Tutorial for using hydroPSO to calibrate TUWmodel

Study area: Trancura River Basin, Chile

Mauricio Zambrano-Bigiarini[*] Oscar M. Baez-Villanueva[†]

version 0.9.3, March 2020

[*]Universidad de La Frontera, Temuco, Chile. e-mail:mauricio.zambrano@ufrontera.cl
[†]TH Köln, Cologne, Germany, e-mail:obaezvil@th-koeln.de

Contents

1 A bit of history

When `hydroPSO` (Zambrano-Bigiarini and Rojas 2013) started to be developed in 2010, to the best of our knowledge, `topmodel` was the only hydrological model fully implemented as an R function and available at CRAN. However, for some reason I never got into it and preferred other R-external models (e.g., SWAT). Therefore, the development of `hydroPSO` was focused on optimising hydrological/environmental models that runs an executable file from the command line of any operative system (OS, e.g., GNU/Linux, Windows, OSX), or to optimise R functions in a way compatible with the `optim` function, but global instead of local.

Since then, R has continuated to gain popularity in the hydrological community, and nowadays there are several hydrological models fully implmented as R functions (even if some of them internally call Fortran or C/C++ routines), such as `TUWmodel`, `airGR`, and `VICmodel`[1]

This tutorial will show how to use `hydroPSO >= 0.5-0` to calibrate the lumped `TUWmodel` hydrological model, using hydrometeorological input data included in the `hydroPSO` package. However, with some R knowledge you should be able to adapt this tutorial (e.g., section `Input data`) to calibrate your own catchment, and even other hydrological models written in R.

2 Citation

This tutorial is inspired in the original hydroPSO vignette (Rojas and Zambrano-Bigiarini 2012) and several scripts developed by Dr. Zambrano-Bigiarini while he was working at the Joint Research Center and afterwards. If you find it useful, please cite it as Zambrano-Bigiarini and Baez-Villanueva (2020):

> Zambrano-Bigiarini, Mauricio and Baez-Villanueva, Oscar M. (2020). Tutorial for interfacing hydroPSO with TUWmodel (version 0.9.3). doi:10.5281/zenodo.3772176. http://doi.org/10.5281/zenodo.3772176.

3 Required `hydroPSO` version

This tutorial was developed for the new `hydroPSO >= 0.5-0`, released in March 2020. Notwhithstanding `hydroPSO` has been always able to optimise any R function, it was not originally designed to calibrate R-based (hydrological/environmental) models. Therefore, versions lower than `0.5-0` only keep track of the goodness-of-fit values (and not the model simulations) when you calibrate an R-based model. However, since **v0.5-0** `hydroPSO` is fully compatible with R-based (hydrological/environmental) models.

4 Objective

The main objective of this tutorial is to show how to use `hydroPSO` to find an **acceptable** *best parameter set* for the `TUWmodel` hydrological model, rather than providing a description of hydrological concepts[2] (e.g., calibration, verification, sesnsitivity analysis, uncertainty analysis) or an in-depth discussion of the hydrological results. Therefore, all the figures generated in this tutorial are not analysed in this document, and their discussion is left to the interested reader. All comments and questions are very welcomed

[1]At the time of writting this tutorial, `VICmodel` has been removed from the CRAN repository on March 3rd, 2020.
[2]Some basic references are provided along this text to guide the reading of new hydrologists.

5 hydroPSO

`hydroPSO` is a multi-OS and model-independent package based on the Particle Swarm Optimisation technique (PSO; Kennedy and Eberhart 1995) designed to allow the user to perform: *i*) model calibration; *ii*) sensitivity analysis; and *iii*) an assessment of the calibration results. This package is fully compatible with calibration tools employing PEST-like files and allows parallelisation.

PSO is a population-based stochastic optimisation technique used to explore a delimited search space with a *swarm* of particles to find the best set of parameters required to maximise (or minimise) a user-defined defined objective function. The search space is explored based on individual and neighbourhood-based best-known particle positions starting with a random initialisation of the particles' position and velocities within the parameter space. The position and velocity of each particle are updated taking into consideration its actual values and the location of the best-known optimum in the neighbourhood. The positions and velocities of the particles evolve throughout iterations until a user-defined criterion is met (e.g., errors are lower than a threshold tolerance, or a maximum number of iterations are achieved). For a detailed description of the `hydroPSO` package please refer to Zambrano-Bigiarini and Rojas (2013) and Zambrano-Bigiarini, Clerc, and Rojas (2013).

6 TUWmodel

`TUWmodel` is a hydrologic model developed by the Technical University of Vienna (Viglione and Parajka 2019) that works at daily or hourly temporal scale and follows the structure of the HBV model (Bergström 1995). The simulation of the hydrological cycle includes snow accumulation, change of humidity in the soil profile, and surface flow throughout the drainage network. It was validated over 320 basins in Austria (Parajka, Merz, and Blöschl 2007), and it has been used in several other studies (e.g., Ceola et al. 2015; Sleziak et al. 2016, 2018, 2020; Parajka et al. 2016; Nijzink et al. 2016, 2018; Zessner et al. 2017; Cisty and Soldanova 2018; Melsen et al. 2018). Sleziak et al. (2016) evaluated `TUWmodel` in basins with different regimes, and reported that it shows good performance in watersheds with snow cover.

The parameters used by `TUWmodel` to represent the different hydrological processes of a basin are briefly described in Table 1. The range of values used to calibrate each of the parameters was taken from Viglione and Parajka (2019), which slightly modified the ranges proposed by Parajka, Merz, and Blöschl (2007).

ID	**Description**	**Units**	**Process**	**Range**
SCF	Snow correction factor	-	Snow	0.9 - 1.5
DDF	Degree-day factor	mm/°C/day	Snow	0.0 - 5.0
Tr	Temperature threshold above which precip. is rain	°C	Snow	1.0 - 3.0
Ts	Temperature threshold below which precip. is snow	°C	Snow	-3.0 - 1.0
Tm	Temperature threshold above which melt starts	°C	Snow	-2.0 - 2.0
LPrat	Parameter related to the limit for potential evaporation	-	Evaporation	0.0 - 1.0
FC	Field capacity	mm	Infiltration	0.0 - 600
BETA	Non-linear parameter for runoff production	-	Infiltration	0.0 - 20
cperc	Constant percolation rate	mm/day	Infiltration	0.0 - 8.0
k0	Storage coefficient for very fast response	day	Runoff	0.0 - 2.0
k1	Storage coefficient for fast response	day	Runoff	2.0 - 30
k2	Storage coefficient for slow response	day	Runoff	30 - 250
lsuz	Threshold storage state	mm	Runoff	1.0 - 100
bmax	Maximum base at low flows	day	Runoff	0.0 - 30
croute	Free scaling parameter	day^2/mm	Runoff	0.0 - 50

Table 1: Parameters used by *TUWmodel* to represent the different hydrological processes in a basin.

7 Installation

Installing the latest stable version of hydroPSO and TUWmodel.

```r
install.packages("hydroPSO")
install.packages("TUWmodel")
```

Alternatively, you can also try the under-development version of hydroPSO (from Github):

```r
if (!require(devtools)) install.packages("devtools")
library(devtools)
install_github("hzambran/hydroPSO")
```

Installing the latest stable version of hydroTSM and hydroGOF. These packages will be used to evaluate the performance of the simulatied streamflow.

```r
install.packages("hydroTSM")
install.packages("hydroGOF")
```

8 Model set up

8.1 Loading required packages

The hydroPSO and TUWmodel packages include all the data and functions used in this analysis. The hydroTSM (Zambrano-Bigiarini, Mauricio 2020) and hydroGOF (Zambrano-Bigiarini 2020) packages are used here to manipulate the hydrological time series and to compute the modified Kling-Gupta efficiency (KGE'; Gupta et al. 2009; Kling, Fuchs, and Paulin 2012), respectively.

```r
library(TUWmodel)
library(hydroPSO)
library(hydroTSM)
library(hydroGOF)
```

8.2 General settings

The variable model.drty will be used as the parent directory where all the output files generated during the calibration of the hydrological model will be stored.

```r
model.drty       <-"/home/hzambran/GIT/hydroPSO_vignette" # modify it according to your specific paths
setwd(model.drty)
```

The variable Figures.drty.out represents the directory that will store all the output figures generated during the analysis of the calibration and verification results.

```r
Figures.drty.out <- paste0(model.drty, "/Figures")

# If the output directory selected to store the figures does not exists, it is created:
if (!file.exists(Figures.drty.out)) dir.create(Figures.drty.out, recursive=TRUE)
```

Setting up the calibration (1979-1997) and verification (1998-2016) periods, each one of them include 1 year of warming up (see Section 9.2):

```r
### Calibration period
Cal.Ini <- "1980-01-01"
Cal.Fin <- "1997-12-31"

### Verification period
Ver.Ini <- "1999-01-01"
Ver.Fin <- "2016-12-31"

### Starting date of the warming up period, one for calibration and other for verification
Warmup.Ini.Cal <- "1979-01-01"
Warmup.Ini.Ver <- "1998-01-01"
```

8.3 Input data

The stuady area for this tutorial is a subcatchment of the Trancura River Basin, which is located in the Araucania Region in Southern Chile. In particular, our study area drains into the "Rio Trancura antes de Llafenco" streamflow station (COD.BNA:9414001).

In order to run `TUWmodel` you only need daily time series of precipitation, mean air temperature and potential evapotranpiration. Daily time series of precipitation (P [mm/day]), mean air temperature (Temp [°C]), potential evaporation (PET [mm/day]), and streamflow (Q [m^3/s]) from 1979-2016 were downloaded from the CAMELS-CL dataset (Alvarez-Garreton et al. 2018). In particular, precipitation and mean air temperature data correspond to spatially-averaged mean daily values (CR2met), while potential evapotranspiration values were computed using the Hargreaves-Samani equation based on the maximum and minimun air temperature data.

All the time series are provided in the new 0.5-0 version of the `hydroPSO` package, but of course you can use your own local data files and read them into R.

Specifying the catchment area, [m^2]:

```r
Area.m2 <- 1415025887
```

Specifying the name and ID of the discharge station used as outlet of the study area (they will be used afterwards, in the title of some figures during the analysis of the calibration results):

```r
Qobs.stationname <- "Rio Trancura antes de LLafenco"
Qobs.ID          <- "9414001"
```

Loading the daily meteorological input data (P, Temp, and PET) and the observed discharges at the outlet of the catchment as a single data.frame object, with values from 01-Jan-1979 to 31-Dec-2016:

```r
data(Trancura9414001)
```

Creating individual time series of P, Temp, PET and Qobs:

```r
dates <- Trancura9414001[, 1]
P     <- Trancura9414001[, 2]
Temp  <- Trancura9414001[, 3]
PET   <- Trancura9414001[, 4]
Qobs  <- Trancura9414001[, 5]
```

It is worth mentioning that to **run** `TUWmodel` you do not need values of observed discharges at the outlet of your catchment. However, to **calibrate** your model you must provide observed discharge values (with the same temporal frequency of the model simulations), in order to be able to compute the user-defined goodness-of-fit measure between your model simulations and their corresponding observed values.

Transforming the previously created numeric values and their correspoding dates into zoo objects:

```
dates    <- as.Date(dates)
P.zoo    <- zoo(P, dates)
Temp.zoo <- zoo(Temp, dates)
PET.zoo  <- zoo(PET, dates)
Qobs.zoo <- zoo(Qobs, dates)
```

Daily discharges simulated by `TUWmodel` are expressed in mm/day, but the previously loaded observed discharges are in m^3/s. Therefore, in order to correctly compare the observations (m^3/s) with the simulated values obtained from `TUWmodel` (in mm/day) it is necessary to convert the observed discharges from m^3/s to mm/day:

```
# 1 day = 86400 s
Qobs.zoo <- (Qobs.zoo * 1000 * 86400) / Area.m2
```

Plotting the full time series of meterological input data (`P`, `Temp`, `PET`) and the observed discharges (`Qobs`):

```
par(mfrow=c(4,1))
plot(P.zoo   , xlab="Time", col="lightblue", main="Precipitation"        , ylab="P [mm/day]")
grid()
plot(Temp.zoo, xlab="Time", col="red"      , main="Air temperature"      , ylab="Temp [degC]" )
grid()
plot(PET.zoo , xlab="Time", col="darkgreen", main="Potential evaporation", ylab="PET [mm/day]")
grid()
plot(Qobs.zoo, xlab="Time" , col="blue"    , main="Observed streamflows" , ylab="Q, [mm]" )
grid()
```

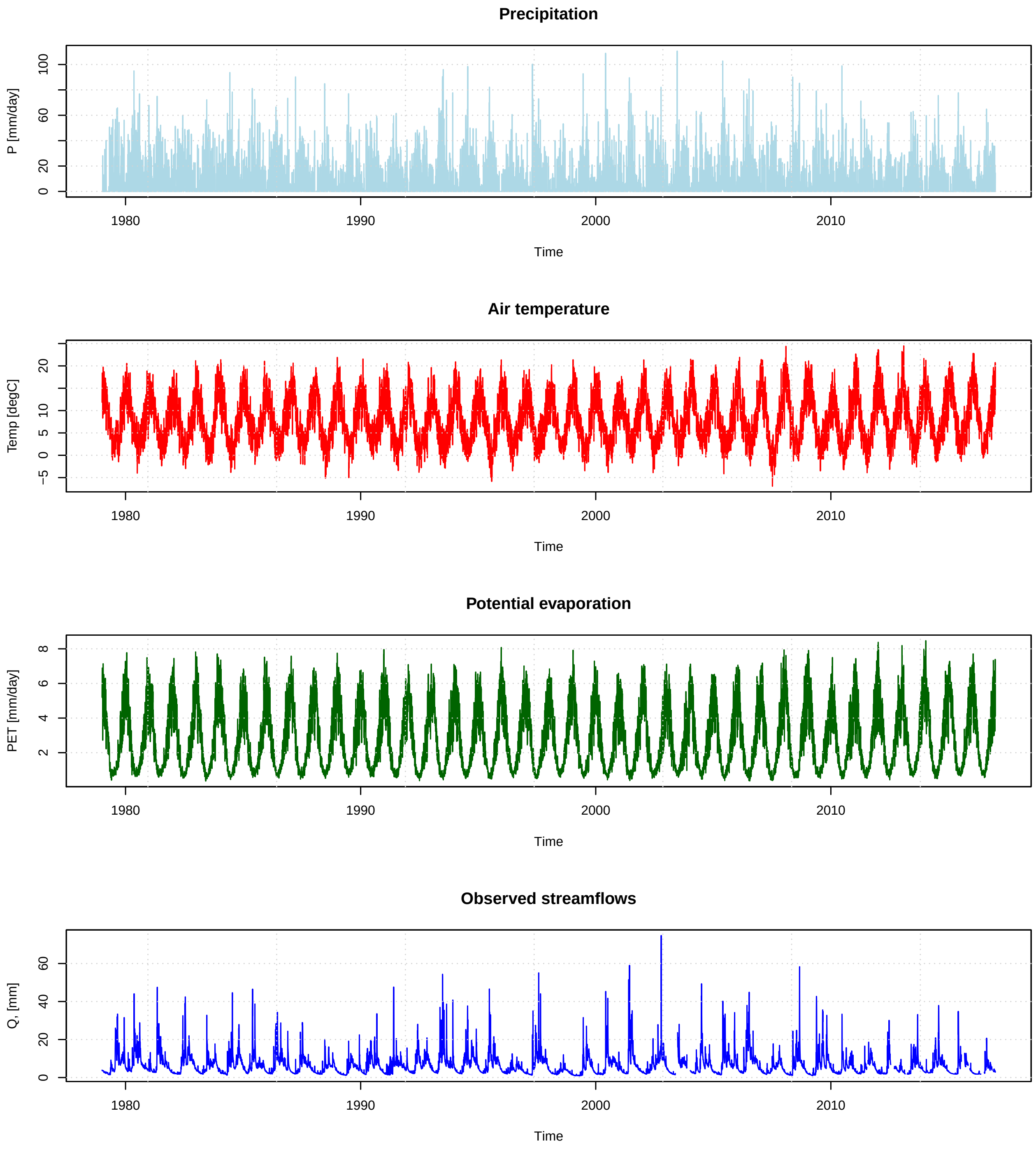


Figure 1: Daily values of precipitation, air temperature, potential evapotranspiration and streamflows for the Trancura River Basin during the 1970-2016 time period.

8.4 Model parameters

It is not a requirement for `hydroPSO`, but setting the names of the `TUWmodel` model parameters will make easier the interpretation of the calibration results. In addition, the lower and upper boundaries for each one of the parameters of the `TUWmodel` are defined following Viglione and Parajka (2019):

```
names <- c("SCF", "DDF", "Tr", "Ts", "Tm", "LPrat", "FC", "Beta",
           "k0", "k1", "k2", "lsuz", "cperc", "bmax", "croute")
lower <- c(0.9,   0.0,   1.0,  -3.0, -2.0,   0.0,     0,     0,
             0,     2,    30,     1,    0,     0,     0)
upper <- c(1.5,   5.0,   3.0,   1.0,  2.0,   1.0,   600,    20,
             2,    30,   250,   100,    8,    30,    50)

names(lower) <- names
names(upper) <- names
```

8.5 Running `TUWmodel`

Unfortunately, `TUWmodel` does not support zoo objects as input data, and it only works with plain numeric values. Therefore, we need to transform the input time series into numeric objects:

```
P    <- as.numeric(P.zoo)
Temp <- as.numeric(Temp.zoo)
PET  <- as.numeric(PET.zoo)
Qobs <- as.numeric(Qobs.zoo)
```

8.5.1 *Average* parameter set

Before any calibration of `TUWmodel` we must be sure that the model is able to run correctly with the previously prepared input data. As first guess, we will use as parameter set the *average* betweeen the lower and upper boundary of each model parameter:

```
params      <- lower + (upper - lower) / 2
modeloutput <- TUWmodel(prec=P,  airt=Temp, ep=PET, param=params)
```

The outuput given by `TUWmodel` has several components (surface, and subsurface flow, baseflow, actual evaporation, soil moisture, etc):

```
str(modeloutput)
```

```
## List of 22
##  $ itsteps: int 13880
##  $ nzones : int 1
##  $ area   : num 1
##  $ param  : num [1, 1:15] 1.2 2.5 2 -1 0 0.5 300 10 1 16 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:15] "SCF" "DDF" "Tr" "Ts" ...
##   ..- attr(*, "names")= chr [1:15] "SCF" "DDF" "Tr" "Ts" ...
##  $ incon  : num [1, 1:4] 50 0 2.5 2.5
##   ..- attr(*, "names")= chr [1:4] "SSM0" "SWE0" "SUZ0" "SLZ0"
##  $ prec   : num [1:13880] 0 0 0 0 0 0 0 0 0 0 ...
##  $ airt   : num [1:13880] 11.9 14.9 15 16.5 17.4 ...
##  $ ep     : num [1:13880] 5.2 5.98 6.04 6.89 6.66 6.77 6.35 5.93 5.39 5.1 ...
##  $ output : num [1, 1:20, 1:13880] 3.62e-04 0.00 4.83e+01 0.00 0.00 ...
##  $ qzones : num [1, 1:13880] 0.000362 0.001447 0.003247 0.005758 0.008975 ...
##  $ q      : num [1, 1:13880] 0.000362 0.001447 0.003247 0.005758 0.008975 ...
##  $ swe    : num [1, 1:13880] 0 0 0 0 0 0 0 0 0 0 ...
##  $ melt   : num [1, 1:13880] 0 0 0 0 0 0 0 0 0 0 ...
##  $ q0     : num [1, 1:13880] 0 0 0 0 0 0 0 0 0 0 ...
##  $ q1     : num [1, 1:13880] 0 0 0 0 0 0 0 0 0 0 ...
##  $ q2     : num [1, 1:13880] 0.0355 0.0352 0.035 0.0347 0.0345 ...
##  $ moist  : num [1, 1:13880] 48.3 46.3 44.5 42.4 40.5 ...
##  $ rain   : num [1, 1:13880] 0 0 0 0 0 0 0 0 0 0 ...
##  $ snow   : num [1, 1:13880] 0 0 0 0 0 0 0 0 0 0 ...
##  $ eta    : num [1, 1:13880] 1.73 1.92 1.87 2.04 1.88 ...
##  $ suz    : num [1, 1:13880] 0 0 0 0 0 0 0 0 0 0 ...
##  $ slz    : num [1, 1:13880] 4.96 4.93 4.89 4.86 4.83 ...
```

For this example, we are interested only in the surface streamflows, because they are comparable to the observed discharges previously loaded:

```
Qsim <- as.numeric(modeloutput$q)
```

To have better graphical comparisons, we will transform our simulated and observed values from numeric into zoo objects:

```
Qsim.zoo <- zoo(Qsim, dates)
Qobs.zoo <- zoo(Qobs, dates)
```

Graphical comparison of observed and simulated time series obtained with the *average* paramer set:

```
ggof(sim=Qsim.zoo, obs=Qobs.zoo, ylab="Q, [mm]", lty=c(1, 1), cex=0.4)
```

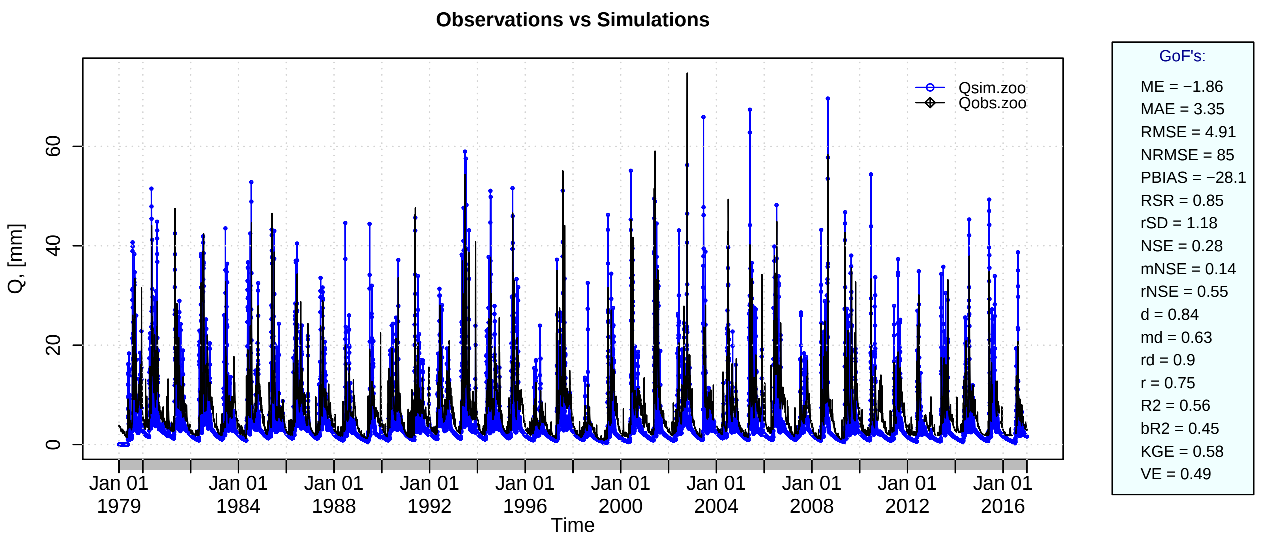


Figure 2: Evaluation of simulated streamflows for the calibration period using several performance indices.

