsimul'`

A call to `WACSsimul` produces an object, say `mySimul`, of the class `'WACSsimul'`. It is a list of one single element, which is an array having the same structure as the data array `myWACSdata$data`.

```
> str(mySimul)
List of 1
 $ sim:'data.frame': 7305 obs. of  11 variables:
  ..$ year  : num [1:7305] 1960 1960 1960 1960 1960 1960 1960 1960 1960 1960 ...
  ..$ month : num [1:7305] 1 1 1 1 1 1 1 1 1 1 ...
  ..$ day   : num [1:7305] 1 2 3 4 5 6 7 8 9 10 ...
  ..$ season: num [1:7305] 1 1 1 1 1 1 1 1 1 1 ...
  ..$ WT    : num [1:7305] 1 1 1 1 2 1 3 2 1 3 ...
  ..$ rain  : num [1:7305] 0 0 0 0 0 ...
  ..$ tmin  : num [1:7305] -8.31 -4.15 -2.59 -5.25 -9.19 ...
  ..$ tmax  : num [1:7305] 8.12 8.87 8.68 2.71 -3.13 ...
  ..$ RG    : num [1:7305] 6.39 6.28 4.52 2.07 5.93 ...
  ..$ V     : num [1:7305] 0.376 1.126 1.148 1.419 2.861 ...
  ..$ ETPP  : num [1:7305] 77.7 85.9 92.4 100 72.6 ...
 - attr(*, "class")= chr "WACSsimul"
```

Appendices

A: Marginal law of a CSN distribution

The marginal distribution of a CSN vector $\mathbf{Y}$. We shall use the Theorem 1 in Gonzàlez-Farías, Domínguez-Molina and Gupta (2004) Let $\mathbf{Y}$ be a $\mathrm{CSN}_{k,l}(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{D}, \boldsymbol{\nu}, \boldsymbol{\Delta})$ and $\mathbf{A}$ be an $k_1 \times k$ ($k_1 \leq k$) matrix of rank k_1. Then,

$$\mathbf{AY} \sim \mathrm{CSN}_{k_1,l}(\boldsymbol{\mu}_\mathbf{A}, \boldsymbol{\Sigma}_\mathbf{A}, \mathbf{D}_\mathbf{A}, \boldsymbol{\nu}, \boldsymbol{\Delta}_\mathbf{A})$$

with

$$\boldsymbol{\mu}_\mathbf{A} = \mathbf{A}\boldsymbol{\mu}; \;\; \boldsymbol{\Sigma}_\mathbf{A} = \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}'; \;\; \mathbf{D}_\mathbf{A} = \mathbf{D}\boldsymbol{\Sigma}\mathbf{A}'\boldsymbol{\Sigma}_\mathbf{A}^{-1}$$

and

$$\boldsymbol{\Delta}_\mathbf{A} = \boldsymbol{\Delta} + \mathbf{D}\boldsymbol{\Sigma}\mathbf{D}' - \mathbf{D}_\mathbf{A}\boldsymbol{\Sigma}_\mathbf{A}\mathbf{D}_\mathbf{A}'.$$

Let us apply this lemma to the case $k_1 \times k$ matrix

$$\mathbf{A} = (\mathbf{I}_{k_1} \;\; \mathbf{0}),$$

for which $\mathbf{AY} = \mathbf{Y}_1$ is the marginal distribution of $\mathbf{Y}$ with the k_1 first coordinates of $\mathbf{Y}$. Then

$$\boldsymbol{\mu}_\mathbf{A} = \boldsymbol{\mu}_1; \;\; \boldsymbol{\Sigma}_\mathbf{A} = \boldsymbol{\Sigma}_{11}; \;\; \mathbf{D}_\mathbf{A} = (\mathbf{D}_1 + \mathbf{D}_2\boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1})$$

and

$$\boldsymbol{\Delta}_\mathbf{A} = \boldsymbol{\Delta} + \mathbf{D}_2(\boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12})\mathbf{D}_2',$$

where

$$\boldsymbol{\mu} = \begin{pmatrix} \boldsymbol{\mu}_1 \\ \boldsymbol{\mu}_2 \end{pmatrix}, \quad \boldsymbol{\Sigma} = \begin{pmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{pmatrix}, \text{ and } \mathbf{D} = (\mathbf{D}_1 \; \mathbf{D}_2).$$

