Goodness-of-fit measures to compare observed and simulated time series with hydroGOF

Mauricio Zambrano-Bigiarini*

version 0.3, 20-Jan-2024

1 Installation

Installing the latest stable version (from CRAN):

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

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

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

2 Setting up the environment

Loading the *hydroGOF* package, which contains data and functions used in this analysis:

```
library(hydroGOF)
```

```
## Loading required package: zoo

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

3 Example using NSE

The following examples use the well-known Nash-Sutcliffe efficiency (NSE), but you can repeat the computations using any of the goodness-of-fit measures included in the *hydroGOF* package (e.g., KGE, ubRMSE, dr).

3.1 Example 1

Basic ideal case with a numeric sequence of integers:

```
obs <- 1:10
sim <- 1:10
NSE(sim, obs)
```

*mauricio.zambrano@ufrontera.cl

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

```r
obs <- 1:10
sim <- 2:11
NSE(sim, obs)
```

```
## [1] 0.8787879
```

3.2 Example 2

From this example onwards, a streamflow time series will be used.

First, we load the daily streamflows of the Ega River (Spain), from 1961 to 1970:

```r
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts
```

Generating a simulated daily time series, initially equal to the observed series:

```r
sim <- obs
```

Computing the 'NSE' for the "best" (unattainable) case

```r
NSE(sim=sim, obs=obs)
```

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

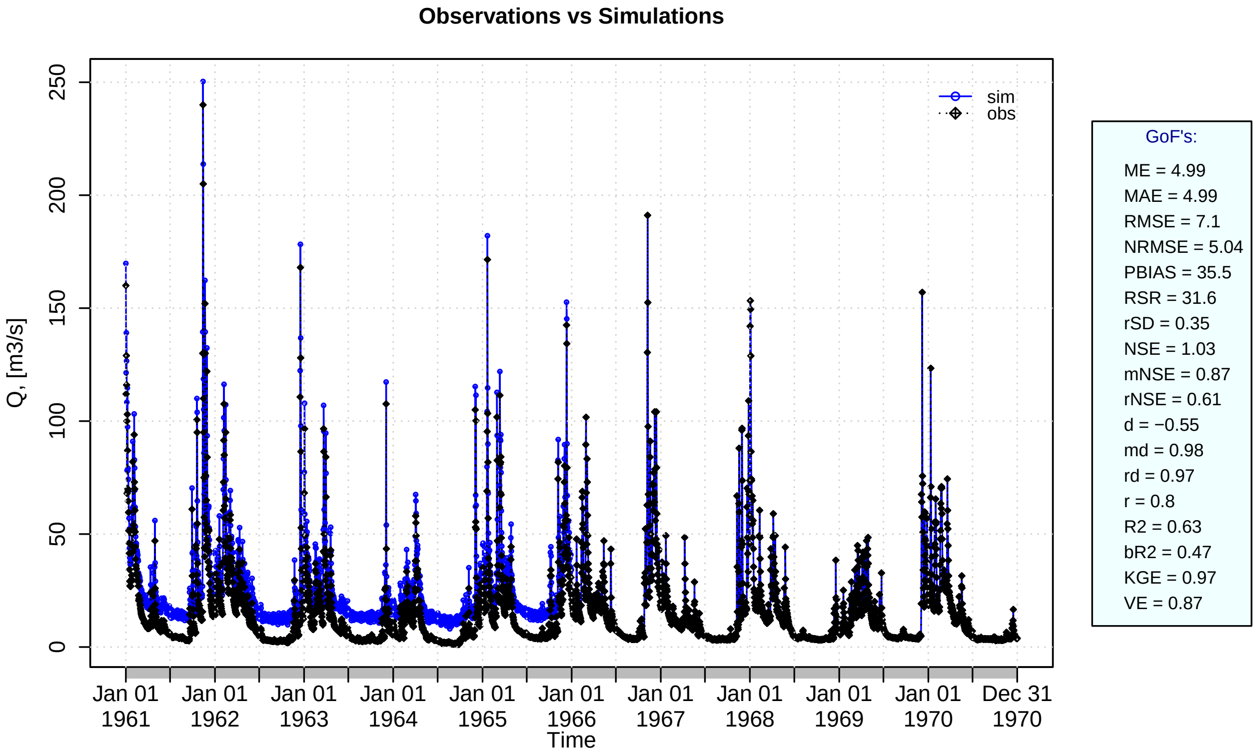
3.3 Example 3

NSE for simulated values equal to observations plus random noise on the first half of the observed values.

This random noise has more relative importance for low flows than for medium and high flows.

Randomly changing the first 1826 elements of 'sim', by using a normal distribution with mean 10 and standard deviation equal to 1 (default of 'rnorm').