Not too bad (except for low flows), but let's try the dafault parameter values suggested by Viglione and Parajka (2019) . . .

8.5.2 *Default* parameter set

As second guess, we will use as parameter set the *default* values for each model parameter, as defined in the manual of the TUWmodel R pacakge (Viglione and Parajka 2019):

```
params         <- c(1.2, 1.2,   2, -2,  0,  0.9,   100, 3.3,
                    0.5,   9, 105, 50,  2,   10,  26.5)
names(params) <- names

modeloutput <- TUWmodel(prec=P, airt=Temp, ep=PET, param=params )
```

Extracting only simulated streamflows from the full model output:

```
Qsim <- as.numeric(modeloutput$q)
```

To have better graphical comparisons, we will transform our simulated and observed values from numeric into zoo objects:

```
Qsim.zoo <- zoo(Qsim, dates)
Qobs.zoo <- zoo(Qobs, dates)
```

Graphical comparison of observed and simulated time series obtained with the *default* paramer set:

```
ggof( sim=Qsim.zoo, obs=Qobs.zoo, ylab="Q, [mm]", lty=c(1, 1), cex=0.4)
```

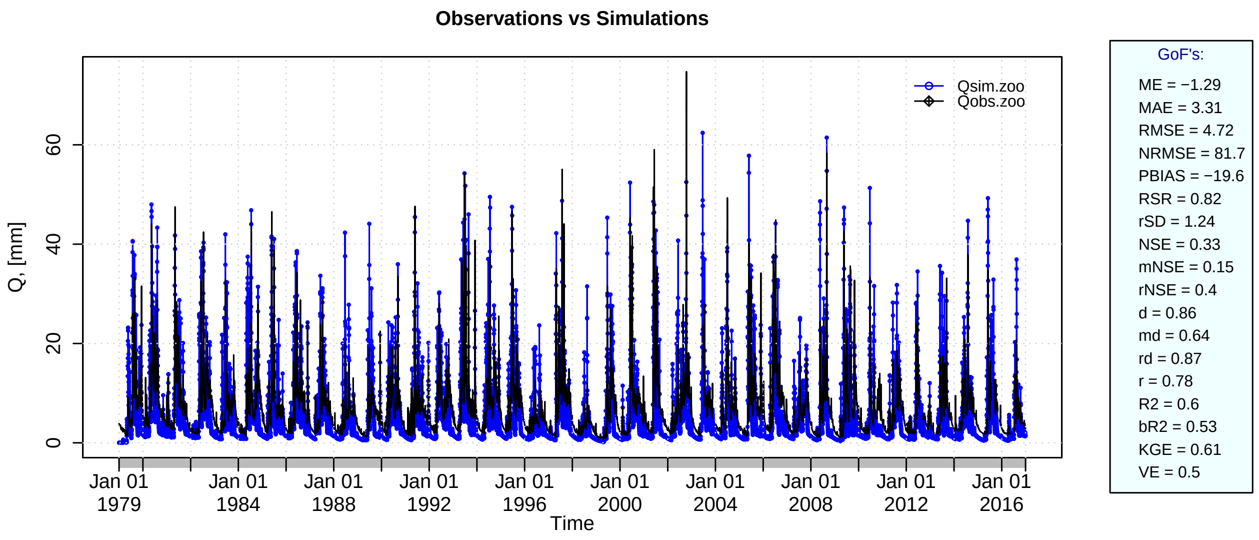


Figure 3: Evaluation of simulated streamflows for the calibration period using several performance indices.

Better than the previous paramter set, but here is still plenty of space for improving these simulated values. . .

9 Calibrating `TUWmodel` with `hydroPSO`

9.1 Subsetting for the calibration period

For the calibration of `TUWmodel` with `hydroPSO` we are going to use only a portion of the available input time series, leaving the rest for an independent verification period:

```
P.zoo.cal    <- window(P.zoo   , start=Warmup.Ini.Cal, end=Cal.Fin)
Temp.zoo.cal <- window(Temp.zoo, start=Warmup.Ini.Cal, end=Cal.Fin)
PET.zoo.cal  <- window(PET.zoo , start=Warmup.Ini.Cal, end=Cal.Fin)
Qobs.zoo.cal <- window(Qobs.zoo, start=Warmup.Ini.Cal, end=Cal.Fin)
```

Storing the dates of the input time series used to drive the model calibration (including the warming up period):

```
dates.cal <- time(P.zoo.cal)
```

As mentioned before, `TUWmodel` does not support zoo objects as input data, and it only works with plain numeric values. Therefore, we need to transform the input time series into numeric objects:

```
P.cal    <- as.numeric(P.zoo.cal)
Temp.cal <- as.numeric(Temp.zoo.cal)
PET.cal  <- as.numeric(PET.zoo.cal)
```

9.2 Wrapper R function implementing `TUWmodel` for `hydroPSO`

For the calibration of `TUWmodel` (or any other hydrological model implemented as an R function), you must build an R function that provides the outputs used by `hydroPSO` to move forward in the parameter space throughout the iterations. **This function MUST fulfill the following requirements**:

1) It first agument must be named **`param.values`**, which represents the numeric values of the parameters to be optimised.

2) One of its arguments must be named **`obs`**, which represents the observed values. `obs` are used to be compared against the simulated values delivered by the hydrological model (`obs`)

3) It must return a list object with -at least- the following elements:

   3.1) **GoF**: the goodness-of-fit of obtained when the hydrological model is run with `param.values` as parameter set.

   3.2) **sim**: model output obtained when the hydrological model is run with `param.values` as parameter set. It MUST have the same object class (and dimensions) than the observed values provided by the user in `obs`.

9.2.1 Basic function

An example of such a function for calibrating `TUWmodel` with `hydroPSO` is the following:

```r
TUWhydromod.basic <- function(param.values,
                              obs=Qobs.zoo.cal
                             ) {

  # Evaluating the hydrological model at the 'param.values' parameter set
  # P.cal, Temp.cal, PET.cal, and Area.m2 are used hee as *global variables*
  # (not good programming style, but it works)
  simLump  <- TUWmodel(param=param.values, prec=P.cal, airt=Temp.cal, ep=PET.cal, area=Area.m2)

  # Extracting only the numeric vector with simulations and observations
  qsim.full <- as.numeric(simLump$q)
  obs        <- as.numeric(obs)

  # Removing the warming up period (1 year, i.e., 365 days)
  n         <- length(qsim.full)
  qsim      <- qsim.full[366:n]
  obs       <- obs[366:n]

  # Computing the goodness-of-fit value (i.e, model performance)
  gof       <- KGE(sim=qsim, obs=obs, method="2012")

  # Creating the output of the R function
  nelements <- 2
  out        <- vector("list", nelements)
  out[[1]]   <- gof
  out[[2]]   <- qsim.full

  # Mandatory names for the elements of the output
  names(out)[1:nelements] <- c("GoF", "sim")

  return(out)

} # 'TUWhydromod.basic' end
```

9.2.2 Advanced funtion

The aforementioned `TUWhydromod` R function will allow you to calibrate your model without problems. However, it is a **good practice** to identify all the arguments used by your hydrological model to run properly, in order to have them correctly identified in the **hydroPSO_logfile.txt** output file. This log file might help when you are trying to remeber what you did many weeks/months/years after the calibration was done.

An example of such a better function for calibrating `TUWmodel` with `hydroPSO` is the following:

```
TUWhydromod <- function(param.values,      # it MUST be present
                        obs=Qobs.zoo.cal, # it MUST be present
                        dates=dates.cal,  # Optional
                        warmup.days=365,  # Optional, 1 year of warming up
                        P=P.cal,          # Optional
                        Temp=Temp.cal,    # Optional
                        PET=PET.cal,      # Optional
                        Area=Area.m2      # Optional
                        ) {

  # Evaluating the hydrological model at the 'param.values' parameter set
  simLump  <- TUWmodel(param=param.values, prec=P, airt=Temp, ep=PET, area=Area)

   # Extracting only the numeric vector with simulations and observa
  qsim <- as.numeric(simLump$q)

  # Converting model simulations into zoo objects
  qsim.zoo.full <- zoo(qsim, dates)

  # Removing the warming up period (1 year, i.e., 365 days)
  n        <- length(qsim.zoo.full)
  qsim.zoo <- qsim.zoo.full[(warmup.days+1):n]
  obs      <- obs[(warmup.days+1):n]

  # Computing the goodness-of-fit value (i.e, model performance)
  gof      <- KGE(sim=qsim.zoo, obs=obs, method="2012")

  # Creating the output of the R function
  nelements <- 2
  out        <- vector("list", nelements)
  out[[1]]  <- gof
  out[[2]]  <- qsim.zoo.full

  # Mandatory names for the elements of the output
  names(out)[1:nelements] <- c("GoF", "sim")

  return(out)

} # 'TUWhydromod' end
```

9.3 Running the calibration

9.3.1 Basic function

After defining the basic wrapper function for the hydrological model, you can calibrate *your* `TUWmodel` with `hydroPSO` as follows (this may take a couple of minutes!):

```
out <- hydroPSO(fn="hydromodInR",
                lower=lower,
                upper=upper,
                method="spso2011",
                control=list(write2disk=TRUE, MinMax="max", npart=80,
                             maxit=50, normalise=TRUE, REPORT=10, parallel="none",
                             reltol=1E-10),
                model.FUN="TUWhydromod.basic"
                )
```

9.3.2 Advanced funtion

You can calibrate your `TUWmodel` with `hydroPSO` using the advanced wrapper function defined in Section 9.2.2 (note the use of `model.FUN.args`) as follows (this may take a couple of minutes!):

```
out <- hydroPSO(fn="hydromodInR",
                lower=lower,
                upper=upper,
                method="spso2011",
                control=list(write2disk=TRUE, MinMax="max", npart=80,
                             maxit=50, normalise=TRUE, REPORT=10, parallel="none",
                             reltol=1E-10),
                model.FUN="TUWhydromod",
                model.FUN.args= list(obs=Qobs.zoo.cal,
                                     dates=dates.cal,
                                     warmup.days=365,
                                     P=P.cal, Temp=Temp.cal, PET=PET.cal, Area=Area.m2
                                     )
                )
```

In the previous code chunk, the arguments passed to `hydroPSO` are briefly explained below (more information can be found with `?hydroPSO`):

a) **fn**: character, used to tell `hydroPSO` that it will have to optimise an R-based hydrological model.

b) **lower**: numeric, representing the lower boundaries of each one of the model parameters.

c) **upper**: numeric, representing the upper boundaries of each one of the model parameters.

d) **method**: character, used to defined the PSO algorithm used to calibrate the hydrological model.

e) **model.FUN**: R function, used to run the hydrological model (delivering model simulations and their corresponding goodness-of-fit value), fulfilling the three (3) requirements mentioned in Section 9.2.

f) **model.FUN.args**: list object, contains all the arguments used to run the hydrological model (in addition to `param.values`).

g) **control**: list object, contains all the arguments used to fine-tune the calibration with `hydroPSO`. All the arguments passed to `control` will impact the way `hydroPSO` explore the parameter space and/or the outputs of the algorithm. However, you must pay particlar attention to the following three:

- *MinMax*: character, indicates whether the calibration will solve a maximization or minimization problem. Valid values are in: c('min', 'max'). Default value is "min".
- *npart*: numeric, number of particles in the swarm. By default npart=NA, which means that the swarm size depends on the value of method: `npart=ceiling(10+2*sqrt(n))`[3] when `method='spso2007'`, or `npart=40` otherwise.
- *maxit*: numeric, maximum number of iterations. By default maxit=1000.

The total number of model runs during the calibration is given by `npart*maxit`, and the **total execution time** of the optimisation will depend on the time taken by an individual model run times the total number of model runs.

Using `npart=40` should be good enough for most model applications. However, if the number of parameters of your model is ~10 or more, I suggest to **explore the use of a larger number of particles in combination with a lower number of model runs** (e.g., `npart=80` and `maxit=50`). The later ensures that `hydroPSO` will have enough number of particles to effectively tackle the *curse of dimensionallity* while exploring the parameter space.

9.4 Basic analysis of calibration results

Looking at the *best* parameter set obtained during the calibration and its corresponding goodness-of-fit value:

```
# 'best' parameter set obtained duirng calibration
out$par
```

```
##        SCF        DDF         Tr         Ts         Tm      LPrat         FC 
##  1.1888742  1.9172542  2.0122495 -2.2511459  0.2740785  0.9310345  3.1785837 
##       Beta         k0         k1         k2       lsuz      cperc       bmax 
##  8.5977524  0.2969852 29.3445921 47.0833073 28.6615290  6.3394072 29.6615193 
##     croute 
## 28.7451313
```

```
# goodness-of-fit obtained for the 'best' parameter set
out$value
```

```
## [1] 0.8509352
```

Saving the *best* parameter set obtained during the calibration:

```
best.param.cal <- out$par
```

Running `TUWmodel` with the *best* parameter set obtained during the calibration:

```
modeloutput <- TUWmodel(param=best.param.cal, prec=P.cal, airt=Temp.cal, ep=PET.cal)
```

Transforming the numeric output of the model into a zoo object:

```
Qsim.zoo.cal <- zoo(as.numeric(modeloutput$q), dates.cal)
```

Defining a customised title for all the following figures:

```
main <- paste0("TUWmodel: ", Qobs.stationname, " (", Qobs.ID, ")")
```

[3]`n` is the number of model parameters being calibrated.

Graphical comparison of the original daily observed and simulated time series obtained with the *best* paramer set for the calibration period (excluding the warming up period):

```r
Qsim.zoo.cal <- window(Qsim.zoo.cal, start=Cal.Ini)
Qobs.zoo.cal <- window(Qobs.zoo.cal, start=Cal.Ini)
ggof(sim=Qsim.zoo.cal, obs=Qobs.zoo.cal, main=main, ylab="Q, [mm]", cex=0.3, lty=c(1,1) )
```

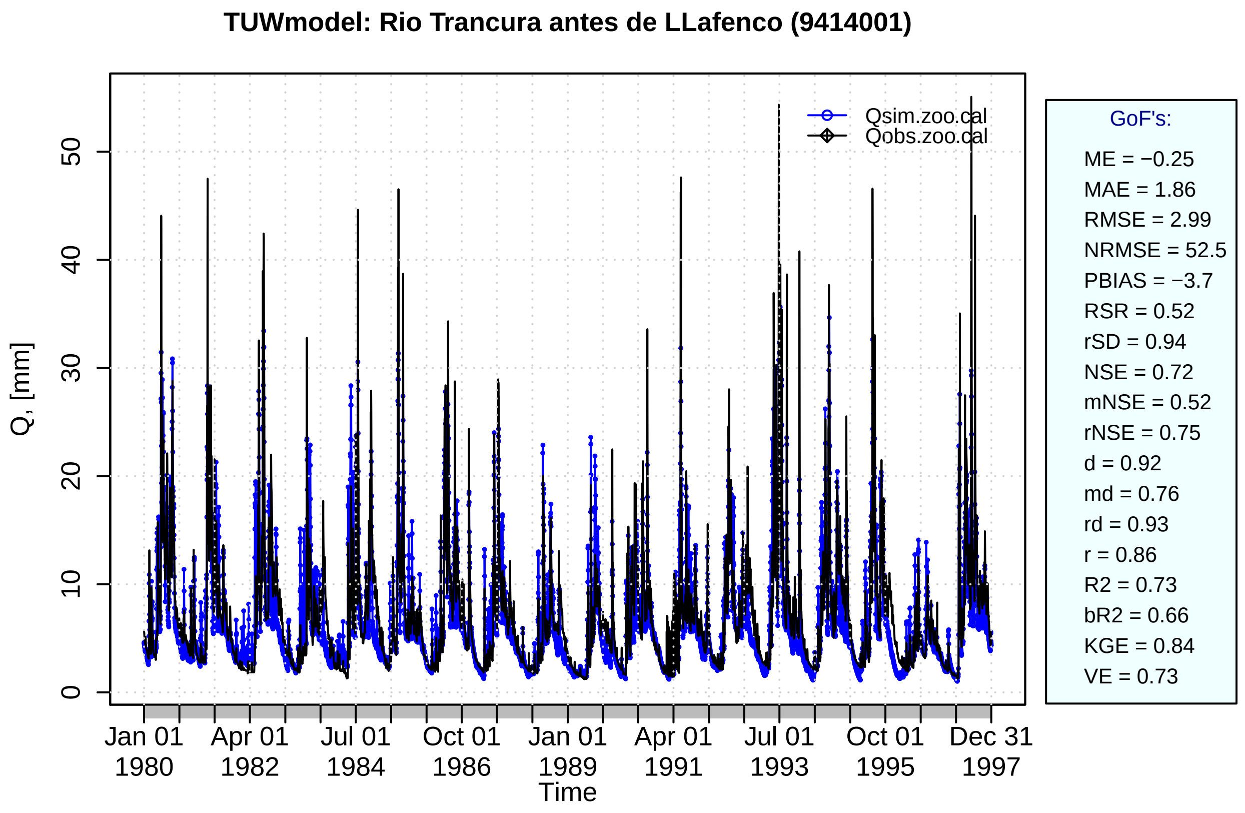


Figure 4: Graphical comparison of streamflows simulated by `TUWmodel` during the calibration period using several performance indices.

Much better than the results obtained with the average default parameter sets, right?