B: Conditional law of a CSN distribution

Let us consider that $\mathbf{Y} \sim \mathrm{CSN}_{p,n}(\boldsymbol{\mu}, \boldsymbol{\Sigma}, \mathbf{D}, \boldsymbol{\nu}, \boldsymbol{\Delta})$ is decomposed into $\mathbf{Y}_1$ and $\mathbf{Y}_2$ according to $\mathbf{Y}' = (\mathbf{Y}_1', \mathbf{Y}_2')'$. Let us denote $(\boldsymbol{\mu}_1', \boldsymbol{\mu}_2')'$ the similar decomposition of the mean vector and

$$\begin{pmatrix} \boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12} \\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22} \end{pmatrix} \text{ and } (\mathbf{D}_1 \;\; \mathbf{D}_2)$$

the associated block decomposition of $\boldsymbol{\Sigma}$ and $\mathbf{D}$. The dimension of $\mathbf{Y}$ is p and the dimension of $\mathbf{Y}_2$ is $p_2 \leq p$.
According to Proposition 1 in Karimi and Mohammadzadeh (2012) and Proposition 16 in Domínguez-Molina, Gonzàlez-Farías and Gupta (2003) the conditional distribution of $\mathbf{Y}_2 \mid \mathbf{Y}_1$ is a CSN distribution:

$$\mathbf{Y}_2 \mid \mathbf{Y}_1 \sim \mathrm{CSN}_{p_2,n}(\boldsymbol{\mu}_{2|1}, \boldsymbol{\Sigma}_{2|1}, \mathbf{D}_2, \boldsymbol{\nu}_{2|1}, \boldsymbol{\Delta})$$

with

$$\boldsymbol{\mu}_{2|1} = \boldsymbol{\mu}_2 + \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}(\mathbf{Y}_1 - \boldsymbol{\mu}_1); \quad \boldsymbol{\Sigma}_{2|1} = \boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12},$$

$$\boldsymbol{\nu}_{2|1} = \boldsymbol{\nu} - (\mathbf{D}_1 + \mathbf{D}_2\boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1})(\mathbf{Y}_1 - \boldsymbol{\mu}_1).$$

C: Simulation algorithm for a CSN*

We use the stochastic decomposition of a CNS distribution (see Allard and Naveau, 2007). Let use define $\mathbf{F} = \mathbf{\Sigma D' Q}^{-1} = \mathbf{\Sigma}^{1/2}\mathbf{S Q}^{-1}$ and $\mathbf{G} = (\mathbf{\Sigma} - \mathbf{\Sigma D' Q}^{-1}\mathbf{D\Sigma})^{1/2}$ with $\mathbf{Q} = \mathbf{\Delta} + \mathbf{D\Sigma D'}$. Then, the vector

$$\boldsymbol{\mu} + \mathbf{F}\mathbf{U}_{\mathbf{U}\geq\boldsymbol{\nu}} + \mathbf{G}\mathbf{V} \tag{12}$$

is distributed as a $\text{CSN}_{p,q}(\boldsymbol{\mu}, \mathbf{\Sigma}, \mathbf{D}, \boldsymbol{\nu}, \mathbf{\Delta})$, where $\mathbf{U} \sim \mathcal{N}_q(\mathbf{0}, \mathbf{Q})$ and $\mathbf{V} \sim \mathcal{N}_p(\mathbf{0}, \mathbf{I})$ independent of $\mathbf{U}$, and where $\mathbf{U} \geq \mathbf{0}$ indicates that for each component $U_i \geq 0$. Simulation of $\mathbf{U}_{\mathbf{U}\geq\mathbf{0}}$ can be achieved by a direct rejection algorithm if q is small; otherwise it can be achieved with a Gibbs Sampling algorithm. In `R`, it is achieved by calling the function `rtmvnorm` of the package `tmvtnorm`. The simulation algorithm is thus the following:

1. Compute $\mathbf{Q} = \mathbf{\Delta} + \mathbf{D\Sigma D'}$.
2. Simulate $\mathbf{U}_{|\mathbf{U}\geq\boldsymbol{\nu}}$ where $\mathbf{U} \sim \mathcal{N}_q(\mathbf{0}, \mathbf{Q})$ by calling `rtmvnorm` of the package `tmvtnorm`.
3. Simulate $\mathbf{V} \sim \mathcal{N}_p(\mathbf{0}, \mathbf{I})$
4. Compute $\mathbf{F} = \mathbf{\Sigma D' Q}^{-1}$ and $\mathbf{G} = (\mathbf{\Sigma} - \mathbf{\Sigma D' Q}^{-1}\mathbf{D\Sigma})^{1/2} = (\mathbf{\Sigma} - \mathbf{FD\Sigma})^{1/2}$.
5. Return $\mathbf{Y} = \boldsymbol{\mu} + \mathbf{F}\mathbf{U}_{\mathbf{U}\geq\boldsymbol{\nu}} + \mathbf{G}\mathbf{V}$

For the special case of a CSN*, $\mathbf{Q}$ reduced to $\mathbf{I}_k$ and $\boldsymbol{\nu} = \mathbf{0}$. Hence $\mathbf{F} = \mathbf{\Sigma}^{1/2}\mathbf{S}$ and

$$\begin{aligned}
\mathbf{G} &= [\mathbf{\Sigma} - \mathbf{\Sigma}^{1/2}\mathbf{S}\mathbf{S}\mathbf{\Sigma}^{1/2}]^{1/2} \\
&= [\mathbf{\Sigma}^{1/2}(\mathbf{I} - \mathbf{S}^2)\mathbf{\Sigma}^{1/2}]^{1/2} \\
&= \mathbf{\Sigma}^{1/2}(\mathbf{I} - \mathbf{S}^2)^{1/2}.
\end{aligned}$$

Then,

$$\mathbf{Y} = \boldsymbol{\mu} + \mathbf{\Sigma}^{1/2}\left[\mathbf{S}\mathbf{U}_{\mathbf{U}\geq\mathbf{0}} + (\mathbf{I}_k - \mathbf{S}^2)^{1/2}\mathbf{V}\right] \tag{13}$$

is distributed as a $\text{CSN}^*(\boldsymbol{\mu}, \mathbf{\Sigma}, \mathbf{S})$, with $\mathbf{U} \sim \mathbf{V} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_k)$. Since the truncated Gaussian vector has independent component, simulation of a CSN* is now much easier. It does not necessitates calling a MCMC algorithm. Note that Eq. (13) proves that if $\mathbf{Y} \sim \text{CSN}^*(\boldsymbol{\mu}, \mathbf{\Sigma}, \mathbf{S})$, then the vector $\tilde{\mathbf{Y}} = \Sigma^{-1/2}(\mathbf{Y} - \boldsymbol{\mu})$ is distributed as a $\text{CSN}^*(\mathbf{0}, \mathbf{I}_k, \mathbf{S})$.

References

Allard, D. and Naveau, P. (2007) A new spatial skew-normal random field model. *Communication in Statistics*, **36**, 1821–1834.

Domínguez-Molina, Gonzàlez-Farías and Gupta, 1.K. (2003) The multivariate Closed Skew-Normal Distribution. Tech. Report, Department of Statistics, Bowling Green university.

Flecher, C., Naveau P., Allard D. and Brisson, N. (2010) A Stochastic Daily Weather Generator for Skewed Data, Water Resource Research, 46, W07519.

Flecher, C., Naveau, P. and Allard, D. (2009) Estimating the closed skew-normal distributions parameters using weighted moments. *Statistic and Probability letters*, **79**, 1977–1984.

Fraley, C. and Raftery, A.E. (2003) Enhanced model-based clustering, density estimation, and discriminant analysis software: MCLUST; *Journal of Classification*, **, 20**, 263-286.

Fraley, C., Raftery, A.E., Murphu, T.B. and Scrucca, L (2012) MCLUST Version 4 for R: Normal Mixture Modeling for Model-Based Clustering, Classification and density Estimation. Technical Report no. 597, Departement of Statistics, University of Washington.

Genton, M.G. (Ed.) (2004) *Skew-Elliptical Distributions and Their Applications: A Journey Beyond Normality*. Boca Raton, FL: Chapman & Hall/CRC.

González-Farías, G.,Domínguez-Molina, J. and Gupta, A. (2004) Additive properties of skew-normal random vectors. *Journal of Statistical Planning and Inference*, **126**, 521–534.