```r
sim[1:1826] <- obs[1:1826] + rnorm(1826, mean=10)
ggof(sim, obs)
```




```r
NSE(sim=sim, obs=obs)
```

```
## [1] 0.8742723
```

Let's have a look at other goodness-of-fit measures:

```r
mNSE(sim=sim, obs=obs)                # modified NSE
```

```
## [1] 0.6054552
```

```r
rNSE(sim=sim, obs=obs)                # relative NSE
```

```
## [1] -0.5500683
```

```r
KGE(sim=sim, obs=obs)                 # Kling-Gupta efficiency (KGE), 2009
```

```
## [1] 0.6809152
```

```r
KGE(sim=sim, obs=obs, method="2012") # Kling-Gupta efficiency (KGE), 2012
```

```
## [1] 0.6176491
```

```r
KGElf(sim=sim, obs=obs)               # KGE for low flows
```

```
## [1] 0.5177778
```

```r
KGEnp(sim=sim, obs=obs)               # Non-parametric KGE
```

```
## [1] 0.633923
```

```r
sKGE(sim=sim, obs=obs)                # Split KGE
```

```
## [1] 0.6546898
```

```r
d(sim=sim, obs=obs)                   # Index of agreement (d)
```

```
## [1] 0.9698176
```

```r
rd(sim=sim, obs=obs)                    # Relative d
```

```
## [1] 0.6278882
```

```r
md(sim=sim, obs=obs)                    # Modified d
```

```
## [1] 0.7982968
```

```r
dr(sim=sim, obs=obs)                    # Refined d
```

```
## [1] 0.8027276
```

```r
VE(sim=sim, obs=obs)                    # Volumetric efficiency
```

```
## [1] 0.6842508
```

```r
cp(sim=sim, obs=obs)                    # Coefficient of persistence
```

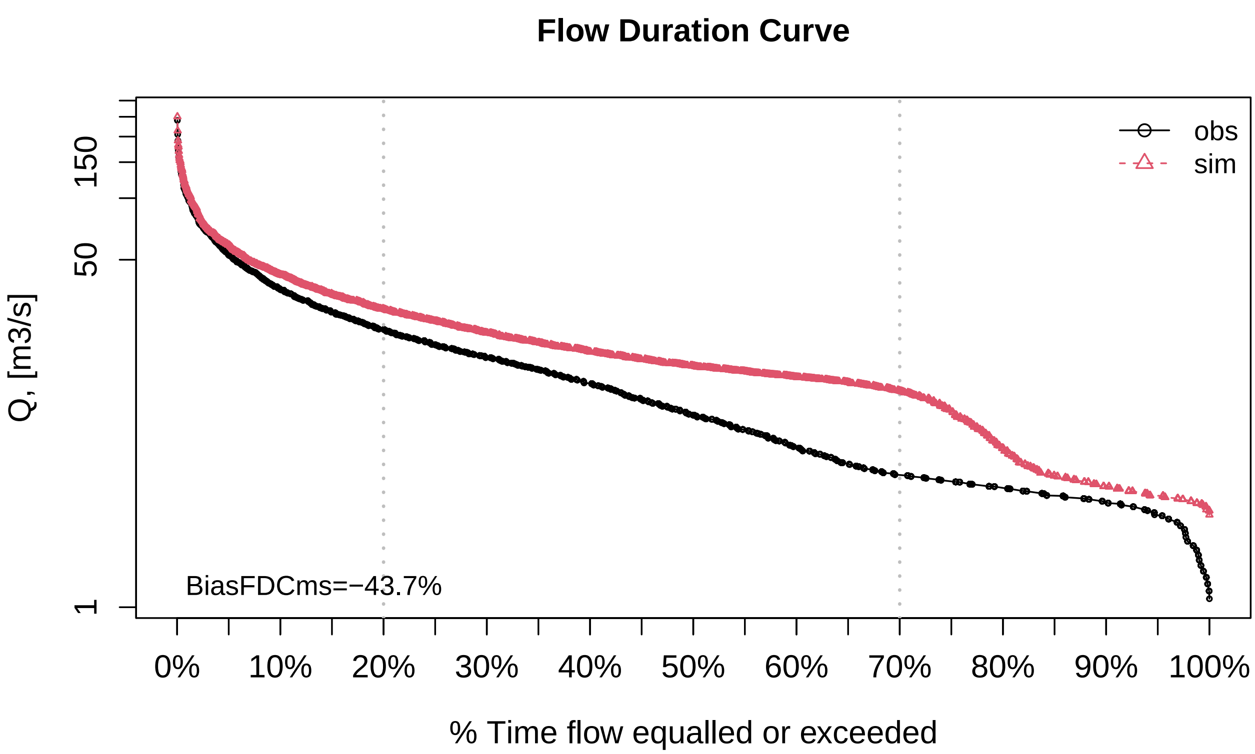
```
## [1] 0.4696152
```