Graphical comparison of daily and monthly simulated (*Qsim.zoo.cal*) and observed (*Qobs.zoo.cal*) values for the calibration period:

```r
ggof(sim=Qsim.zoo.cal, obs=Qobs.zoo.cal, ftype="dm",
     FUN=mean, main=main, ylab="Q, [mm]", cex=0.4, lty=c(1,1) )
```

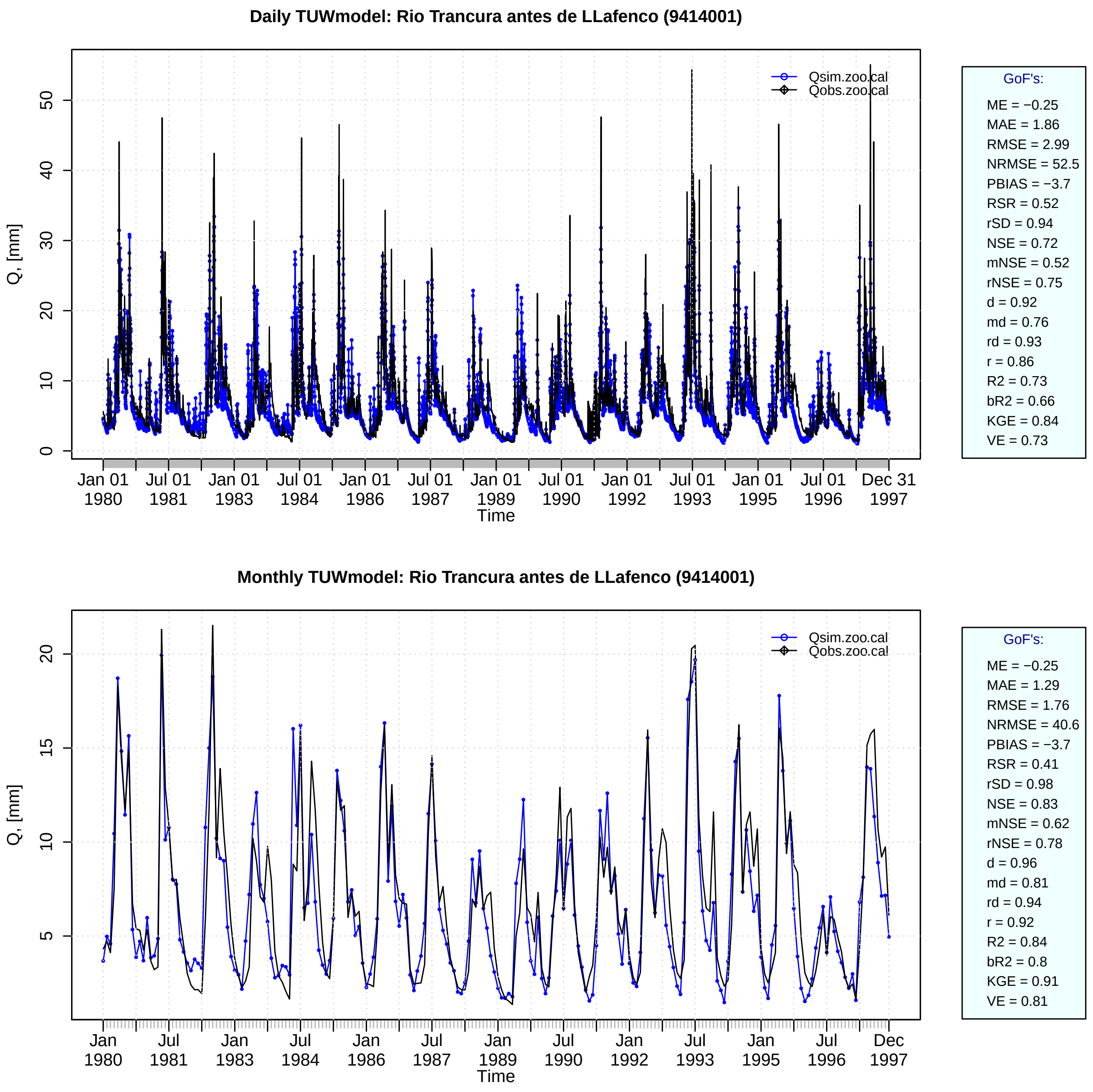


Figure 5: Daily and monthly evaluation of simulated streamflows for the calibration period using several performance indices.

Graphical comparison of monthly and annual simulated and observed values for the calibration period:

```
ggof(sim=Qsim.zoo.cal, obs=Qobs.zoo.cal, ftype="ma", pt.style="bar",
     FUN=mean, main=main, ylab="Q, [mm]", cex=0.4, lty=c(1,1) )
```

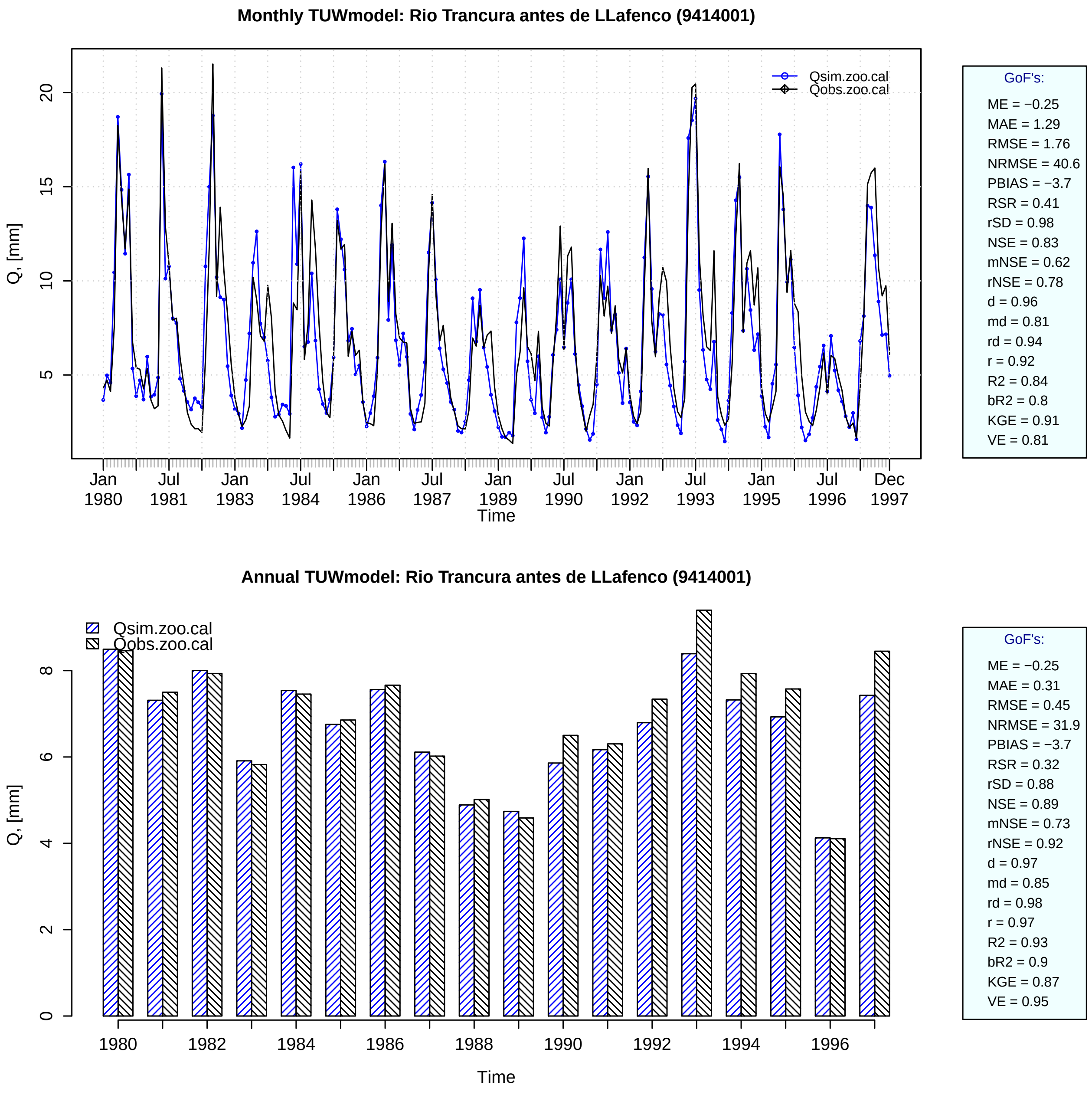


Figure 6: Monthly and annual evaluation of simulated streamflows for the calibration period using several performance indices.

Graphical comparison of seasonal simulated and observed values (one plot for each weather season) for the calibration period:

```
ggof(sim=Qsim.zoo.cal, obs=Qobs.zoo.cal, ftype="seasonal", ylab="Q, [mm]",
     season.names=c("Summer", "Autumn", "Winter", "Spring"),
     FUN=mean, main=main, cex.main=2)
```

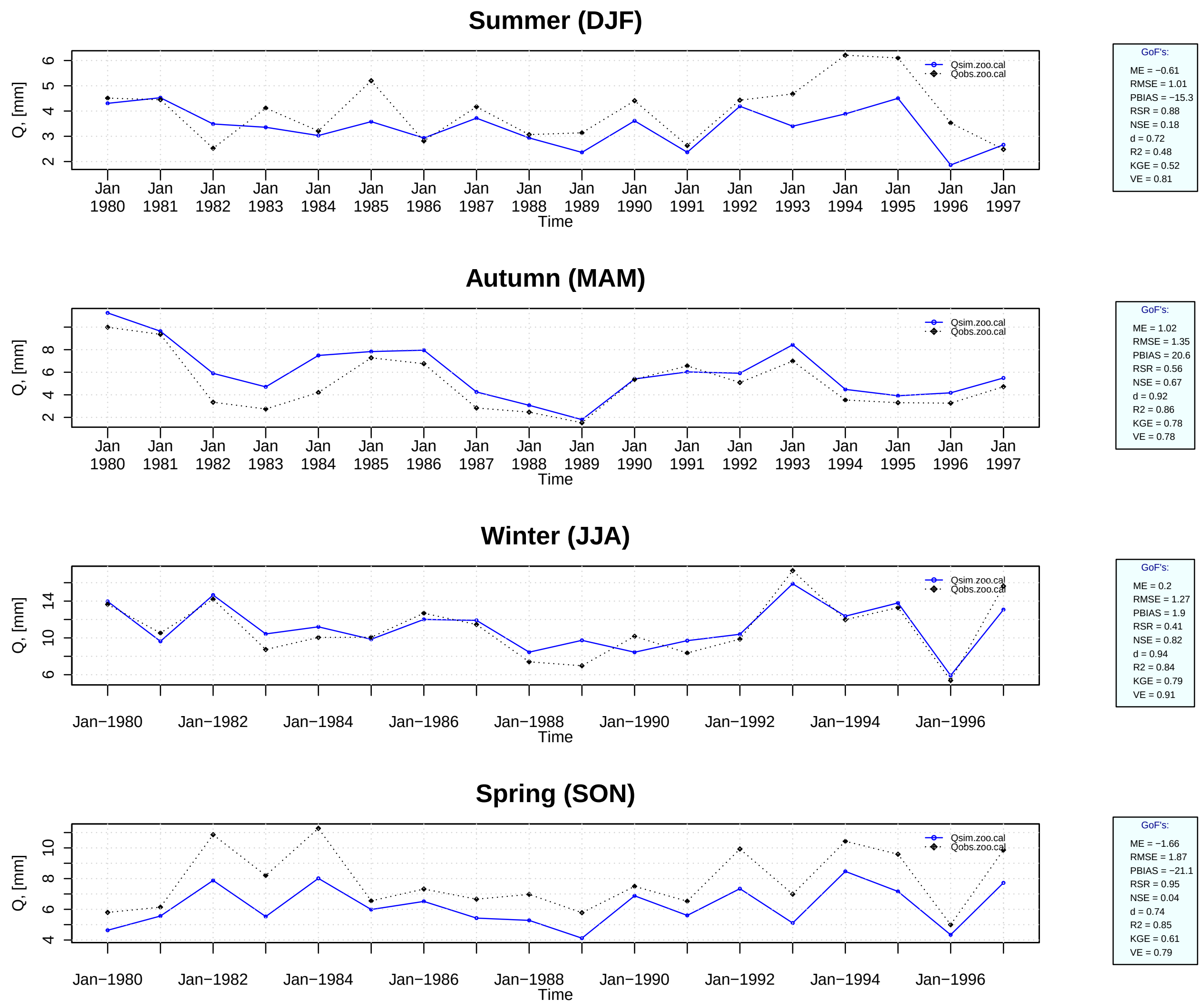


Figure 7: Seasonal evaluation of simulated streamflows for the calibration period using several performance indices.

Computing the daily residuals for the calibration period:

```
residuals <- Qsim.zoo.cal - Qobs.zoo.cal
```

Plotting daily, monthly and annual time series, boxplots and histograms of the residuals for the calibration period:

```
hydroplot(residuals, FUN=mean, var.unit="mm")
```

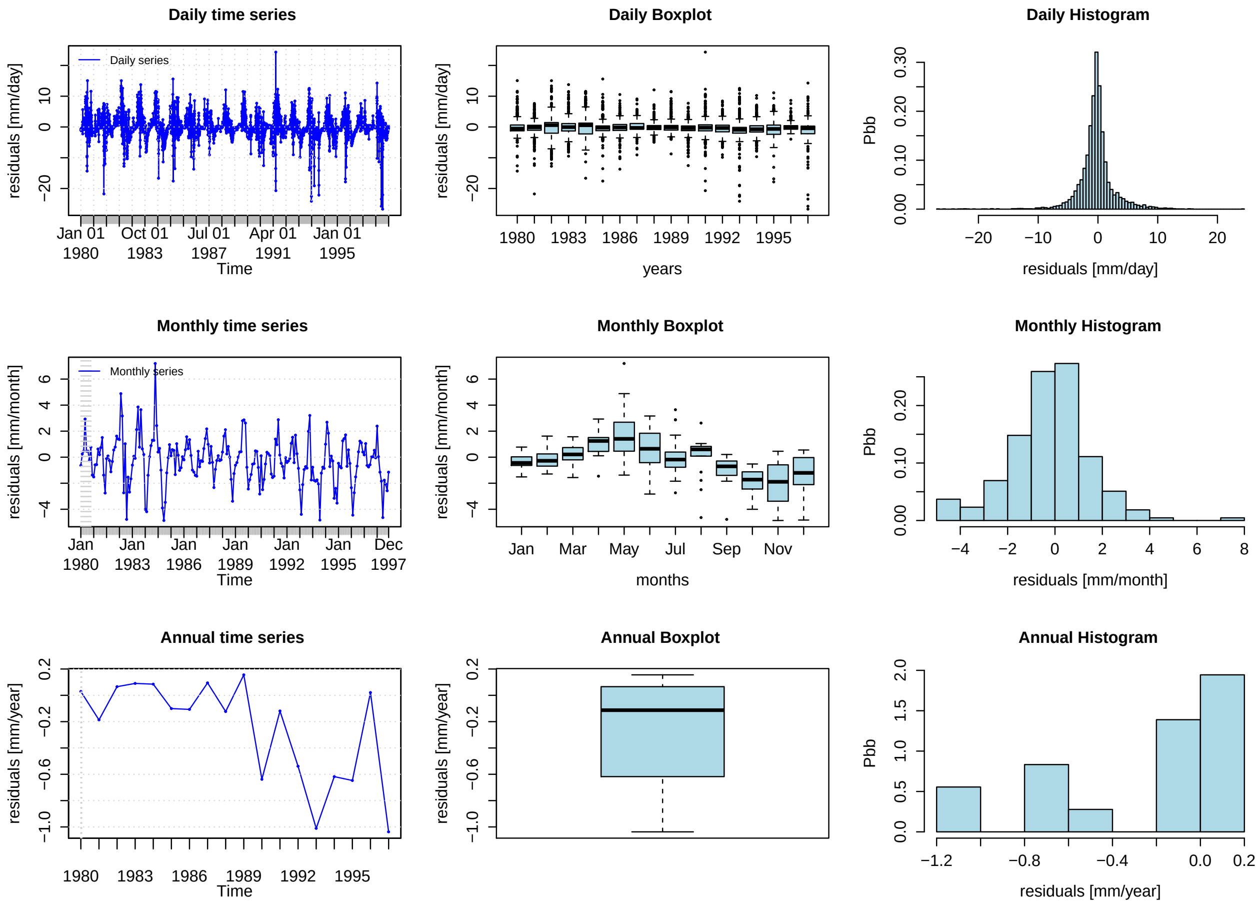


Figure 8: Analysis of the residuals (simulations - observations) for the calibration period at the daily, monthly, and annual temporal scales.

Seasonal time series and boxplots of the residuals for the calibration period:

```
hydroplot(residuals, FUN=mean, pfreq="seasonal",
          season.names=c("Summer", "Autumn", "Winter", "Spring"),
          h=0, main=main, cex.main=2)
```

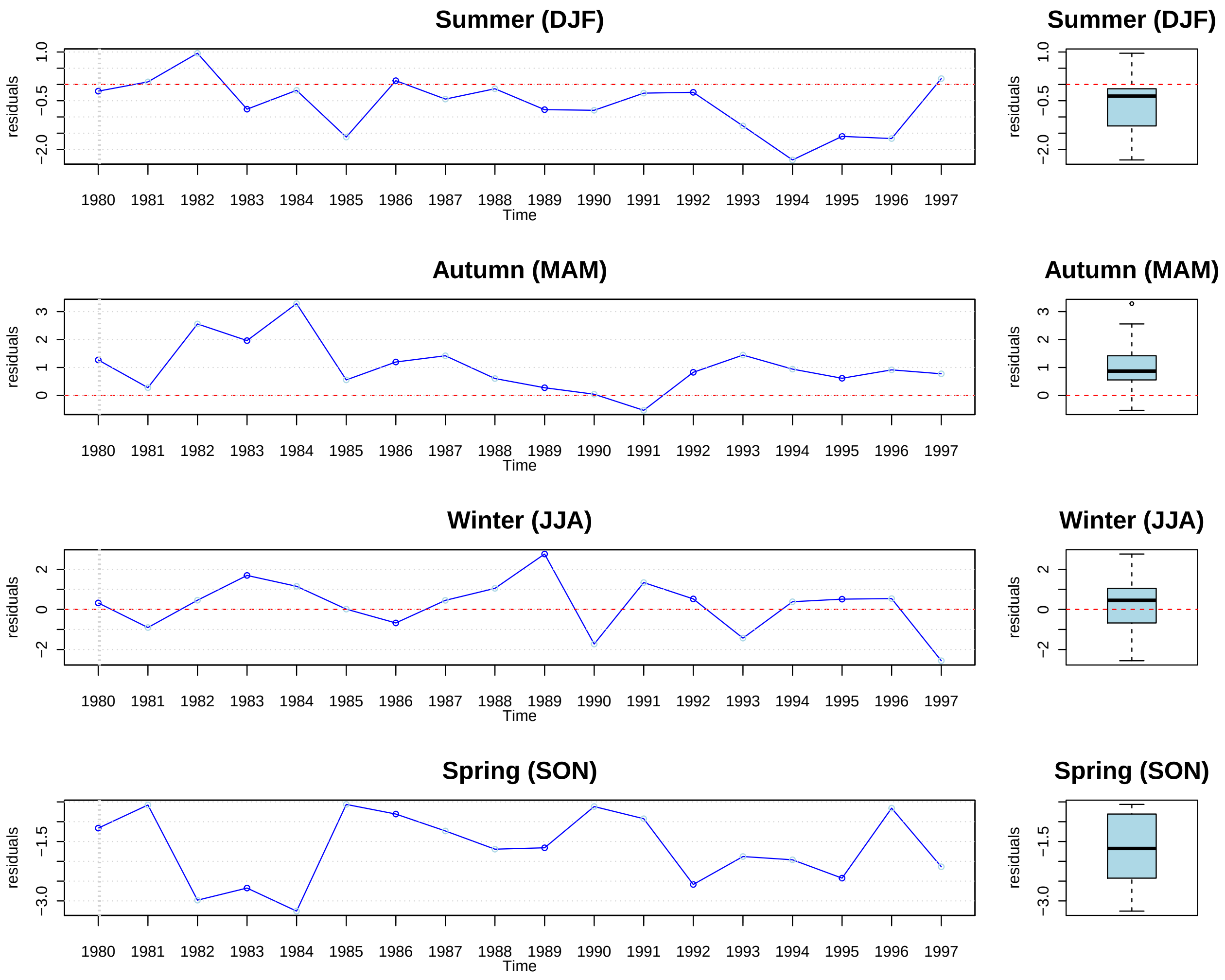


Figure 9: Analysis of the residuals (simulations - observations) for the calibration period at the seasonal scale.

9.5 Plotting calibration results

Customised title for all the figures in this section:

```r
main <- paste0("TUWmodel: ", Qobs.stationname, " (", Qobs.ID, ")")
```

9.5.1 Basic plotting

The function `plot_results` included in `hydroPSO` can automatically plot several figures that will help you to evaluate how good are the results of the calibration procedure. This function, internally calls the `read_results` function and then produces the following eleven (11) figures:

1) Dotty plots of parameter values.
2) Histograms of parameter values.
3) Boxplots of parameter values.
4) Correlation matrix among parameter values (optional).
5) Empirical CDFs of parameter values.
6) Parameter values vs Number of Model Evaluations.
7) (pseudo) 3D dotty plots of (selected) parameter values.
8) GoF for each particle against Number of Model Evaluations.
9) Velocity values vs Number of Model Evaluations.

10a) Scatterplot between Best Simulated values and Observations (OPTIONAL, only if `MinMax` is provided).

10b) Empirical CDFs for model's output (only produced if `obs` is NOT a zoo object).

10b) ggof (See `?ggof`) between Best Simulated values and Observations (OPTIONAL, only if `obs` is a zoo object).

10d) Empirical CDFs for selected quantiles of model's output (OPTIONAL, only if `obs` is a zoo object).

11) Convergence Measures (Gbest and normSwarmRadius) vs Iteration Number.

You can try the basic automatic plotting of the calibration results into the current graphic device (usually, your screen) with:

```r
plot_results()
```

9.5.2 Advanced plotting

Notwithstanding the automatic plotting of the calibration results might be good enough for most purposes, you should read `?plot_results` and customise some of its arguments for getting figures that fullfil your case-specific needs:

```r
plot_results(do.png=TRUE,        # plots go to PNG figures instead of to the screen
             MinMax="max",       # 'best' parameter set is the one with the highest GoF
             beh.thr=0.3,        # parameter sets with GoF >= 0.3 will be considered as behavioural
             ftype="dm",         # daily and monthly time series of 'sim' vs 'obs' will be produced
             FUN=mean,           # monthly values are obtained from daily ones using 'mean'
             do.pairs=TRUE,      # a basic correlation plot is produced among parameters and GoF
             gof.name="KGE2012", # name used for axis titles in plots related to parameters
             legend.pos="right", # position of the legend in the convergence plot
             main=main           # user-defined title for most of the output figures
             )
```

```
## [                                              ]

## [         Reading output files ...             ]

## [                                              ]

##

## [ Reading the file 'Particles.txt' ... ]

## [ Total number of parameter sets: 4000 ]

## [ Number of behavioural parameter sets: 3517 ]

##

## [ Reading the file 'Velocities.txt' ... ]

## [ Total number of parameter sets: 4000 ]

## [ Number of behavioural parameter sets: 3517 ]

##

## [ Reading the file 'Model_out.txt' ... ]

## [ Total number of parameter sets: 4000 ]

## [ Number of model outputs for each parameter set ('nsim'): 6940 ]

## [ Number of behavioural model outputs           : 3517 ]

##

## [ Reading the file 'ConvergenceMeasures.txt' ... ]

## [ Total number of iterations: 50 ]

## [ Number of iterations with Gbest >= 0.3: 50 ]

##

## [ Reading the file 'Particles_GofPerIter.txt' ... ]

## [ Number of particles : 80 ]

## [ Number of iterations: 50 ]

## [                                              ]
## [                  Plotting ...                 ]
## [                                               ]
##
## [ Plotting dotty plots for parameter values into 'Params_DottyPlots.png' ... ]
## [ Plotting histograms for parameter values into 'Params_Histograms.png' ... ]
## [ Plotting boxplots for parameter values into 'Params_Boxplots.png' ... ]
## [ Plotting correlation matrix for parameter values into 'Params_Pairs.png' ... ]
## [ Plotting empirical CDFs for parameter values into 'Params_ECDFs.png' ... ]
## [ Plotting parameter values vs Number of Model Evaluations into 'Params_ValuesPerRun.png' ... ]
## [ Plotting 3D dotty plots for parameter values into 'Params_dp3d.png' ... ]
## [ Plotting GoF for each particle vs Number of Model Evaluations into 'Particles_GofPerIter.png' ... ]
## [ Plotting velocity values vs Number of Model Evaluations into 'Velocities_ValuePerRun.png' ...]
## [ Plotting best simulated values vs observations into 'ModelOut_BestSim_vs_Obs-Corr.png' ... ]
## [ Plotting best simulated values vs observations into 'ModelOut_BestSim_vs_Obs-ggof.png' ... ]
## [ Plotting ECDFs of simulated quantiles vs observations into 'ModelOut_Quantiles.png' ... ]
## [ Computing un-weighted quantiles for each ROW of 'sim' ... ]
##   |                                                                               |
```

All the figures generated by the previous call to `plot_results` **are not discussed in this tutorial**, but you can read some discussions about them on Rojas and Zambrano-Bigiarini (2012), Zambrano-Bigiarini and Rojas (2013), Abdelaziz and Zambrano-Bigiarini (2014) and Abdelaziz et al. (2019).

9.5.2.1 Convergence along iternations

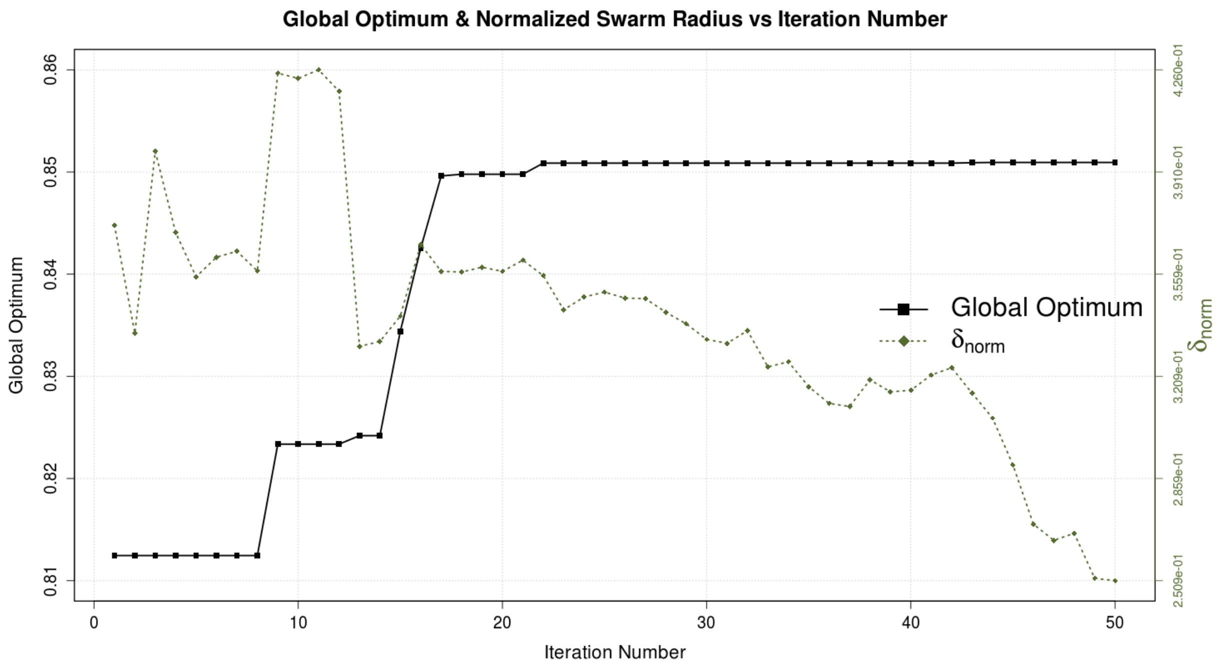


Figure 10: Evolution of global optimum (i.e., KGE2012) and the normalized swarm radius (a measure of swarm spread on the search space) against the number of iterations.

9.5.2.2 Parameter figures

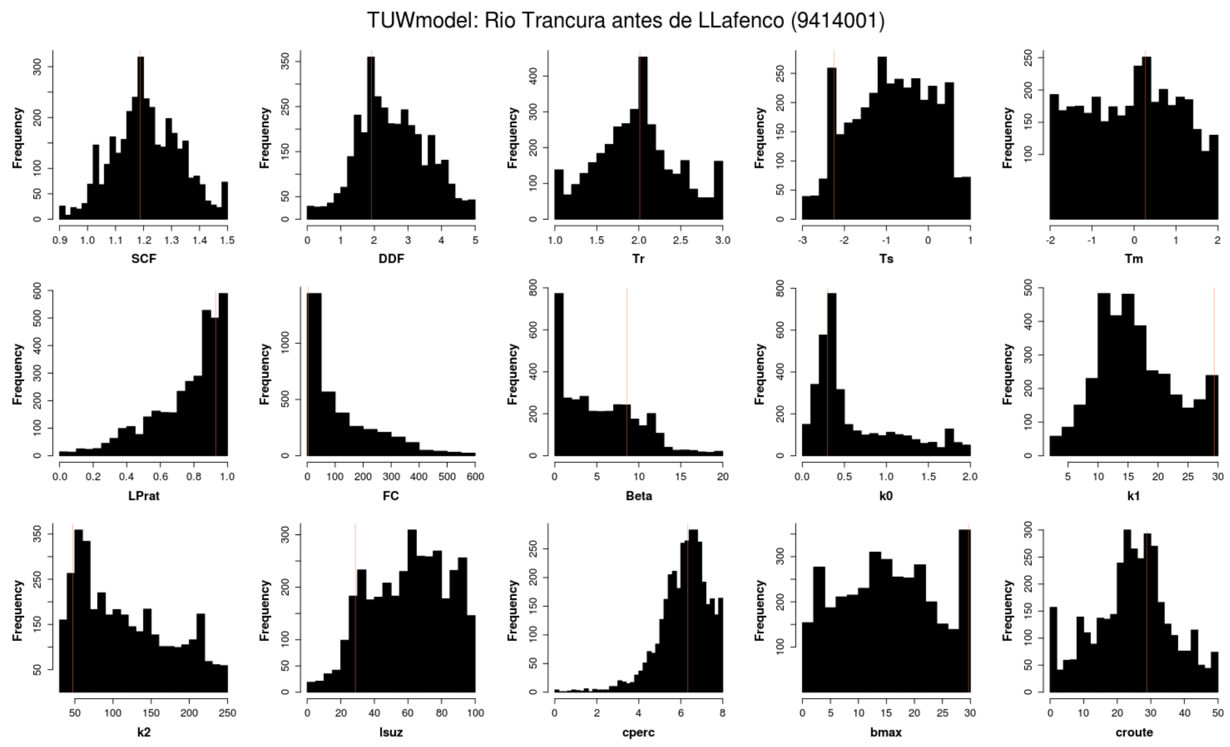


Figure 11: Histograms of user-defined behavioural parameter sets (‘KGE2012 >= beh.thr‘) versus parameter values. The vertical red line indicates the optimum parameter value. Parameter names are described in Table 1.

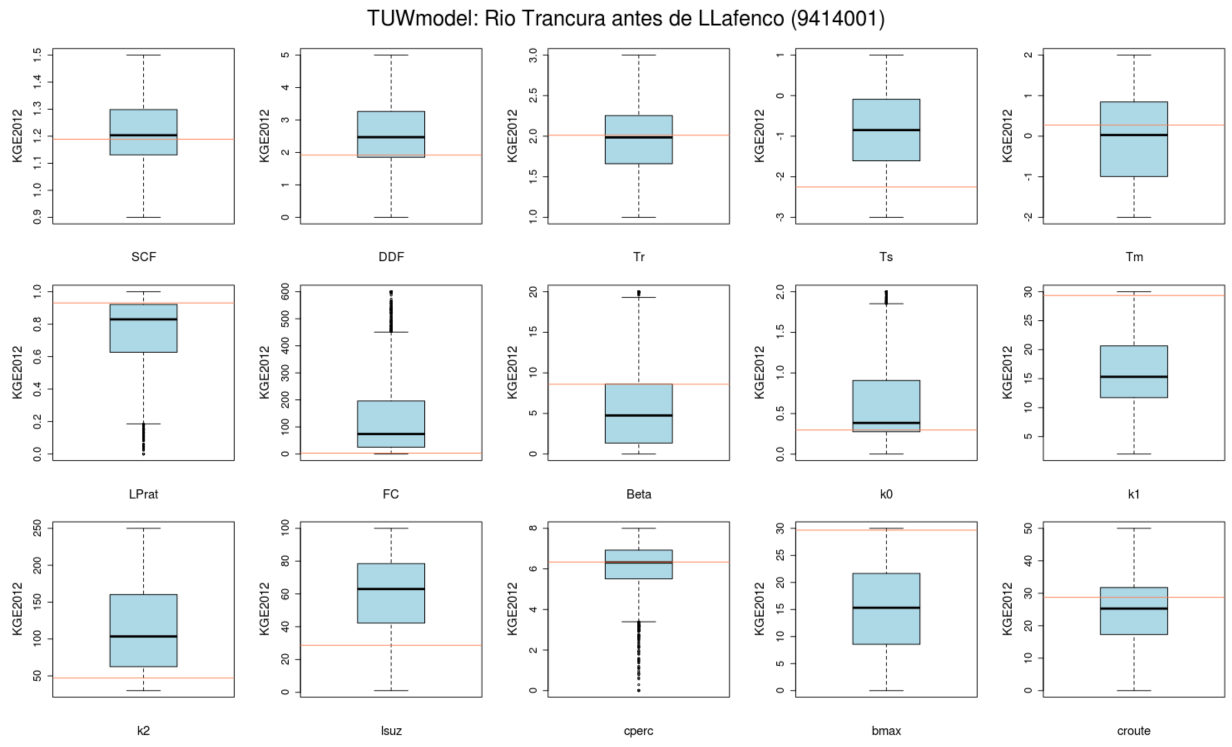


Figure 12: Box-and-whisker plots (or boxplots) of user-defined behavioural parameter sets (‘KGE2012 >= beh.thr‘) versus parameter values. The horizontal red line indicates the optimum parameter value found during the optimisation. Parameter names are described in Table 1.

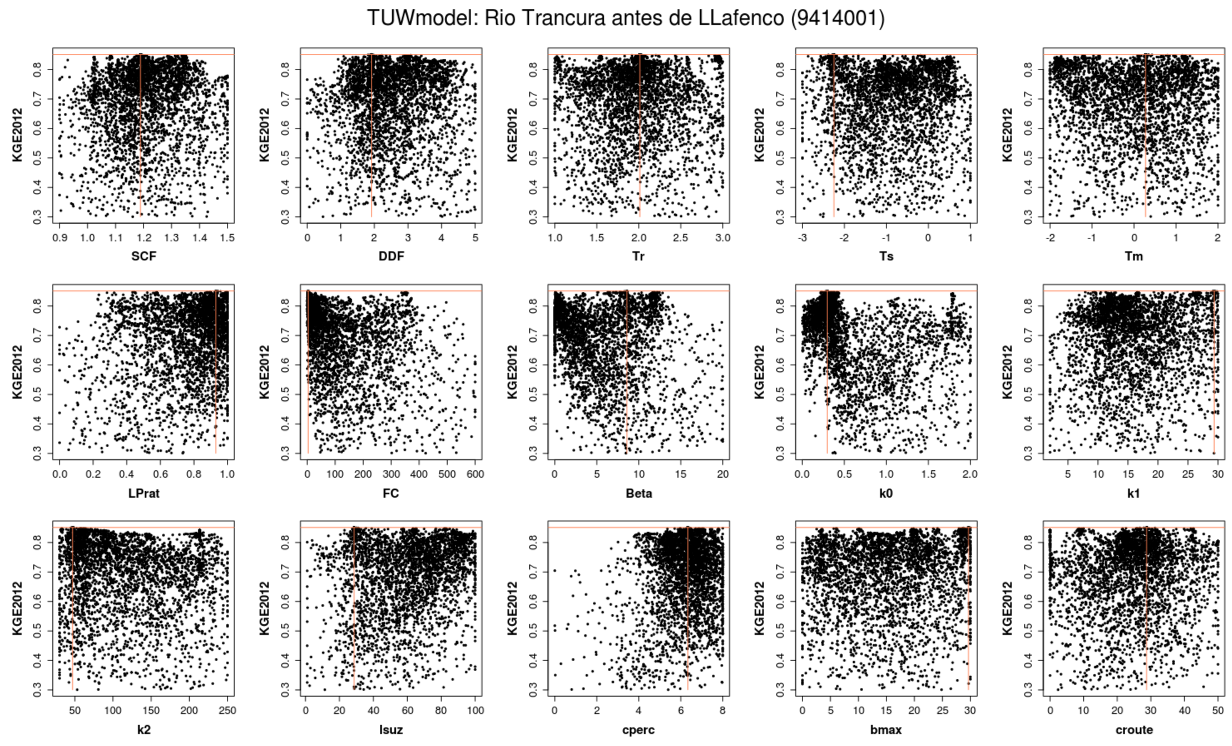


Figure 13: Dotty plots showing the model performance of behavioural parameter sets (‘KGE2012 >= beh.thr‘) versus parameter values. The vertical red line indicates the optimum parameter value found during the optimisation, whereas the horizontal red line shows the corresponding best model performance. Parameter names are described in Table 1.

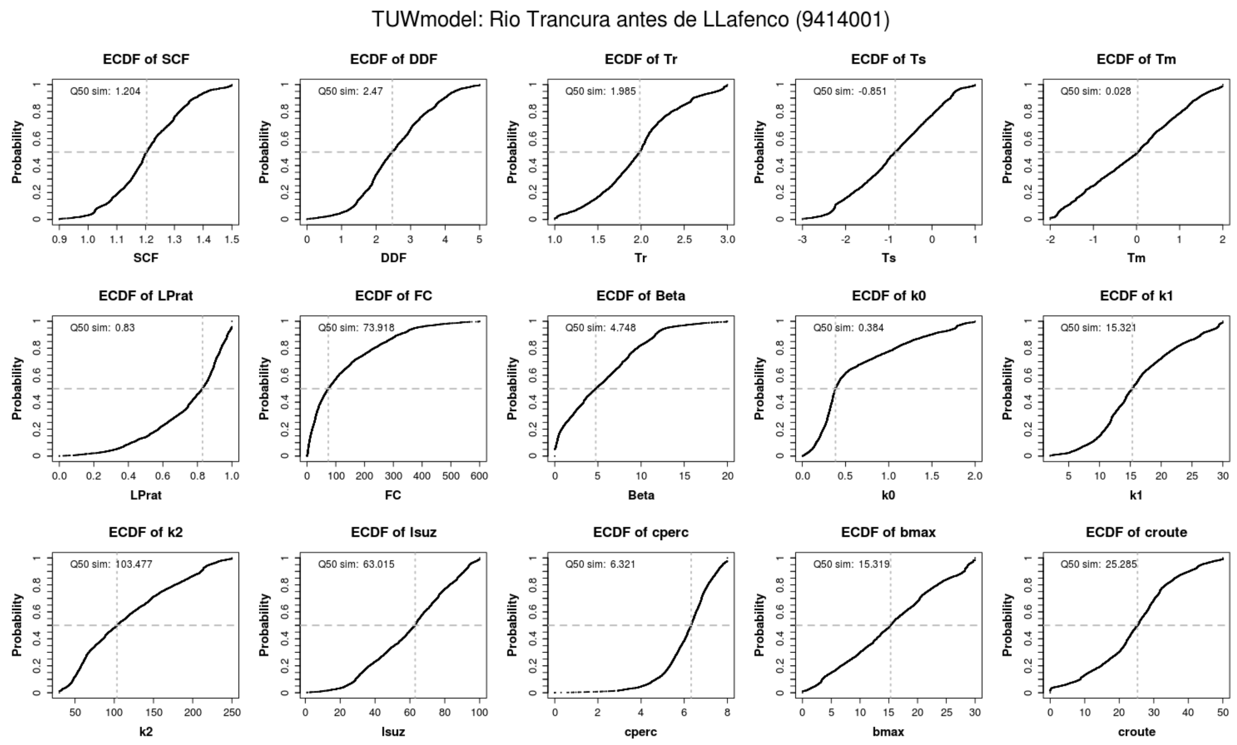


Figure 14: Empirical CDFs (continuous black lines) of parameter values. Horizontal gray dotted lines represent a cumulative probability equal to 0.5 (median of the distribution). Vertical gray dotted lines represent the parameter value corresponding to a cumulative probability equal to 0.5 (its value is shown in the upper part of each figure).

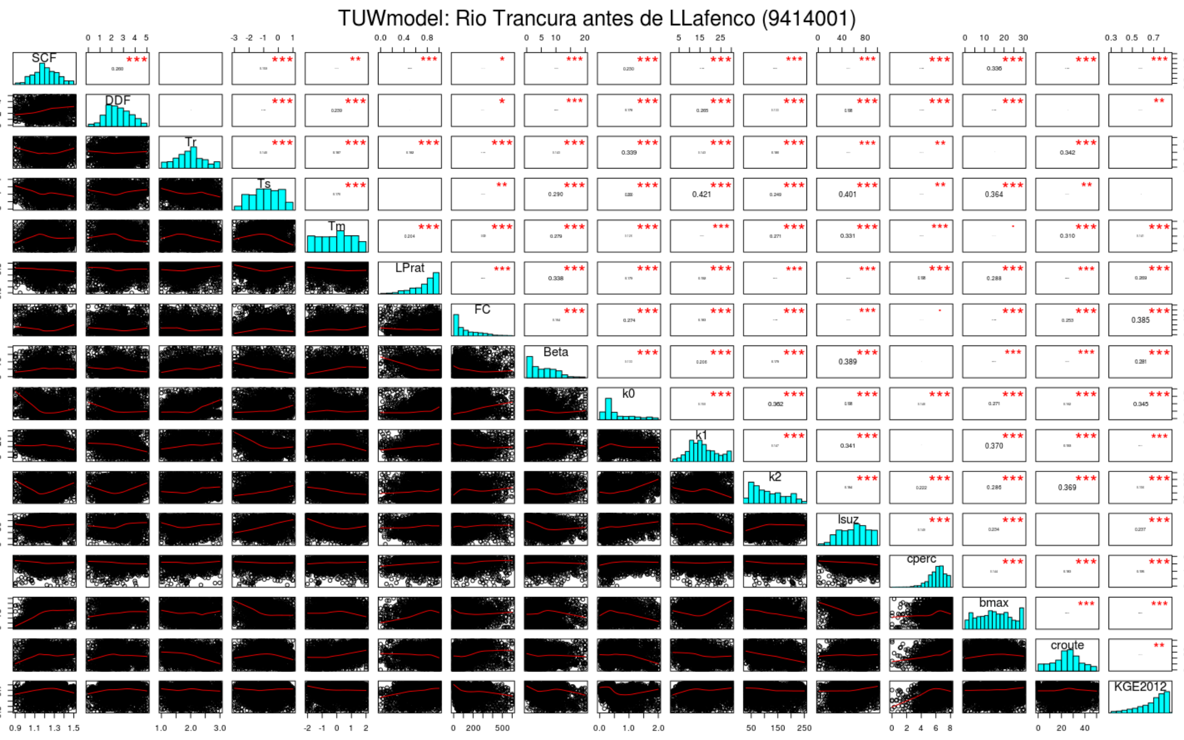


Figure 15: Corelation matrix between parameters and model performance (KGE2012). Horizontal and vertical axes show the physical range used for the calibration of each parameter. Red curved lines represents are fitted to all the pair of points using a lowess smoother, i.e., a locally-weighted polynomial regression.