```r
pbias(sim=sim, obs=obs)                 # Percent bias (PBIAS)
```

```
## [1] 31.6
```

```r
pbiasfdc(sim=sim, obs=obs)              # PBIAS in the slope of the midsegment of the FDC
```

```
## [Note: 'thr.shw' was set to FALSE to avoid confusing legends...]
```




```
## [1] -43.70342
```

```r
rmse(sim=sim, obs=obs)                  # Root mean square error (RMSE)
```

```
## [1] 7.09606
```

```r
ubRMSE(sim=sim, obs=obs)                # Unbiased RMSE
```

```
## [1] 5.042509
```

```r
rPearson(sim=sim, obs=obs)          # Pearson correlation coefficient
```

```
## [1] 0.9699098
```

```r
rSpearman(sim=sim, obs=obs)         # Spearman rank correlation coefficient
```

```
## [1] 0.8353165
```

```r
R2(sim=sim, obs=obs)                # Coefficient of determination (R2)
```

```
## [1] 0.8742723
```

```r
br2(sim=sim, obs=obs)               # R2 multiplied by the slope of the regression line
```

```
## [1] 0.7780847
```

3.4 Example 4:

NSE for simulated values equal to observations plus random noise on the first half of the observed values and applying (natural) logarithm to 'sim' and 'obs' during computations.

```r
NSE(sim=sim, obs=obs, fun=log)
```

```
## [1] 0.4814797
```

Verifying the previous value:

```r
lsim <- log(sim)
lobs <- log(obs)
NSE(sim=lsim, obs=lobs)
```

```
## [1] 0.4814797
```

Let's have a look at other goodness-of-fit measures:

```r
mNSE(sim=sim, obs=obs, fun=log)               # modified NSE
```

```
## [1] 0.4829788
```

```r
rNSE(sim=sim, obs=obs, fun=log)               # relative NSE
```

```
## [1] -4.478529
```

```r
KGE(sim=sim, obs=obs, fun=log)                # Kling-Gupta efficiency (KGE), 2009
```

```
## [1] 0.7165954
```

```r
KGE(sim=sim, obs=obs, method="2012", fun=log) # Kling-Gupta efficiency (KGE), 2012
```

```
## [1] 0.6360821
```

```r
KGElf(sim=sim, obs=obs)                       # KGE for low flows (it does not allow 'fun' argument)
```

```
## [1] 0.5177778
```

```r
KGEnp(sim=sim, obs=obs, fun=log)              # Non-parametric KGE
```

```
## [1] 0.7433967
```

```r
sKGE(sim=sim, obs=obs, fun=log)               # Split KGE
```

```
## [1] 0.4657589
```

```r
d(sim=sim, obs=obs, fun=log)                        # Index of agreement (d)
```

```
## [1] 0.8613334
```

```r
rd(sim=sim, obs=obs, fun=log)                       # Relative d
```

```
## [1] -0.465109
```

```r
md(sim=sim, obs=obs, fun=log)                       # Modified d
```

```
## [1] 0.7388478
```

```r
dr(sim=sim, obs=obs, fun=log)                       # Refined d
```

```
## [1] 0.7414894
```

```r
VE(sim=sim, obs=obs, fun=log)                       # Volumetric efficiency
```