9.5.2.3 Model output (streamflow) figures

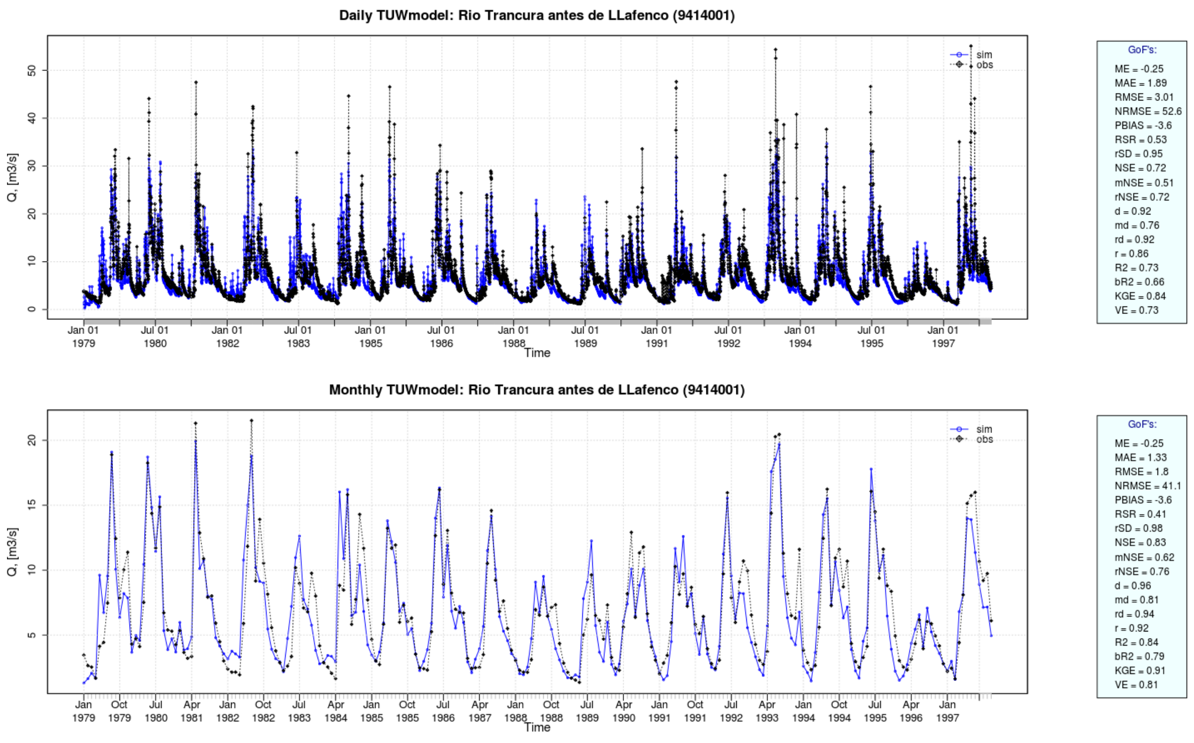


Figure 16: Comparison of daily (upper panel) and monthly (lower pannel) streamflows obtained with the 'best' parameter set and their corresponding observed counterparts.

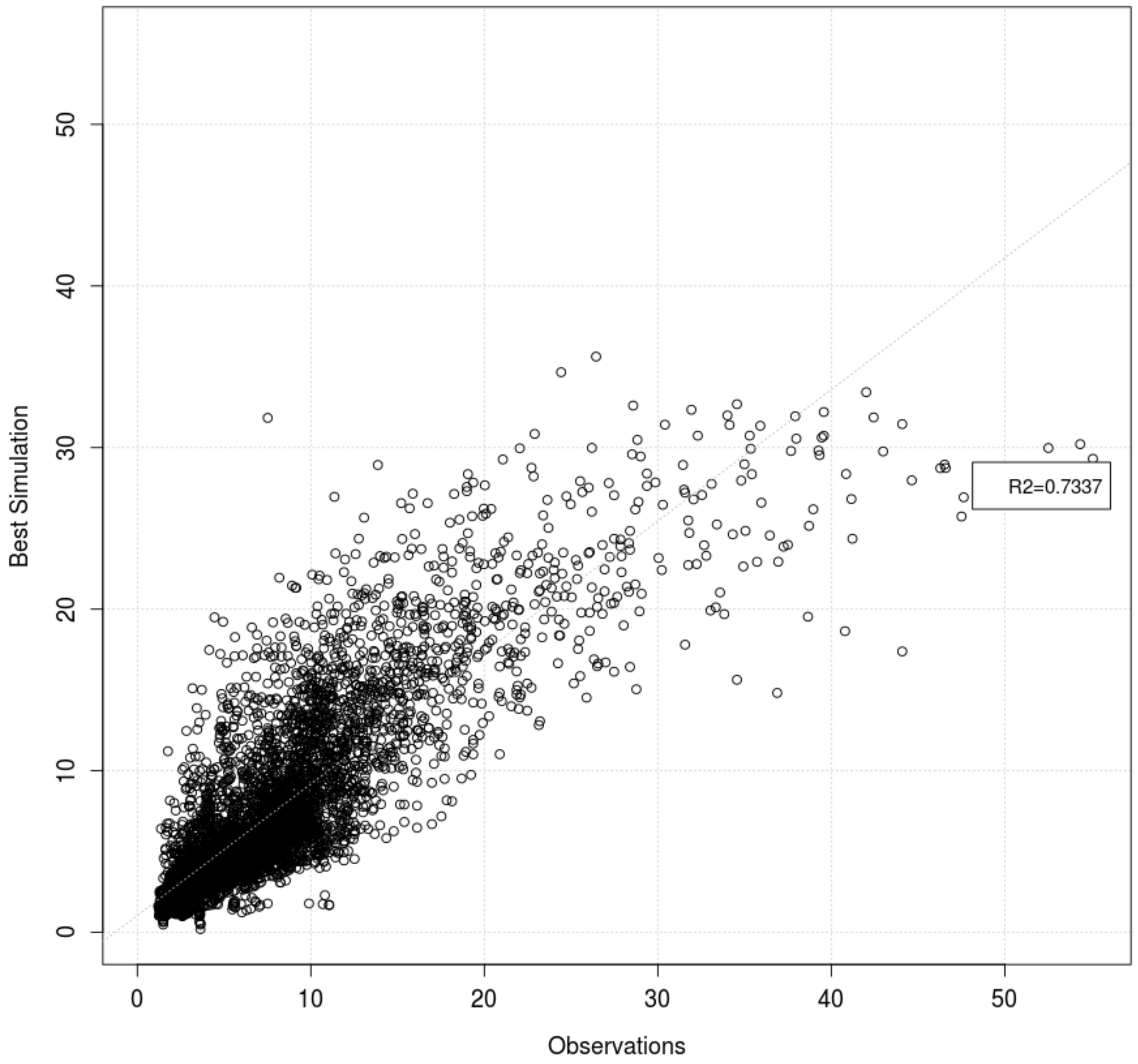


Figure 17: Scatterplot of daily streamflows obtained with the 'best' parameter set obtained during calibration and their corresponding observed counterparts.

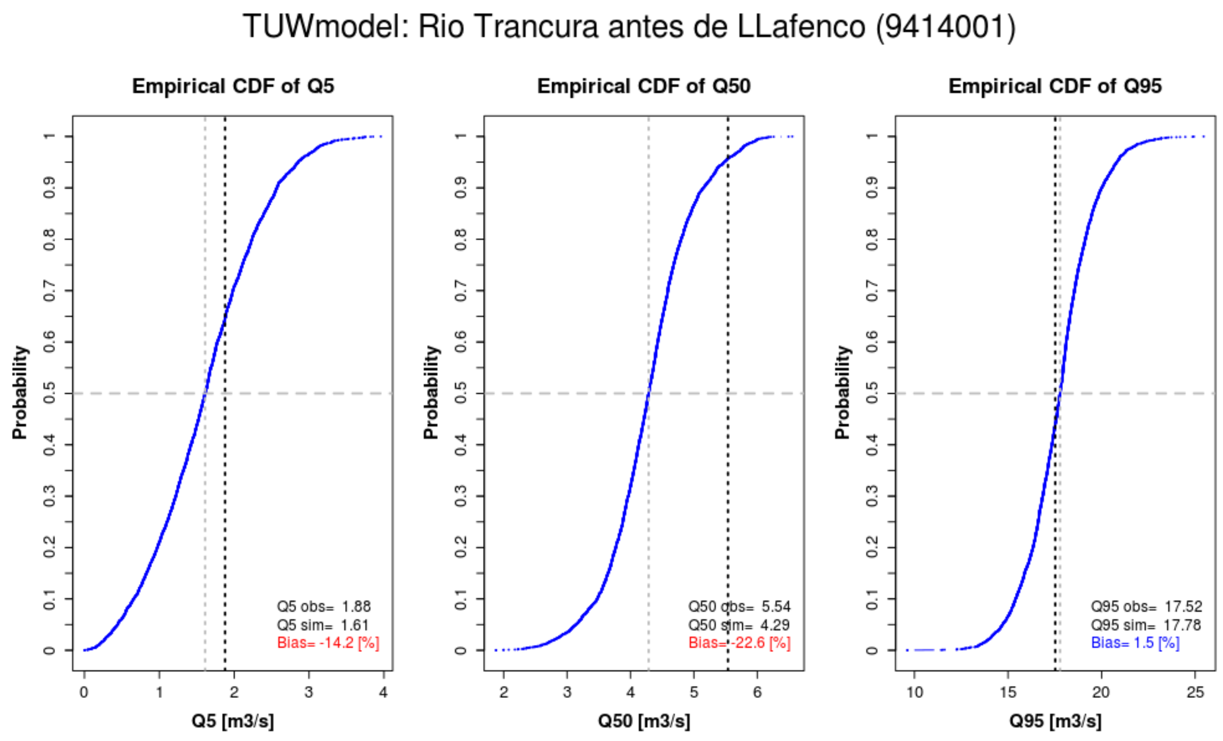


Figure 18: Empirical CDFs (blue lines) for 5, 50 and 95 quantiles of simulated discharges obtained by 'TUWmodel' during the calibration period. Vertical black-dotted lines represent the observed quantiles, while vertical grey-dotted lines indicate the median of the simulated quantiles.

9.6 Uncertainty analysis of calibration results

If you are not familiar with uncertainty analysis, you might be interested in Refsgaard et al. (2007), who review the terminology and typology of uncertainty, its interaction with the broader water management process and the role of uncertainty at different stages in the modelling processes.

9.6.1 Reading all calibration results

In order to proceed with the uncertainty analysis, we need to read all the calibrations results generated by `hydroPSO` during the optimisation. However, for this tutorial we are only interested in **behavioural parameter sets** (Beven and Binley 1992; Beven and Freer 2001; Beven 2006), as defined by an arbitrary and user-defined goodness-of-fit value (i.e., a KGE2012 value). For this particular case, we choose `beh.thr=0.3`. Setting `beh.thr=0.3` and `MinMax='max'` will consider as behavioural all those parameter sets with goodness-of-fit values higher than the user-defined threshold 0.3:

```
PSO.drty <- paste0(model.drty, "/PSO.out/")
res      <- read_results(drty.out=PSO.drty, MinMax="max", beh.thr=0.3)
```

```
## [                                                ]

## [         Reading output files ...               ]

## [                                                ]

##

## [ Reading the file 'Particles.txt' ... ]

## [ Total number of parameter sets: 4000 ]

## [ Number of behavioural parameter sets: 3517 ]

##

## [ Reading the file 'Velocities.txt' ... ]

## [ Total number of parameter sets: 4000 ]

## [ Number of behavioural parameter sets: 3517 ]

##

## [ Reading the file 'Model_out.txt' ... ]

## [ Total number of parameter sets: 4000 ]

## [ Number of model outputs for each parameter set ('nsim'): 6940 ]

## [ Number of behavioural model outputs           : 3517 ]

##

## [ Reading the file 'ConvergenceMeasures.txt' ... ]

## [ Total number of iterations: 50 ]

## [ Number of iterations with Gbest >= 0.3: 50 ]

##

## [ Reading the file 'Particles_GofPerIter.txt' ... ]

## [ Number of particles : 80 ]

## [ Number of iterations: 50 ]
```

The next code chunck assigns all the behavioural parameter sets obtained during calibration (*params*), i.e., those with goodness-of-fit value higher than `beh.thr`; the goodness-of-fit value of each behavioural parameter set (*gofs*), the model outputs (i.e., streamflows for the `TUWmodel` case) corresponding to each beahvioural parameter set (*Qsims*), the model output corresponding to the best parameter set obtained during calibration (*model.best*), and the observed values used in the calibration to compute the goodness-of-fit value (*model.obs*):

```
params     <- res[["params"]]
gofs       <- res[["gofs"]]
Qsims      <- res[["model.values"]]
model.best <- res[["model.best"]]
model.obs  <- res[["model.obs"]]
```

9.6.2 Weighted quantiles of parameter values

The computation of weighted quantiles of model parameters would provide an estimate of the uncertainty in each model parameter, based on the behavioural parameter sets obtained during calibration and the goodness-of fit values obtained for each one of them. User-defined weighted quantiles for each model parameter are obtained multiplying the standard quantile derived for each model parameter based on all the behavioural pameter sets (i.e., for each column in `params`) by the corresponding goodness-of-fit value obtained by the parameter set (*gofs*)[4]. For this tutorial, we are going use the `wquantile` function of `hydroPSO` to compute the 2.5, 50 and 97.5 weighted quantiles for each model parameter, by setting `probs=c(0.025, 0.5, 0.975)`:

```
params.025.50.975 <- wquantile(params, weights=gofs, byrow=FALSE, probs=c(0.025, 0.5, 0.975),
                               normwt=TRUE, verbose=FALSE)
```

You migh be interested in how far are the best parameter set foubnd during the calibration from the median of the weighted behavioural parameter sets. In the following code chunck we will add a new column to `params.025.50.975`, just after the median (quantile 50):

```
params.025.50.best.975 <- cbind(params.025.50.975[, c(1,2)],
                                Best=as.numeric(best.param.cal),
                                params.025.50.975[, 3] )
colnames(params.025.50.best.975)[4] <- "97.5%"
round( params.025.50.best.975, 2)
```

```
FALSE          2.5%    50%  Best  97.5%
FALSE SCF      1.00   1.20  1.19   1.46
FALSE DDF      0.70   2.48  1.92   4.53
FALSE Tr       1.03   1.99  2.01   2.96
FALSE Ts      -2.55  -0.84 -2.25   0.67
FALSE Tm      -1.87   0.01  0.27   1.92
FALSE LPrat    0.26   0.84  0.93   1.00
FALSE FC       2.56  62.67  3.18 426.67
FALSE Beta     0.00   4.38  8.60  14.55
FALSE k0       0.06   0.36  0.30   1.83
FALSE k1       5.33  15.32 29.34  29.42
FALSE k2      33.61 100.63 47.08 233.10
FALSE lsuz    17.85  63.78 28.66  98.10
FALSE cperc    3.69   6.33  6.34   7.97
FALSE bmax     1.20  15.40 29.66  29.66
FALSE croute   0.12  25.48 28.75  47.28
```

[4] For more information about the computation of the weighted quantiles, please read `?wtd.stats`.

9.6.3 Weighted quantiles of model simulations

The computation of weighted quantiles of model simulations would provide an estimate of the uncertainty in each model simulated value (i.e., each daily streamflow), based on the goodness-of fit values obtained for the full time series simulated by the model. User-defined weighted quantiles for each model simulation are obtained multiplying the standard quantile derived for each model simulation based on all the model simulations (i.e., for each column in `Qsims`) by the corresponding goodness-of-fit value obtained by the full time series simulated by the model (*gofs*). For this tutorial, we are going use the `wquantile` function of `hydroPSO` to compute the 2.5, 50 and 97.5 weighted quantiles for each value simulated by the model, by setting `probs=c(0.025, 0.5, 0.975)`:

```
n     <- ncol(Qsims)     # number of time steps
Qsims <- Qsims[, 366:n] # removing the first year as warming up period
```

```
Qsim.025.q50.q975   <- wquantile(Qsims, weights=gofs, byrow=FALSE, probs=c(0.025, 0.5, 0.975),
                                 normwt=TRUE, verbose=FALSE)
round( head(Qsim.025.q50.q975), 3)
```

```
FALSE         2.5%   50% 97.5%
FALSE Sim366 2.553 4.118 5.287
FALSE Sim367 2.515 4.079 5.217
FALSE Sim368 2.498 4.032 5.137
FALSE Sim369 2.465 3.980 5.062
FALSE Sim370 2.436 3.927 4.993
FALSE Sim371 2.424 3.882 4.922
```

To plot the uncertainty bounds for each model simulated value, we are going to transform the weighted quantiles 2.5 and 97.5 into zoo objects:

```
dates.cal <- dip(Cal.Ini, Cal.Fin)
q025      <- zoo(Qsim.025.q50.q975[,1], dates.cal)
q975      <- zoo(Qsim.025.q50.q975[,3], dates.cal)
```

9.6.4 P-factor and R-factor

Now we are going to compute the **P-factor** (Schuol et al. 2008; Abbaspour et al. 2009), which represents the percent of observations that are within the user-defined uncertainty bounds:

```
pf <- pfactor(x=Qobs.zoo.cal, lband=q025, uband=q975, na.rm=TRUE)
pf
```

```
## [1] 0.754981
```

Also, we are going to compute the **R-factor** (Schuol et al. 2008; Abbaspour et al. 2009), which represents the average width of the user-defined uncertainty bounds divided by the standard deviation of the observations:

```
rf <- rfactor(x=Qobs.zoo.cal, lband=q025, uband=q975, na.rm=TRUE)
rf
```

```
## [1] 0.9308071
```

> *A good model should bracket most of the measured data (i.e., large P-factor, with a maximum of 1) with the smallest possible R-factor (minimum of 0).*

9.6.5 95 Percent Prediction Uncertainty (95 PPU)

Now we are ready to plot the *best* simulated streamflows along with the **95 Percent Prediction Uncertainty** (**95 PPU**) (Abbaspour et al. 2007, 2009; Schuol et al. 2008):

```
main.uncert   <- paste0("Uncertainty bounds TUWmodel: ",
                        Qobs.stationname, " (", Qobs.ID, ")" )
Qobs.col      <- "black"
best.sim.col <- "blue"
bands.col     <- "lightblue"

Qsim.best.zoo <- zoo(model.best, time(Qobs.zoo.cal) )

plot(Qobs.zoo.cal, xaxt="n", xlab="", type="n", main=main.uncert, ylab="Q, [mm]") # empty plotting area
drawTimeAxis(Qobs.zoo.cal)                                      # time axis
plotbandsonly(lband=q025, uband=q975)                           # 95 PPU
lines(Qobs.zoo.cal, col=Qobs.col, lwd=0.3)                      # Qobs
lines(Qsim.best.zoo, col=best.sim.col, lwd=0.3)                 # best simulation
grid()

legend("topright", legend=c("Qobs", "best.sim", "95PPU"), lty=c(1, 1, NA),
       pch=c(NA, NA, 15), col=c(Qobs.col, best.sim.col, bands.col),
       bty="n", cex = 1.2)

legend("topleft", legend=c(paste0("P-factor: ", round(pf, 2)),
                           paste0("R-factor: ", round(rf, 2)) ),
       bty="n", cex = 1.2)
```

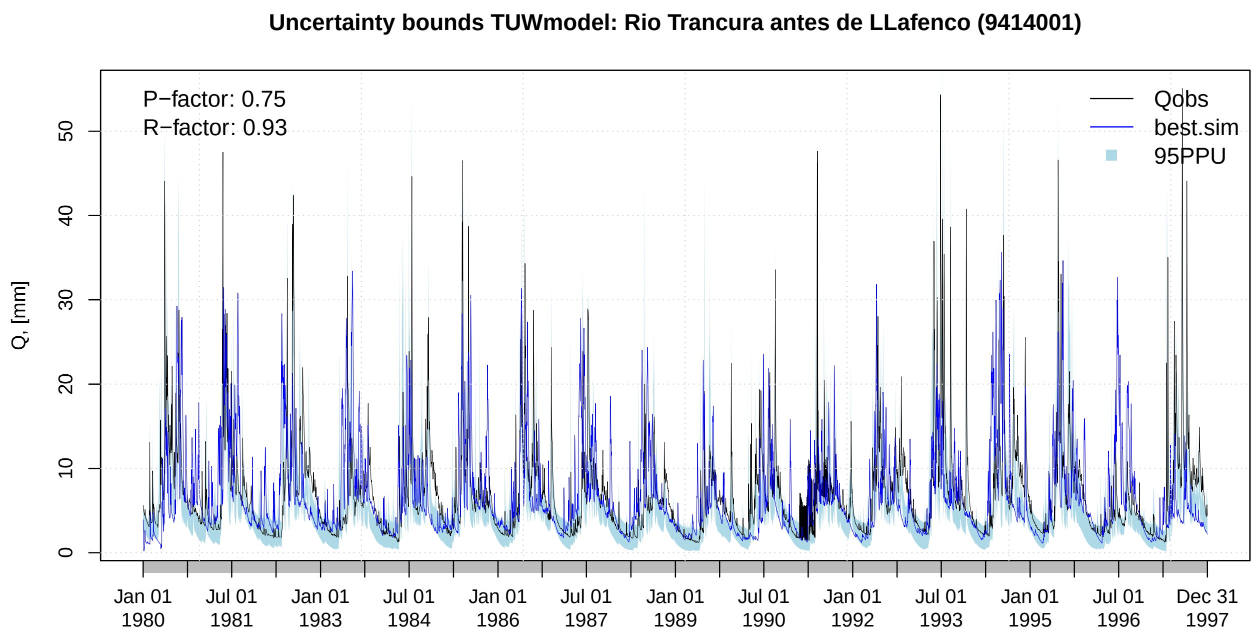


Figure 19: 95 Percent Prediction Uncertainty (95 PPU) for model simulations during the calibration period.