```
## [1] 0.8126949
```

```r
cp(sim=sim, obs=obs, fun=log)                       # Coefficient of persistence
```

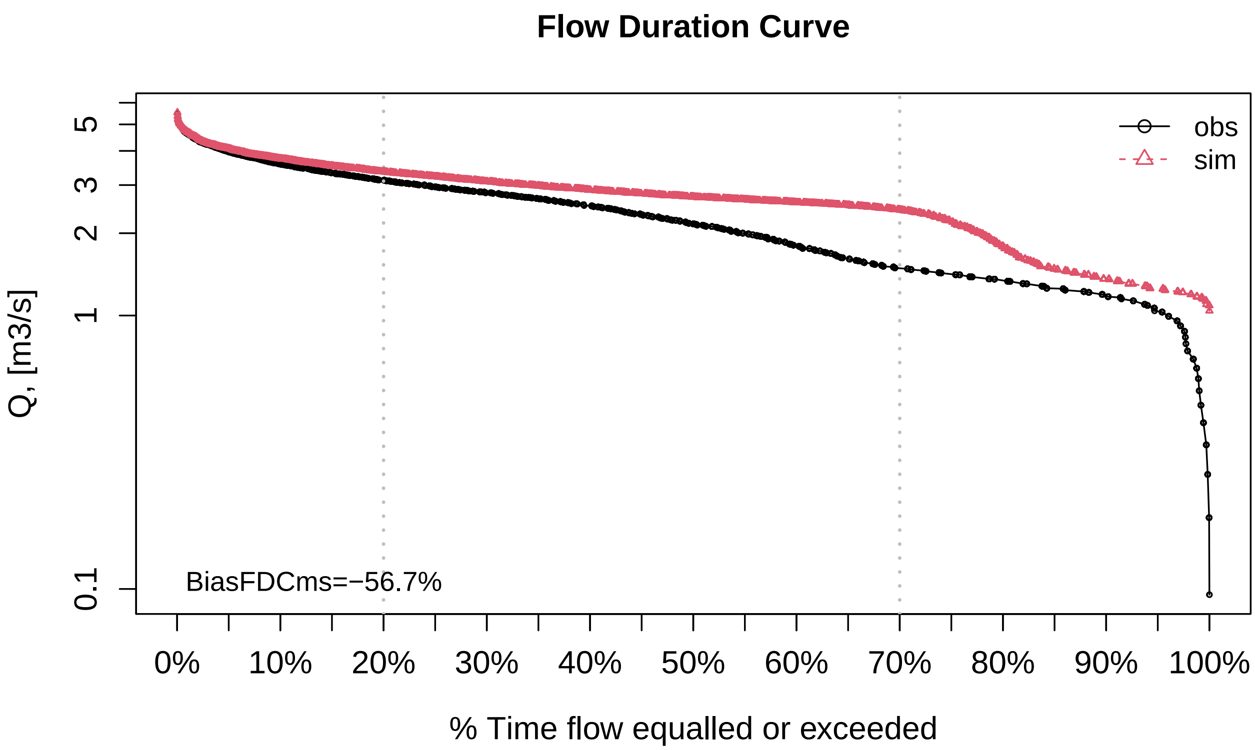
```
## [1] -7.921552
```

```r
pbias(sim=sim, obs=obs, fun=log)                    # Percent bias (PBIAS)
```

```
## [1] 18.7
```

```r
pbiasfdc(sim=sim, obs=obs, fun=log)                 # PBIAS in the slope of the midsegment of the FDC
```

```
## [Note: 'thr.shw' was set to FALSE to avoid confusing legends...]
```




```
## [1] -56.69516
```

```r
rmse(sim=sim, obs=obs, fun=log)                     # Root mean square error (RMSE)
```

```
## [1] 0.6945079
```

```r
ubRMSE(sim=sim, obs=obs, fun=log)              # Unbiased RMSE
```

```
## [1] 0.5511774
```

```r
rPearson(sim=sim, obs=obs, fun=log)            # Pearson correlation coefficient (r)
```

```
## [1] 0.8227826
```

```r
rSpearman(sim=sim, obs=obs, fun=log)           # Spearman rank correlation coefficient (rho)
```

```
## [1] 0.8353165
```

```r
R2(sim=sim, obs=obs, fun=log)                  # Coefficient of determination (R2)
```

```
## [1] 0.4814797
```

```r
br2(sim=sim, obs=obs, fun=log)                 # R2 multiplied by the slope of the regression line
```

```
## [1] 0.4314256
```

3.5 Example 5

NSE for simulated values equal to observations plus random noise on the first half of the observed values and applying (natural) logarithm to 'sim' and 'obs' and adding the Pushpalatha2012 constant during computations

```r
NSE(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012")
```

```
## [1] 0.4886352
```

Verifying the previous value, with the epsilon value following Pushpalatha2012:

```r
eps  <- mean(obs, na.rm=TRUE)/100
lsim <- log(sim+eps)
lobs <- log(obs+eps)
NSE(sim=lsim, obs=lobs)
```

```
## [1] 0.4886352
```

Let's have a look at other goodness-of-fit measures:

```r
gof(sim=sim, obs=obs, fun=log, epsilon.type="Pushpalatha2012", do.spearman=TRUE, do.pbfdc=TRUE)
```

```
##              [,1]
## ME           0.41
## MAE          0.41
## MSE          0.46
## RMSE         0.68
## ubRMSE       0.54
## NRMSE %     71.50
## PBIAS %     18.20
## RSR          0.71
## rSD          0.89
## NSE          0.49
## mNSE         0.48
## rNSE        -2.04
## wNSE         0.74
## d            0.86
## dr           0.74
## md           0.74
```

```
## rd             0.19
## cp            -7.65
## r              0.83
## R2             0.49
## bR2            0.44
## KGE            0.72
## KGElf          0.52
## KGEnp          0.74
## sKGE           0.53
## VE             0.82
## rSpearman      0.84
## pbiasFDC % -55.99
```

3.6 Example 6

NSE for simulated values equal to observations plus random noise on the first half of the observed values and applying (natural) logarithm to 'sim' and 'obs' and adding a user-defined constant during computations

```r
eps <- 0.01
NSE(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps)
```

```
## [1] 0.4819478
```

Verifying the previous value:

```r
lsim <- log(sim+eps)
lobs <- log(obs+eps)
NSE(sim=lsim, obs=lobs)
```

```
## [1] 0.4819478
```

Let's have a look at other goodness-of-fit measures:

```r
gof(sim=sim, obs=obs, fun=log, epsilon.type="otherValue", epsilon.value=eps, do.spearman=TRUE, do.pbfdc=
```

```
##              [,1]
## ME           0.42
## MAE          0.42
## MSE          0.48
## RMSE         0.69
## ubRMSE       0.55
## NRMSE %     72.00
## PBIAS %     18.70
## RSR          0.72
## rSD          0.88
## NSE          0.48
## mNSE         0.48
## rNSE        -4.14
## wNSE         0.74
## d            0.86
## dr           0.74
## md           0.74
## rd          -0.37
## cp          -7.90
## r            0.82
## R2           0.48
## bR2          0.43
```

```
## KGE          0.72
## KGElf        0.51
## KGEnp        0.74
## sKGE         0.48
## VE           0.81
## rSpearman    0.84
## pbiasFDC % -56.65
```

3.7 Example 7

NSE for simulated values equal to observations plus random noise on the first half of the observed values and applying (natural) logarithm to 'sim' and 'obs' and using a user-defined factor to multiply the mean of the observed values to obtain the constant to be added to 'sim' and 'obs' during computations

```
fact <- 1/50
NSE(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact)
```

```
## [1] 0.4953177
```

Verifying the previous value:

```
fact <- 1/50
eps  <- fact*mean(obs, na.rm=TRUE)
lsim <- log(sim+eps)
lobs <- log(obs+eps)
NSE(sim=lsim, obs=lobs)
```

```
## [1] 0.4953177
```

Let's have a look at other goodness-of-fit measures:

```
gof(sim=sim, obs=obs, fun=log, epsilon.type="otherFactor", epsilon.value=fact, do.spearman=TRUE, do.pbf
```

```
##              [,1]
## ME           0.41
## MAE          0.41
## MSE          0.44
## RMSE         0.66
## ubRMSE       0.52
## NRMSE %     71.00
## PBIAS %     17.60
## RSR          0.71
## rSD          0.89
## NSE          0.50
## mNSE         0.49
## rNSE        -1.31
## wNSE         0.74
## d            0.87
## dr           0.74
## md           0.74
## rd           0.39
## cp          -7.41
## r            0.83
## R2           0.50
## bR2          0.44
## KGE          0.73
## KGElf        0.53
```