9.6.6 Uncertainty in flow duration curve

Now we are going to use the `fdcu` included in the `hydroTSM` R package to plot the observed flow duration curve (FDC), the one of the *best* simulated streamflows, and the flow duration curves for the 2.5 and 97.5 weighted quantiles of model simulations obained during the calibration period:

```
main.uncert  <- paste0("Flow duration curves GR4J model: ", Qobs.stationname, " (", Qobs.ID, ")" )
bands.col    <- "lightskyblue1"

par(mfrow=c(3,1))
fdcu(x=Qobs.zoo.cal, lband=q025, uband=q975, sim=Qsim.best.zoo, bands.col=bands.col,
     main=main.uncert, log="") # empty plotting area
grid()
fdcu(x=Qobs.zoo.cal, lband=q025, uband=q975, sim=Qsim.best.zoo, bands.col=bands.col,
     main=main.uncert, log="y") # empty plotting area
grid()
fdcu(x=Qobs.zoo.cal, lband=q025, uband=q975, sim=Qsim.best.zoo, bands.col=bands.col,
     main=main.uncert, log="x") # empty plotting area
grid()
```

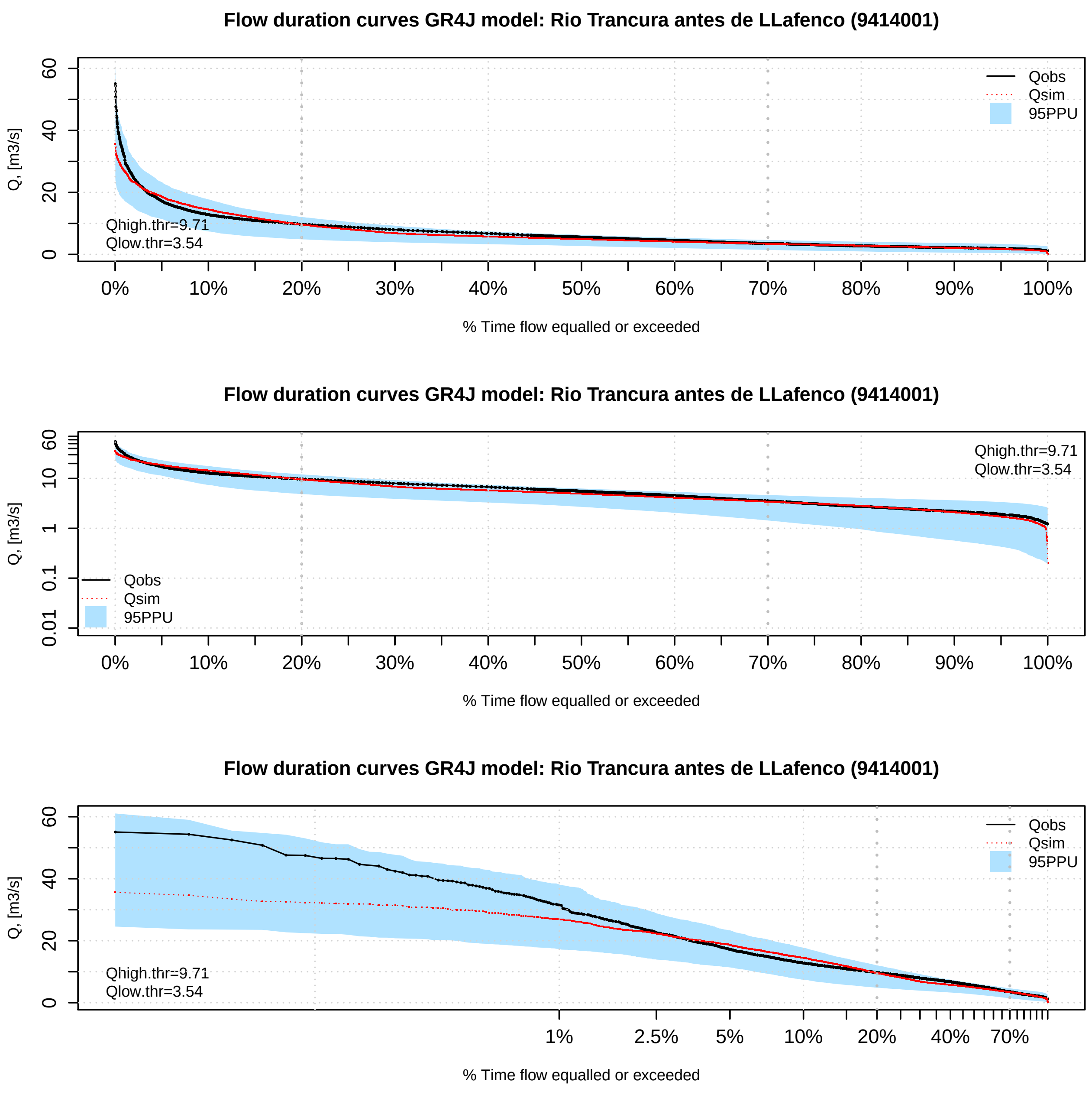


Figure 20: Flow duration curve of the observed (black line) and best simulated (blue line) streamflows. In addition, flow duration curves for the 2.5 and 97.5 weighted quantiles of model simulations obained during the calibration period. The upper panel is the normal flow duration curve, the middle panel has focus on low flows (log='y') and the lower panel has focus on high flows (log='x').

9.6.7 Some comments

GLUE-like uncertainty analysis (Beven and Binley 1992; Beven 2006; Beven, Smith, and Freer 2008) is sensitive to the threshold value used to select behavioral parameter sets (`beh.thr`), with lower threshold values resulting in a wider uncertainty bounds of the posterior distribution of parameters and model simulations (Jin et al. 2010). Usually, the lower the `beh.thr` value the higher the P-factor, but at expense of a higher R-factor. . . .

10 Verification analysis

10.1 Basic analysis of the verification period

Running `TUWmodel` with the *best* parameter set obtained during the calibration for the whole (calibration + verification) period:

```r
modeloutput <- TUWmodel(param=best.param.cal, prec=P, airt=Temp, ep=PET)
```

Transforming the numeric output of the model into a zoo object:

```r
Qsim.zoo <- zoo(as.numeric(modeloutput$q), dates)
```

Subsetting the model simulations and observed streamflows to the verification period only:

```r
Qsim.zoo.ver <- window(Qsim.zoo, start=Ver.Ini, end=Ver.Fin)
Qobs.zoo.ver <- window(Qobs.zoo, start=Ver.Ini, end=Ver.Fin)
```

Graphical comparison of the daily observed and simulated time series obtained with the *best* paramer set for the verification period:

```r
ggof(sim=Qsim.zoo.ver, obs=Qobs.zoo.ver, main=main, ylab="Q, [mm]", cex=0.3, lty=c(1,1) )
```

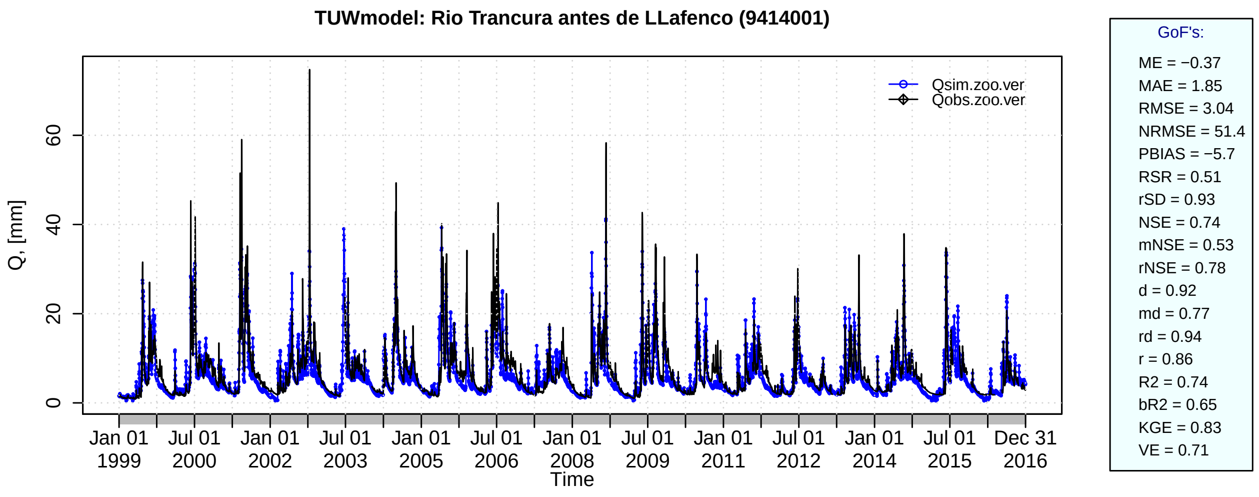
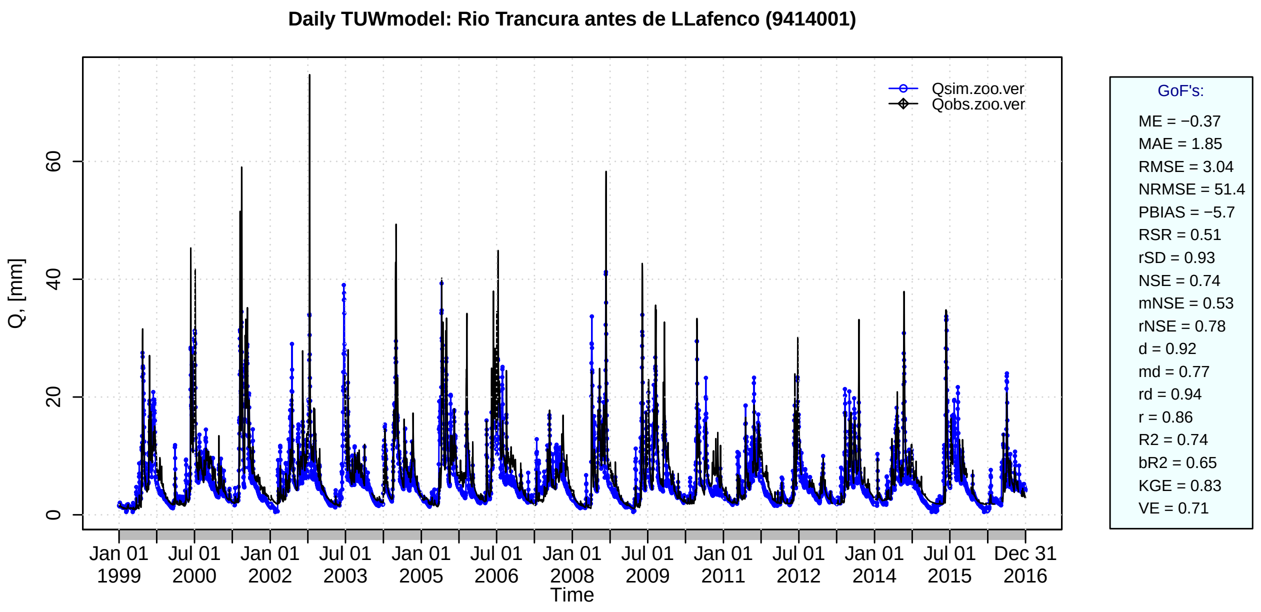


Figure 21: Graphical comparison of observed and simulated streamflow time series obtained with the *best* paramer set obtained during the calibration.

It could be good enough, right?

Graphical comparison of daily and monthly simulated ($Qsim.zoo.ver$) and observed ($Qobs.zoo.ver$) values for the verification period:

```
ggof(sim=Qsim.zoo.ver, obs=Qobs.zoo.ver, ftype="dm",
     FUN=mean, main=main, ylab="Q, [mm]", cex=0.4, lty=c(1,1) )
```



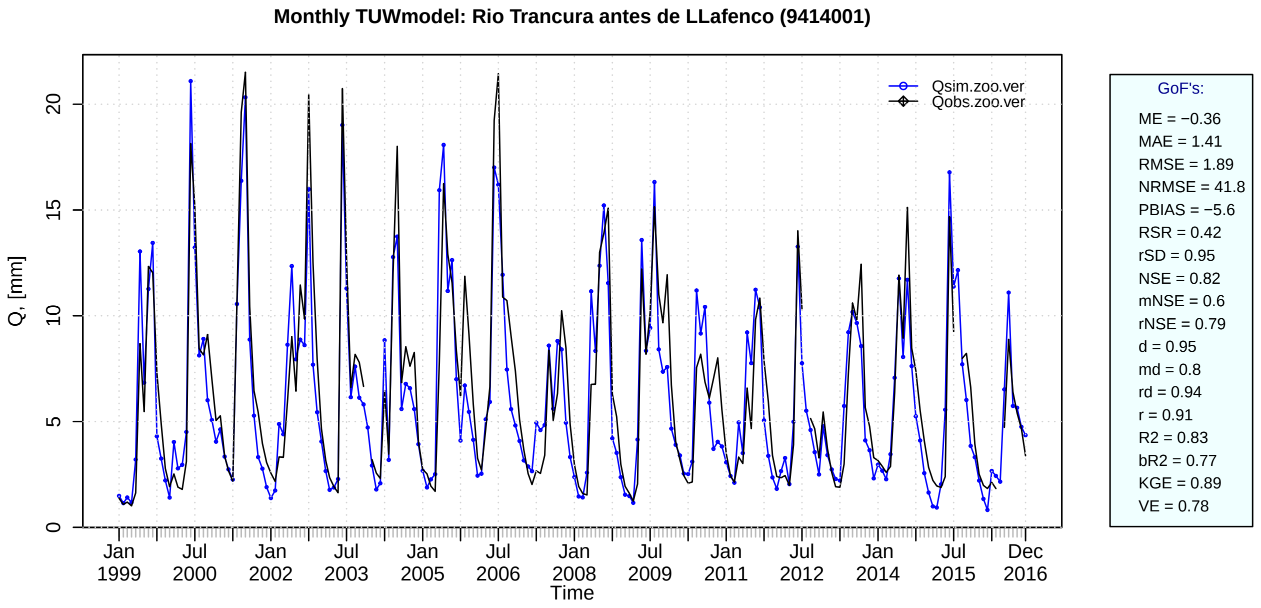


Figure 22: Daily and monthly evaluation of simulated streamflows for the verification period using several performance indices.

Graphical comparison of monthly and annual simulated and observed values for the verification period:

```
ggof(sim=Qsim.zoo.ver, obs=Qobs.zoo.ver, ftype="ma", pt.style="bar",
     FUN=mean, main=main, ylab="Q, [mm]", cex=0.4, lty=c(1,1) )
```

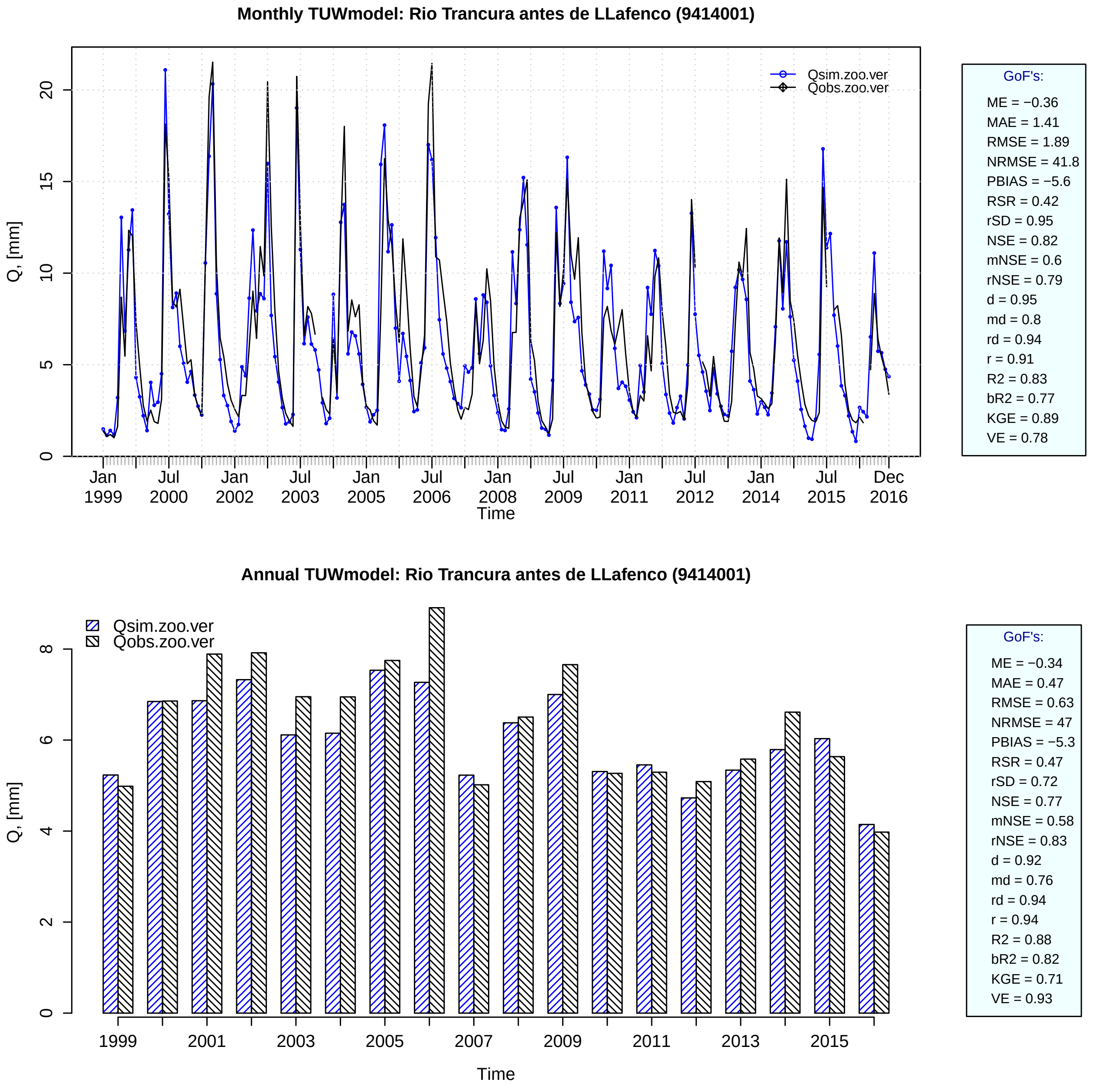


Figure 23: Monthly and annual evaluation of simulated streamflows for the verification period using several performance indices.