```
## KGEnp        0.74
## sKGE         0.56
## VE           0.82
## rSpearman    0.84
## pbiasFDC % -55.31
```

3.8 Example 8

NSE for simulated values equal to observations plus random noise on the first half of the observed values and applying a user-defined function to 'sim' and 'obs' during computations:

```
fun1 <- function(x) {sqrt(x+1)}
NSE(sim=sim, obs=obs, fun=fun1)
```

```
## [1] 0.727312
```

Verifying the previous value, with the epsilon value following Pushpalatha2012:

```
sim1 <- sqrt(sim+1)
obs1 <- sqrt(obs+1)
NSE(sim=sim1, obs=obs1)
```

```
## [1] 0.727312
```

```
gof(sim=sim, obs=obs, fun=fun1, do.spearman=TRUE, do.pbfdc=TRUE)
```

```
##               [,1]
## ME            0.65
## MAE           0.65
## MSE           0.92
## RMSE          0.96
## ubRMSE        0.71
## NRMSE %      52.20
## PBIAS %      17.70
## RSR           0.52
## rSD           0.97
## NSE           0.73
## mNSE          0.54
## rNSE          0.34
## wNSE          0.89
## d             0.93
## dr            0.77
## md            0.76
## rd            0.83
## cp           -1.16
## r             0.92
## R2            0.73
## bR2           0.65
## KGE           0.81
## KGElf         0.51
## KGEnp         0.75
## sKGE          0.84
## VE            0.82
## rSpearman     0.84
## pbiasFDC % -40.96
```

4 A short example from hydrological modelling

Loading observed streamflows of the Ega River (Spain), with daily data from 1961-Jan-01 up to 1970-Dec-31

```
require(zoo)
data(EgaEnEstellaQts)
obs <- EgaEnEstellaQts
```

Generating a simulated daily time series, initially equal to the observed values (simulated values are usually read from the output files of the hydrological model)

```
sim <- obs
```

Computing the numeric goodness-of-fit measures for the "best" (unattainable) case