Graphical comparison of seasonal simulated and observed values (one plot for each weather season) for the verification period:

```r
ggof(sim=Qsim.zoo.ver, obs=Qobs.zoo.ver, ftype="seasonal", ylab="Q, [mm]",
     season.names=c("Summer", "Autumn", "Winter", "Spring"),
     FUN=mean, main=main, cex.main=2)
```

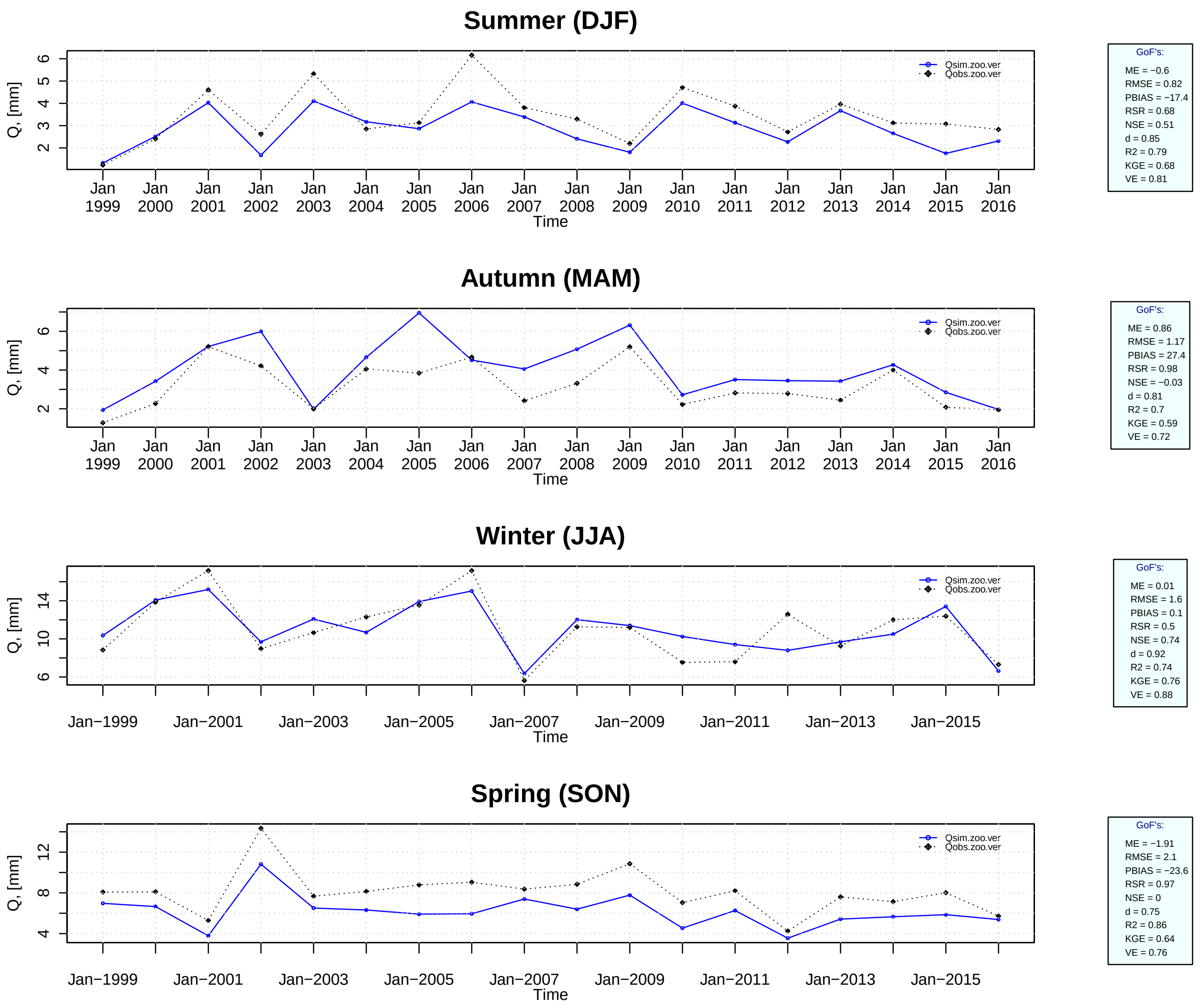


Figure 24: Seasonal evaluation of simulated streamflows for the verification period using several performance indices.

Computing the daily residuals for the verification period:

```
residuals <- Qsim.zoo.ver - Qobs.zoo.ver
residuals <- window(residuals, start=Ver.Ini)
```

Plotting daily, monthly and annual time series, boxplots and histograms of the residuals for the verification period:

```
hydroplot(residuals, FUN=mean, var.unit="mm")
```

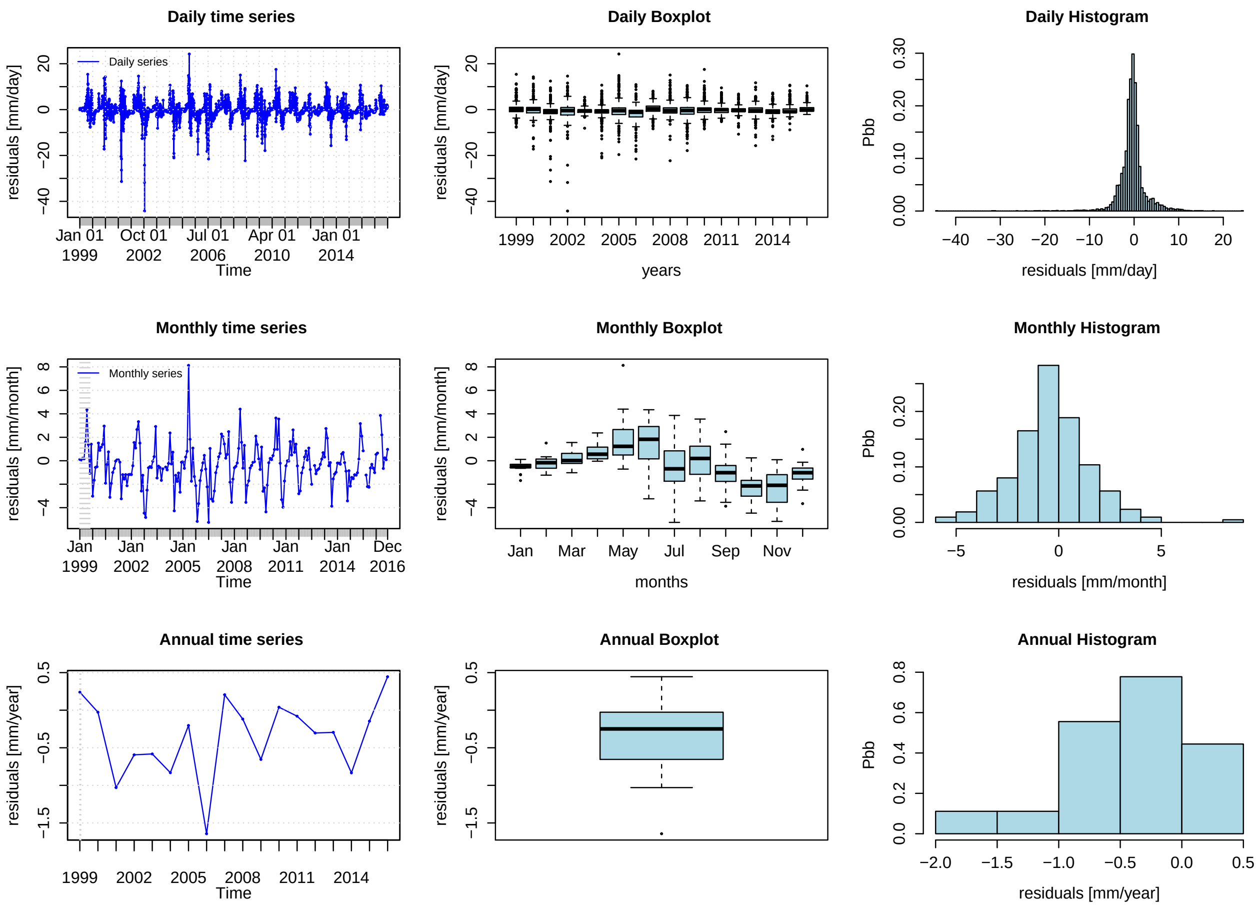


Figure 25: Analysis of the residuals (simulations - observations) for the verification period at the daily, monthly, and annual temporal scales.

Seasonal time series and boxplots of the residuals for the verification period:

```
hydroplot(residuals, FUN=mean, pfreq="seasonal",
          season.names=c("Summer", "Autumn", "Winter", "Spring"),
          h=0, main=main, cex.main=2)
```

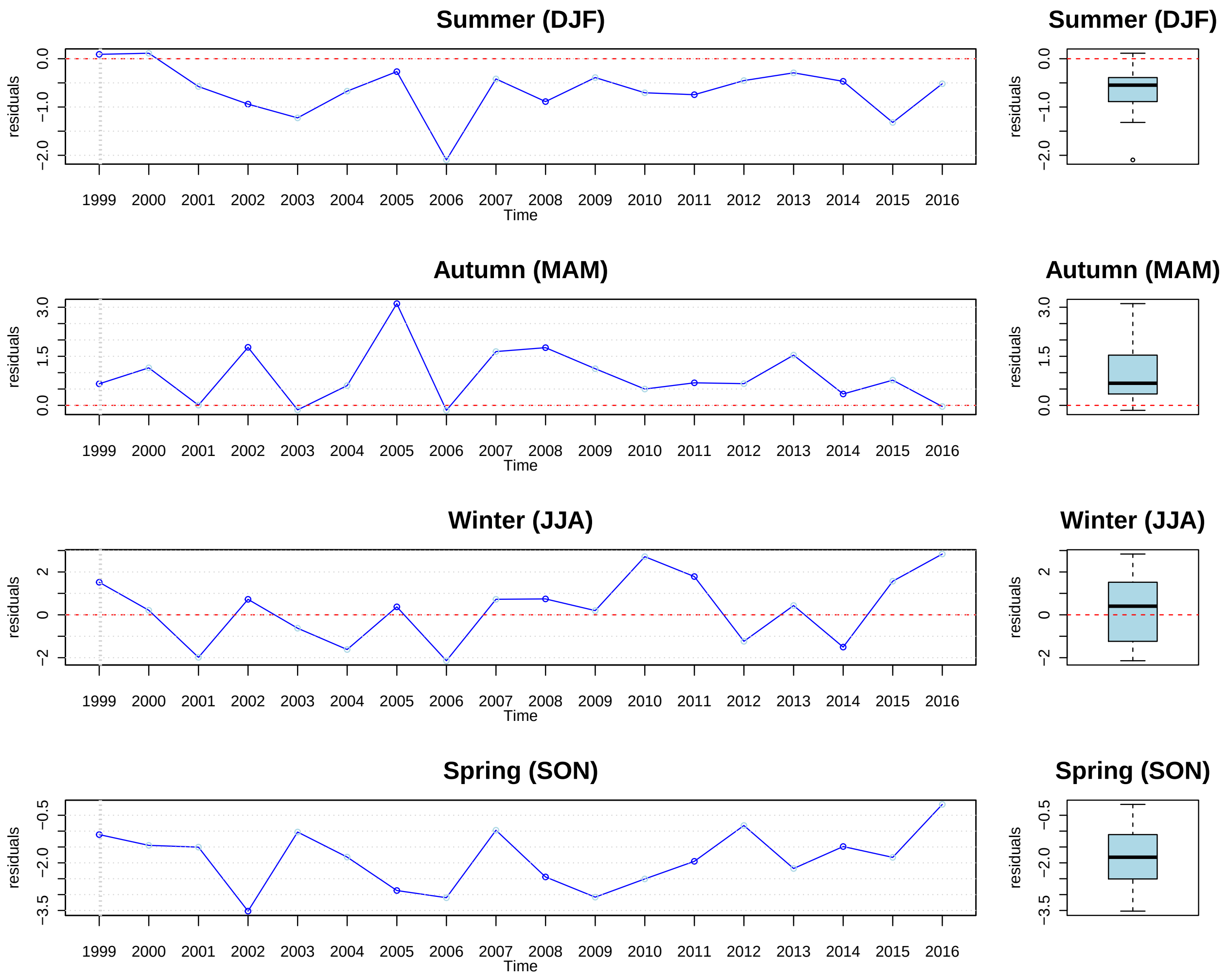


Figure 26: Analysis of the residuals (simulations - observations) for the verification period at the seasonal scale.

10.2 Running behavioural parameter sets

Before running TUWmodel with the behavioural parameter sets obtained during calibration, we are going to use as model inputs only time series belonging to the independent verification period:

```
P.zoo.ver    <- window(P.zoo   , start=Warmup.Ini.Ver, end=Ver.Fin)
Temp.zoo.ver <- window(Temp.zoo, start=Warmup.Ini.Ver, end=Ver.Fin)
PET.zoo.ver  <- window(PET.zoo , start=Warmup.Ini.Ver, end=Ver.Fin)
```

Storing the dates of the input time series used to drive the model verification (including the warming up period):

```
dates.ver <- time(P.zoo.ver)
```

As mentioned before, TUWmodel does not support zoo objects as input data, and it only works with plain numeric values. Therefore, we need to transform the input time series into numeric objects:

```
P.ver    <- as.numeric(P.zoo.ver)
Temp.ver <- as.numeric(Temp.zoo.ver)
PET.ver  <- as.numeric(PET.zoo.ver)
```

The wrapper function defined for the calibration period used as input time series values belonging to the calibration period, adn therefore, in order to use that function during the verification period we just need to modify those inputs to match the verification period:

```
TUWhydromod <- function(param.values,     # it MUST be present
                        obs=Qobs.zoo.ver, # it MUST be present
                        dates=dates.ver,  # Optional
                        warmup.days=365,  # Optional, 1 year of warming up
                        P=P.ver,          # Optional
                        Temp=Temp.ver,    # Optional
                        PET=PET.ver,      # Optional
                        Area=Area.m2      # Optional
                        ) {

  # Evaluating the hydrological model at the 'param.values' parameter set
  simLump  <- TUWmodel(param=param.values, prec=P, airt=Temp, ep=PET, area=Area)

   # Extracting only the numeric vector with simulations and observa
  qsim <- as.numeric(simLump$q)

  # Converting model simulations into zoo objects
  qsim.zoo.full <- zoo(qsim, dates)

  # Removing the warming up period (1 year, i.e., 365 days)
  n        <- length(qsim.zoo.full)
  qsim.zoo <- qsim.zoo.full[(warmup.days+1):n]
  obs      <- obs[(warmup.days+1):n]

  # Computing the goodness-of-fit value (i.e, model performance)
  gof      <- KGE(sim=qsim.zoo, obs=obs, method="2012")
  # Creating the output of the R function
  nelements <- 2
  out       <- vector("list", nelements)
  out[[1]]  <- gof
  out[[2]]  <- qsim.zoo.full

  # Mandatory names for the elements of the output
  names(out)[1:nelements] <- c("GoF", "sim")

  return(out)

} # 'TUWhydromod' end
```

Since `hydroPSO v0.5-0`, the `verification` function is fully compatible with R-based (hydrological/environmental) models, and therefore it can be used to run your model with all the behavioural parameter sets obtained during the calibration:

```
out <- verification(fn="hydromodInR",
                    par=params,
                    control=list(gof.name="KGE2012",  # Character, only used for identifying the goodne
                                 MinMax="max",         # Character, indicating if PSO have to find a min
                                 do.plots=FALSE,
                                 write2disk=TRUE,
                                 parallel="none",
                                 verbose= TRUE),
                    model.FUN="TUWhydromod",
                    model.FUN.args= list(obs=Qobs.zoo.ver,
                                         dates=dates.ver,
                                         warmup.days=365,
                                         P=P.ver, Temp=Temp.ver, PET=PET.ver, Area=Area.m2
                                        )
                    )
```

In the previous code chunk, the arguments passed to `verification` have the same meaning as those of `hydroPSO` (more information can be found with `?verification`), with the exception of:

*) **par**: `data.frame` with all the parameter sets that will be run with `TUWmodel`. In this case, `params` was already read using a behavioural threshold equal to `beh.thr`.

10.3 Uncertainty analysis of verification results

10.3.1 Reading all verification results

In order to proceed with the uncertainty analysis, we need to read all the verification results generated by the previous call to the `verification` function:

```
gofs           <- out[["gofs"]]
Qsims          <- out[["model.values"]]
best.param.ver <- out[["best.param"]]
best.gof       <- out[["best.gof"]]
```

10.3.2 Weighted quantiles of model simulations

The computation of weighted quantiles of model simulations would provide an estimate of the uncertainty in each model simulated value (i.e., each daily streamflow), based on the goodness-of fit values obtained during the verification for the full time series simulated by the model. User-defined weighted quantiles for each model simulation are obtained multiplying the standard quantile derived for each model simulation based on all the model simulations (i.e., for each column in `Qsims`) by the corresopnding goodness-of-fit value obtained by the full time series simulated by the model during the verification period (*gofs*). For this tutorial, we are going use the `wquantile` function of `hydroPSO` to compute the 2.5, 50 and 97.5 weighted quantiles for each value simulated by the model, by setting `probs=c(0.025, 0.5, 0.975)`:

```
n     <- ncol(Qsims)     # number of time steps
Qsims <- Qsims[, 366:n] # removing the first year as warming up period
```

```
Qsim.025.q50.q975   <- wquantile(Qsims, weights=gofs, byrow=FALSE, probs=c(0.025, 0.5, 0.975),
                                 normwt=TRUE, verbose=FALSE)
round( head(Qsim.025.q50.q975), 3)
```

```
FALSE         2.5%   50% 97.5%
FALSE par366 0.391 1.315 2.276
FALSE par367 0.384 1.304 2.241
FALSE par368 0.382 1.291 2.216
FALSE par369 0.373 1.278 2.200
FALSE par370 0.368 1.262 2.180
FALSE par371 0.354 1.248 2.164
```

To plot the uncertainty bounds for each value simulated during the verification period, we are going to transform the weighted quantiles 2.5 and 97.5 into zoo objects:

```
dates.ver <- dip(Ver.Ini, Ver.Fin)
q025      <- zoo(Qsim.025.q50.q975[,1], dates.ver)
q975      <- zoo(Qsim.025.q50.q975[,3], dates.ver)
```

10.3.3 P-factor and R-factor

Now we are going to compute the **P-factor** (Schuol et al. 2008; Abbaspour et al. 2009), which represents the percent of observations that are within the user-defined uncertainty bounds:

```
pf <- pfactor(x=Qobs.zoo.ver, lband=q025, uband=q975, na.rm=TRUE)
pf
```

```
## [1] 0.7158935
```

Also, we are going to compute the **R-factor** (Schuol et al. 2008; Abbaspour et al. 2009), which represents the average width of the user-defined uncertainty bounds divided by the standard deviation of the observations:

```
rf <- rfactor(x=Qobs.zoo.ver, lband=q025, uband=q975, na.rm=TRUE)
rf
```

```
## [1] 0.8564446
```

A good model should bracket most of the measured data (i.e., large P-factor, with a maximum of 1) with the smallest possible R-factor (minimum of 0).

10.3.4 95 Percent Prediction Uncertainty (95 PPU)

Now we are ready to plot the *best* simulated streamflows obtained during the verification along with the **95 Percent Prediction Uncertainty** (**95 PPU**) (Abbaspour et al. 2007, 2009; Schuol et al. 2008):

```
main.uncert  <- paste0("Uncertainty bounds TUWmodel: ",
                        Qobs.stationname, " (", Qobs.ID, ")" )
Qobs.col     <- "black"
best.sim.col <- "blue"
bands.col    <- "lightblue"

plot(Qobs.zoo.ver, xaxt="n", xlab="", type="n", main=main.uncert, ylab="Q, [mm]")  # empty plotting are
drawTimeAxis(Qobs.zoo.ver)                                     # time axis
plotbandsonly(lband=q025, uband=q975)                          # 95 PPU
lines(Qobs.zoo.ver, col=Qobs.col, lwd=0.3)                     # Qobs
lines(Qsim.zoo.ver, col=best.sim.col, lwd=0.3)                 # best simulation during the verificati
grid()

legend("topright", legend=c("Qobs", "best.sim", "95PPU"), lty=c(1, 1, NA),
       pch=c(NA, NA, 15), col=c(Qobs.col, best.sim.col, bands.col),
       bty="n", cex = 1.2)

legend("topleft", legend=c(paste0("P-factor: ", round(pf, 2)),
                           paste0("R-factor: ", round(rf, 2)) ),
       bty="n", cex = 1.2)
```

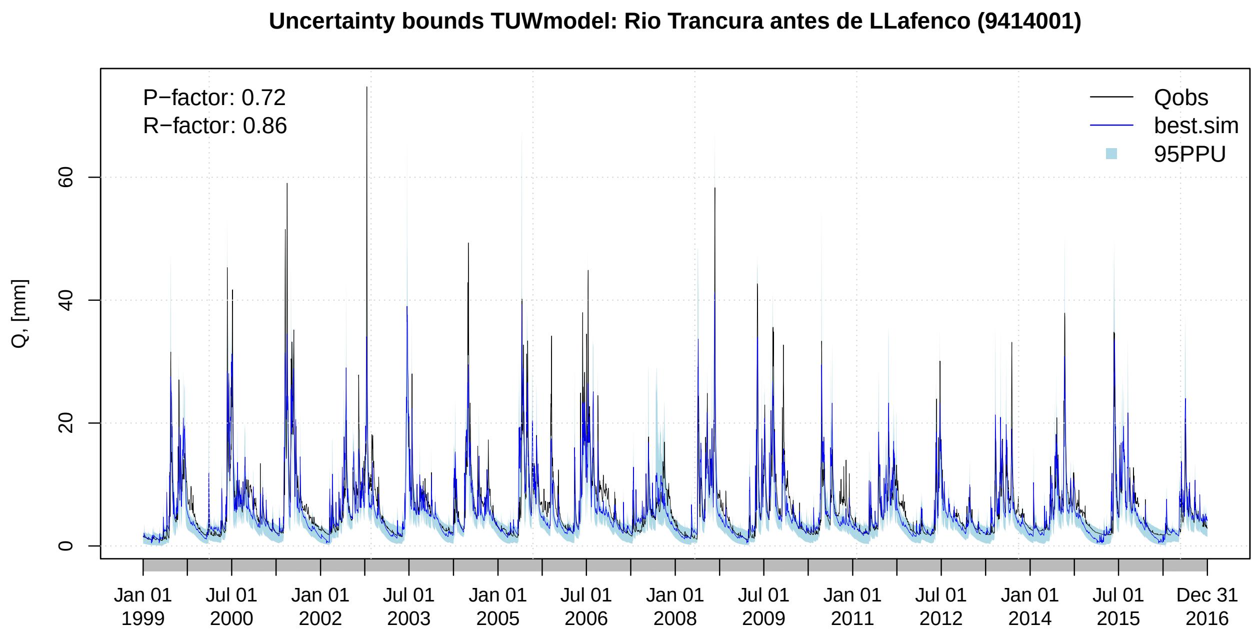


Figure 27: 95 Percent Prediction Uncertainty (95 PPU) for model simulations during the verification period.

10.3.5 Uncertainty in flow duration curve

Now we are going to use the `fdcu` included in the `hydroTSM` R package to plot the observed flow duration curve (FDC), the one of the *best* simulated streamflows, and the flow duration curves for the 2.5 and 97.5 weighted quantiles of model simulations obained during the verification period:

```
main.uncert  <- paste0("Flow duration curves GR4J model: ", Qobs.stationname, " (", Qobs.ID, ")" )
bands.col    <- "lightskyblue1"

par(mfrow=c(3,1))
fdcu(x=Qobs.zoo.ver, lband=q025, uband=q975, sim=Qsim.zoo.ver, bands.col=bands.col,
     main=main.uncert, log="") # empty plotting area
grid()
fdcu(x=Qobs.zoo.ver, lband=q025, uband=q975, sim=Qsim.zoo.ver, bands.col=bands.col,
     main=main.uncert, log="y") # empty plotting area
grid()
fdcu(x=Qobs.zoo.ver, lband=q025, uband=q975, sim=Qsim.zoo.ver, bands.col=bands.col,
     main=main.uncert, log="x") # empty plotting area
grid()
```

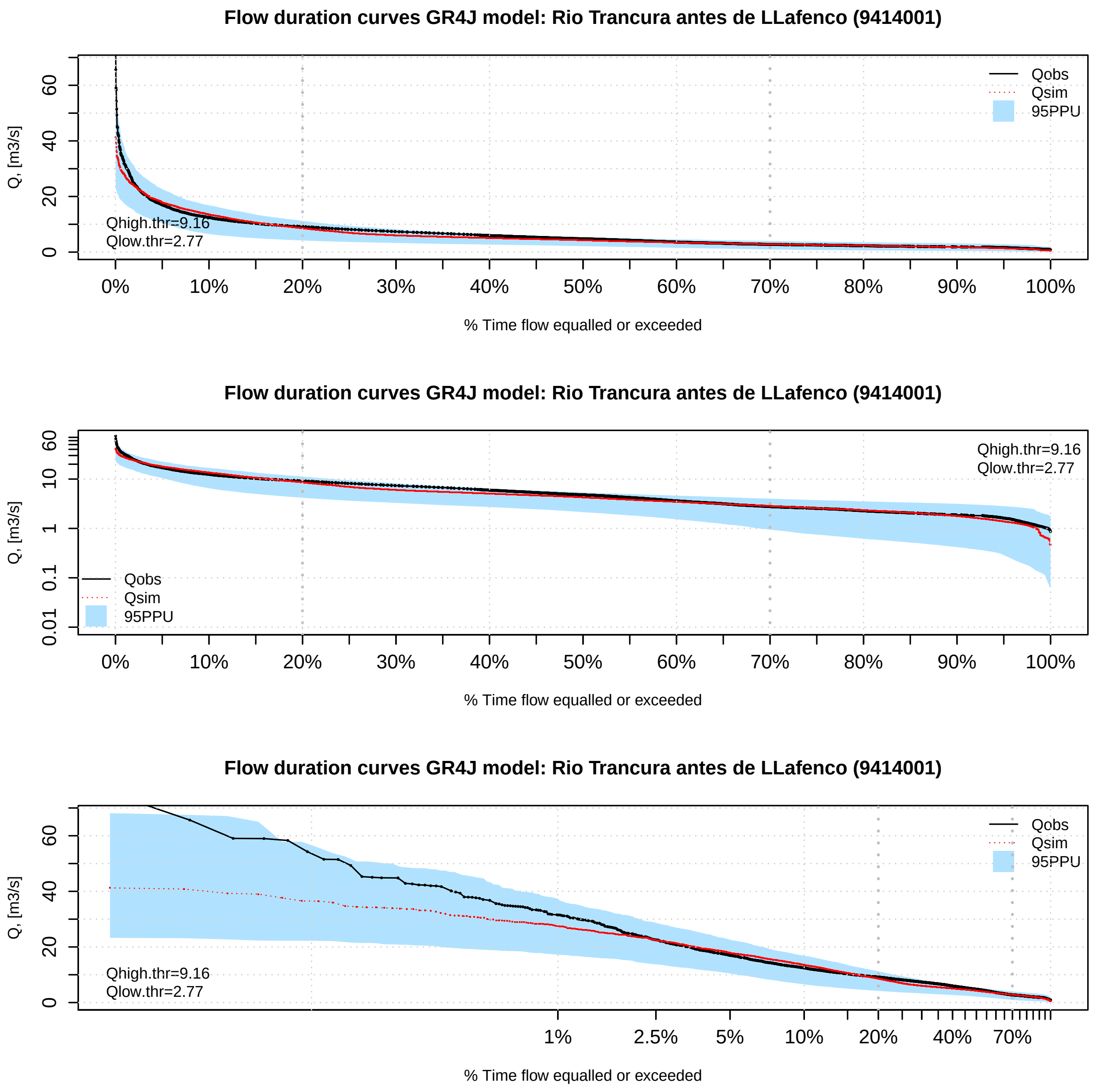


Figure 28: Flow duration curve of the observed (black line) and best simulated (blue line) streamflows. In addition, flow duration curves for the 2.5 and 97.5 weighted quantiles of model simulations obained during the verification period. The upper panel is the normal flow duration curve, the middle panel has focus on low flows (log=‘y’) and the lower panel has focus on high flows (log=‘x’).

11 FAQ

1) **How can I paralellise my computations?**

You should use `parallel='parallel'` or `parallel='parallelWin'` in the `control` argument of the `hydroPSO` (Section 9.3) and/or `verification` (Section 10.2) functions.

2) **How can I calibrate only a reduced number of parameters?**

Let's say that after looking at the figures obtained for the model parameters shown in Section 9.5.2.2 you want to re-calibrate `TUWmodel` with all the parameter but `DDF` (position 2 in `param.values`) and `k2` (position 11 in `param.values`), and you fix their values in `DDF=3` and `k2=140`. For re-calibrating the remaining 13 parameters you should re-define the lower and upper bound of your model parameters and the wrapper function you use for running your hydrological model (Section 9.2) as follows:

```
names <- c("SCF", "Tr", "Ts", "Tm", "LPrat", "FC", "Beta",
           "k0", "k1",  "lsuz", "cperc", "bmax", "croute")
lower <- c(0.9,   1.0,  -3.0, -2.0,   0.0,     0,     0,
             0,     2,     1,    0,     0,     0)
upper <- c(1.5,   3.0,   1.0,  2.0,   1.0,   600,    20,
             2,    30,   100,    8,    30,    50)

names(lower) <- names
names(upper) <- names
```

```
TUWhydromod.basic <- function(param.values,   # Now it has 13 parameters intead of 15
                              obs=Qobs.zoo.cal
                             ) {

  # "SCF" "Tr" "Ts" "Tm" "LPrat" "FC" "Beta" "k0" "k1" "lsuz" "cperc" "bmax" "croute"
  param.values2 <- c(param.values[1], 3, param.values[2:10], 140, param.values[11:13])

  # Evaluating the hydrological model at the 'param.values' parameter set
  # P.cal, Temp.cal, PET.cal, and Area.m2 are used hee as *global variables*
  # (not good programming style, but it works)
   simLump  <- TUWmodel(param=param.values2, prec=P.cal, airt=Temp.cal, ep=PET.cal, area=Area.m2)

  # Extracting only the numeric vector with simulations and observations
  qsim.full <- as.numeric(simLump$q)
  obs       <- as.numeric(obs)

  # Removing the warming up period (1 year, i.e., 365 days)
  n        <- length(qsim.full)
  qsim     <- qsim.full[366:n]
  obs      <- obs[366:n]

  # Computing the goodness-of-fit value (i.e, model performance)
  gof      <- KGE(sim=qsim, obs=obs, method="2012")

  # Creating the output of the R function
```

```r
  nelements <- 2
  out       <- vector("list", nelements)
  out[[1]]  <- gof
  out[[2]]  <- qsim.full

  # Mandatory names for the elements of the output
  names(out)[1:nelements] <- c("GoF", "sim")

  return(out)

} # 'TUWhydromod.basic' end
```

and then run your model with:

```r
out <- hydroPSO(fn="hydromodInR",
                lower=lower,
                upper=upper,
                method="spso2011",
                control=list(write2disk=TRUE, MinMax="max", npart=80,
                             maxit=50, normalise=TRUE, REPORT=10, parallel="none",
                             reltol=1E-10),
                model.FUN="TUWhydromod.basic"
                )
```

3) **Why do I get some warnings after the calibration?**

It is likely that you get sone or more of the following warning after finishing the calibration of `TUWmodel` with `hydroPSO`:

> There are no pairs of 'sim' and 'obs' without missing values !

This is due to the fact that some parameter sets returns `NA` when run with `TUWmodel`. You might want to have a look to the `./PSO.out/Particles.txt` and `./PSO.out/Model_out.txt` files, to get the values of the parameter sets and the model outputs, respectively, that returned `NA` values.

Parameter sets that gives place to `NA` during the optimisation are ignored during the calibration by `hydroPSO`, as you can verify comparing lines 20 (*Total model calls*) and 21 (*Effective model calls*) of the `./PSO.out/hydroPSO_logfile.txt` output file.

If you want to know the reason why a particular parameter set gives place to `NA`, please contact the mantainer of the `TUWmodel` package.

4) **Why the KGE values shown by ggof does not match the one obtained during the optimisation?**

The KGE value shown by `ggof` uses `method="2009"`. If the KGE you got during the optimisation is different from the one shown by `ggof`, very likely is due to the fact that you used `method="2012"` for the computation of KGE during the calibration.

5) **Why do I get different results (i.e., parameter sets and goodness-of-fit values) every time that I run `hydroPSO` with the same input data?**

As mentioned in Section 5, `hydroPSO` is a stochastic optimisation technique and, therefore, the initialization of particles' position and the exploration of the parameter space is different every time that you run `hydroPSO`. If you want to get the same results every time that you run `hydroPSO` (i.e., reproducibility) you should use `set.seed` with some numeric argument (e.g., `set.seed(100)`) just before calling `hydroPSO`.

12 Version history

-) v0.9.3: 27-April, 2020

-) v0.9.2: 18-March, 2020

-) v0.9.1: 17-March, 2020

-) v0.9: 16-March, 2020

-) v0.8: 30-Jan, 2020

-) v0.7: 30-Dec, 2019

-) v0.6: December 2017

-) v0.5: March 2017

-) v0.4: March 2016

-) v0.3: March 2015

-) v0.2: March 2014

-) v0.1: March 2013

References

Abbaspour, Karim C., Monireh Faramarzi, Samaneh Seyed Ghasemi, and Hong Yang. 2009. "Assessing the Impact of Climate Change on Water Resources in Iran." *Water Resources Research* 45 (10): W10434. https://doi.org/10.1029/2008WR007615.

Abbaspour, Karim C., Jing Yang, Ivan Maximov, Rosi Siber, Konrad Bogner, Johanna Mieleitner, Juerg Zobrist, and Raghavan Srinivasan. 2007. "Modelling Hydrology and Water Quality in the Pre-Alpine/Alpine Thur Watershed Using SWAT." *Journal of Hydrology* 333 (2-4): 413–30. https://doi.org/10.1016/j.jhydrol.2006.09.014.

Abdelaziz, Ramadan, Broder J Merkel, Mauricio Zambrano-Bigiarini, and Sreejesh Nair. 2019. "Particle Swarm Optimization for the Estimation of Surface Complexation Constants with the Geochemical Model PHREEQC-3.1.2." *Geoscientific Model Development* 12 (1). https://doi.org/10.5194/gmd-12-167-2019.

Abdelaziz, Ramadan, and Mauricio Zambrano-Bigiarini. 2014. "Particle Swarm Optimization for Inverse Modeling of Solute Transport in Fractured Gneiss Aquifer." *Journal of Contaminant Hydrology* 164: 285–98. https://doi.org/10.1016/j.jconhyd.2014.06.003.

Alvarez-Garreton, Camila, Pablo A Mendoza, Juan P Boisier, Nans Addor, Mauricio Galleguillos, Mauricio Zambrano-Bigiarini, Antonio Lara, et al. 2018. "The CAMELS-CL Dataset: Catchment Attributes and Meteorology for Large Sample Studies-Chile Dataset." *Hydrology and Earth System Sciences* 22 (11). Copernicus Publications: 5817–46. https://doi.org/10.5194/hess-22-5817-2018.

Bergström, S. 1995. "The Hbv Model." *Computer Models of Watershed Hydrology*. Water Resources Publications.

Beven, Keith J. 2006. "A Manifesto for the Equifinality Thesis." *Journal of Hydrology* 320 (1-2): 18–36. https://doi.org/10.1016/j.jhydrol.2005.07.007.

Beven, Keith J., and A Binley. 1992. "The Future of Distributed Models - Model Calibration and Uncertainty Prediction." *Hydrological Processes* 6 (3): 279–98. https://doi.org/10.1002/hyp.3360060305.

Beven, Keith J., and Jim Freer. 2001. "Equifinality, Data Assimilation, and Uncertainty Estimation in Mechanistic Modelling of Complex Environmental Systems Using the GLUE Methodology." *Journal of Hydrology* 249 (1-4): 11–29. https://doi.org/10.1016/S0022-1694(01)00421-8.

Beven, Keith J., Paul J. Smith, and Jim E. Freer. 2008. "So Just Why Would a Modeller Choose to Be Incoherent ?" *Journal of Hydrology* 354 (1-4): 15. https://doi.org/10.1016/j.jhydrol.2008.02.007.

Ceola, Serena, Berit Arheimer, E Baratti, G Blöschl, Réne Capell, Attilio Castellarin, Jim Freer, et al. 2015. "Virtual Laboratories: New Opportunities for Collaborative Water Science." *Hydrology and Earth System Sciences* 19 (4): 2101–17.

Cisty, Milan, and Veronika Soldanova. 2018. "Flow Prediction Versus Flow Simulation Using Machine Learning Algorithms." In *International Conference on Machine Learning and Data Mining in Pattern Recognition*, 369–82. Springer.

Gupta, Hoshin V, Harald Kling, Koray K Yilmaz, and Guillermo F Martinez. 2009. "Decomposition of the Mean Squared Error and Nse Performance Criteria: Implications for Improving Hydrological Modelling." *Journal of Hydrology* 377 (1-2). Elsevier: 80–91.

Jin, Xiaoli, Chong-Yu Xu, Qi Zhang, and V.P. Singh. 2010. "Parameter and Modeling Uncertainty Simulated by GLUE and a Formal Bayesian Method for a Conceptual Hydrological Model." *Journal of Hydrology* 383 (3-4): 147–55. https://doi.org/10.1016/j.jhydrol.2009.12.028.

Kennedy, James, and Russell Eberhart. 1995. "Particle Swarm Optimization." In *Proceedings of Icnn'95-International Conference on Neural Networks*, 4:1942–8. IEEE.

Kling, Harald, Martin Fuchs, and Maria Paulin. 2012. "Runoff Conditions in the Upper Danube Basin Under an Ensemble of Climate Change Scenarios." *Journal of Hydrology* 424. Elsevier: 264–77.

Melsen, L. A., N. Addor, N. Mizukami, A. J. Newman, P. J. J. F. Torfs, M. P. Clark, R. Uijlenhoet, and A. J. Teuling. 2018. "Mapping (Dis)agreement in Hydrologic Projections." *Hydrology and Earth System Sciences* 22 (3): 1775–91. https://doi.org/10.5194/hess-22-1775-2018.

Nijzink, RC, Susana Almeida, IG Pechlivanidis, Rene Capell, D Gustafssons, Berit Arheimer, Juraj Parajka, et al. 2018. "Constraining Conceptual Hydrological Models with Multiple Information Sources." *Water Resources Research* 54 (10). Wiley Online Library: 8332–62.

Nijzink, Remko, Christopher Hutton, Ilias Pechlivanidis, René Capell, Berit Arheimer, Jim Freer, Dawei Han, et al. 2016. "The Evolution of Root-Zone Moisture Capacities After Deforestation: A Step Towards Hydrological Predictions Under Change?" *Hydrology and Earth System Sciences* 20 (12): 4775–99.

Parajka, J., R. Merz, and G. Blöschl. 2007. "Uncertainty and Multiple Objective Calibration in Regional Water Balance Modelling: Case Study in 320 Austrian Catchments." *Hydrological Processes* 21 (4): 435–46. https://doi.org/10.1002/hyp.6253.

Parajka, Juraj, Alfred Paul Blaschke, Günter Blöschl, Klaus Haslinger, Gerold Hepp, Gregor Laaha, Wolfgang Schöner, Helene Trautvetter, Alberto Viglione, and Matthias Zessner. 2016. "Uncertainty Contributions to Low-Flow Projections in Austria." *Hydrology and Earth System Sciences* 20 (5). Copernicus GmbH: 2085–2101.

Refsgaard, Jens Christian, Jeroen P. van der Sluijs, Anker Lajer Højberg, and Peter A. Vanrolleghem. 2007. "Uncertainty in the Environmental Modelling Process - a Framework and Guidance." *Environmental Modelling & Software* 22 (11): 1543–56. https://doi.org/10.1016/j.envsoft.2007.02.004.

Rojas, Rodrigo, and Mauricio Zambrano-Bigiarini. 2012. *Tutorial for Interfacing hydroPSO with SWAT-2005 and MODFLOW-2005.* https://doi.org/10.5281/zenodo.3701891.

Schuol, Jürgen, Karim C. Abbaspour, Raghavan Srinivasan, and Hong Yang. 2008. "Estimation of Freshwater Availability in the West African Sub-Continent Using the SWAT Hydrologic Model." *Journal of Hydrology* 352 (1-2): 30. https://doi.org/10.1016/j.jhydrol.2007.12.025.

Sleziak, Patrik, Ján Szolgay, Kamila Hlavčová, Doris Duethmann, Juraj Parajka, and Michal Danko. 2018. "Factors Controlling Alterations in the Performance of a Runoff Model in Changing Climate Conditions." *Journal of Hydrology and Hydromechanics* 66 (4). Sciendo: 381–92.

Sleziak, P, J Szolgay, K Hlavčová, M Danko, and J Parajka. 2020. "The Effect of the Snow Weighting on the Temporal Stability of Hydrologic Model Efficiency and Parameters." *Journal of Hydrology.* Elsevier, 124639.

Sleziak, P, J Szolgay, K Hlavčová, and J Parajka. 2016. "The Impact of the Variability of Precipitation and Temperatures on the Efficiency of a Conceptual Rainfall-Runoff Model." *Slovak Journal of Civil Engineering* 24 (4). De Gruyter Open: 1–7.

Viglione, Alberto, and Juraj Parajka. 2019. *TUWmodel: Lumped/Semi-Distributed Hydrological Model for Education Purposes.* https://CRAN.R-project.org/package=TUWmodel.

Zambrano-Bigiarini, Mauricio. 2020. *hydroTSM: Time Series Management, Analysis and Interpolation for Hydrological Modelling. R Package Version 0.6-0. doi:10.5281/Zenodo.83964. https://github.com/hzambran/hydroTSM.* https://github.com/hzambran/hydroTSM.

Zambrano-Bigiarini, Mauricio. 2020. *hydroGOF: Goodness-of-Fit Functions for Comparison of Simulated and Observed Hydrological Time Series. R Package Version 0.4-0. doi:10.5281/Zenodo.839854. https://github.com/hzambran/hydroGOF.* https://github.com/hzambran/hydroGOF.

Zambrano-Bigiarini, Mauricio, and Oscar M. Baez-Villanueva. 2020. *Tutorial for Using hydroPSO to Calibrate TUWmodel. Version 0.6.* https://doi.org/10.5281/zenodo.3701903.

Zambrano-Bigiarini, Mauricio, Maurice Clerc, and Rodrigo Rojas. 2013. "Standard Particle Swarm Optimisation 2011 at CEC-2013: A Baseline for Future PSO Improvements." In *2013 IEEE Congress on Evolutionary Computation*, 2337–44. IEEE. https://doi.org/10.1109/CEC.2013.6557848.

Zambrano-Bigiarini, Mauricio, and Rodrigo Rojas. 2013. “A Model-Independent Particle Swarm Optimisation Software for Model Calibration.” *Environmental Modelling & Software* 43: 5–25. https://doi.org/10.1016/j.envsoft.2013.01.004.

Zessner, Matthias, Martin Schönhart, Juraj Parajka, Helene Trautvetter, Hermine Mitter, Mathias Kirchner, Gerold Hepp, Alfred Paul Blaschke, Birgit Strenn, and Erwin Schmid. 2017. “A Novel Integrated Modelling Framework to Assess the Impacts of Climate and Socio-Economic Drivers on Land Use and Water Quality.” *Science of the Total Environment* 579. Elsevier: 1137–51.