```
gof(sim=sim, obs=obs)
```

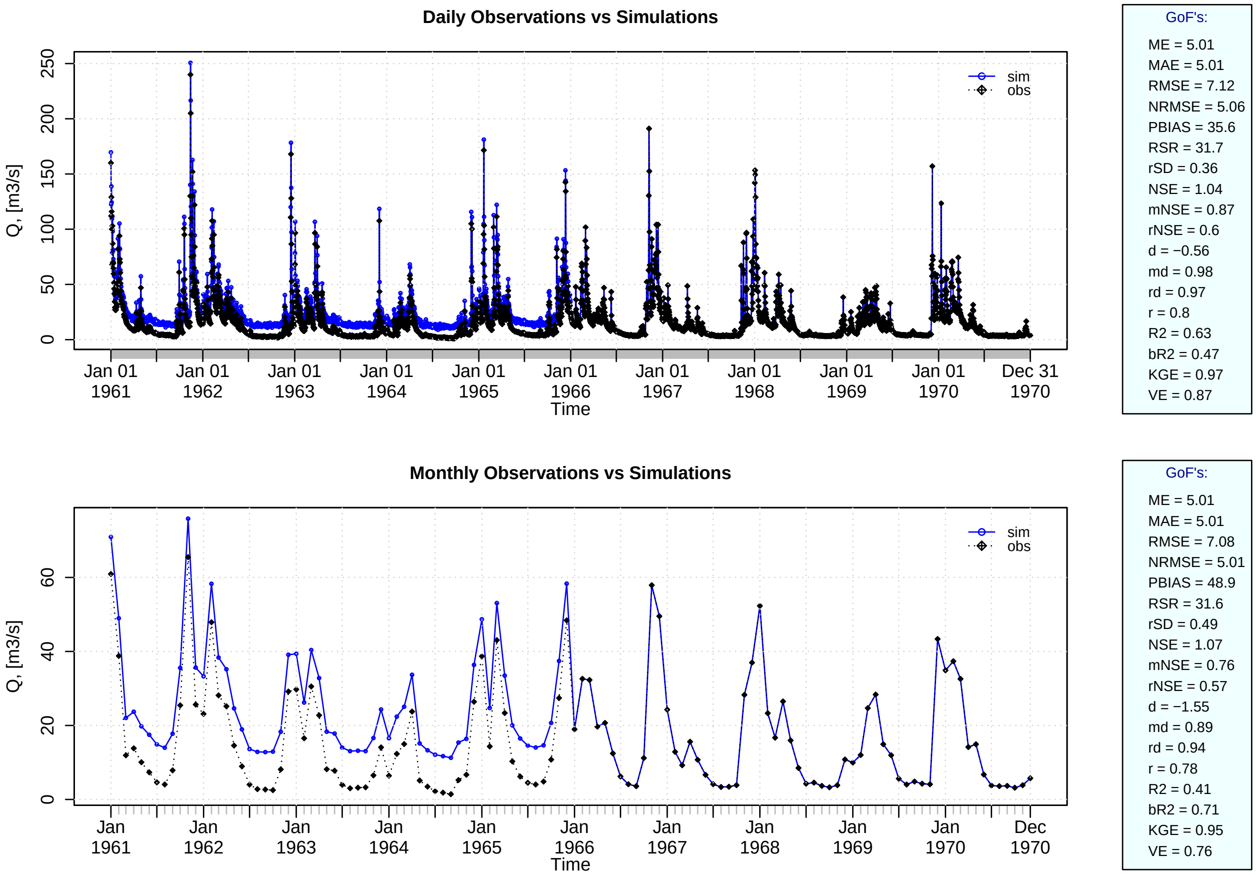
```
##          [,1]
## ME          0
## MAE         0
## MSE         0
## RMSE        0
## ubRMSE      0
## NRMSE %     0
## PBIAS %     0
## RSR         0
## rSD         1
## NSE         1
## mNSE        1
## rNSE        1
## wNSE        1
## d           1
## dr          1
## md          1
## rd          1
## cp          1
## r           1
## R2          1
## bR2         1
## KGE         1
## KGElf       1
## KGEnp       1
## sKGE        1
## VE          1
```

- Randomly changing the first 1826 elements of 'sim' (half of the ts), by using a normal distribution with mean 10 and standard deviation equal to 1 (default of 'rnorm').

```
sim[1:1826] <- obs[1:1826] + rnorm(1826, mean=10)
```

Plotting the graphical comparison of 'obs' against 'sim', along with the numeric goodness-of-fit measures for the daily and monthly time series

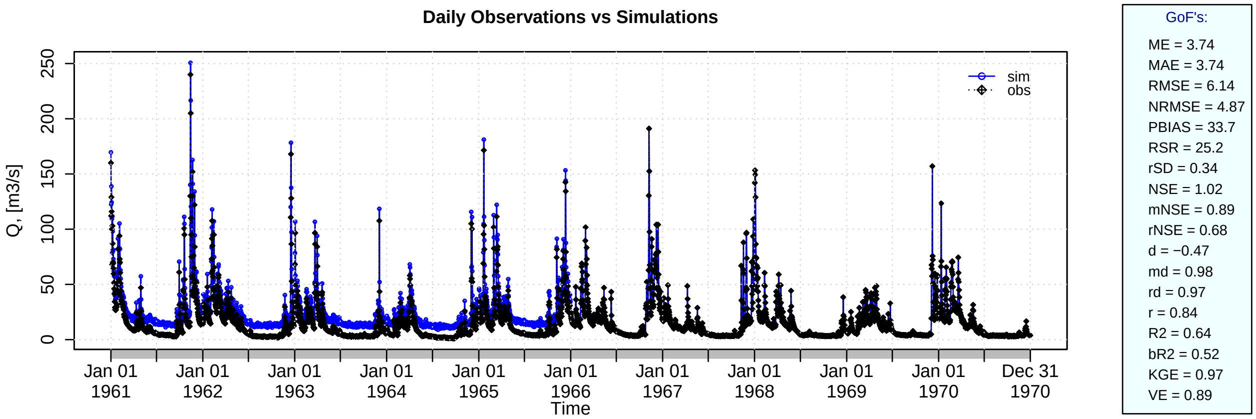
```r
ggof(sim=sim, obs=obs, ftype="dm", FUN=mean)
```

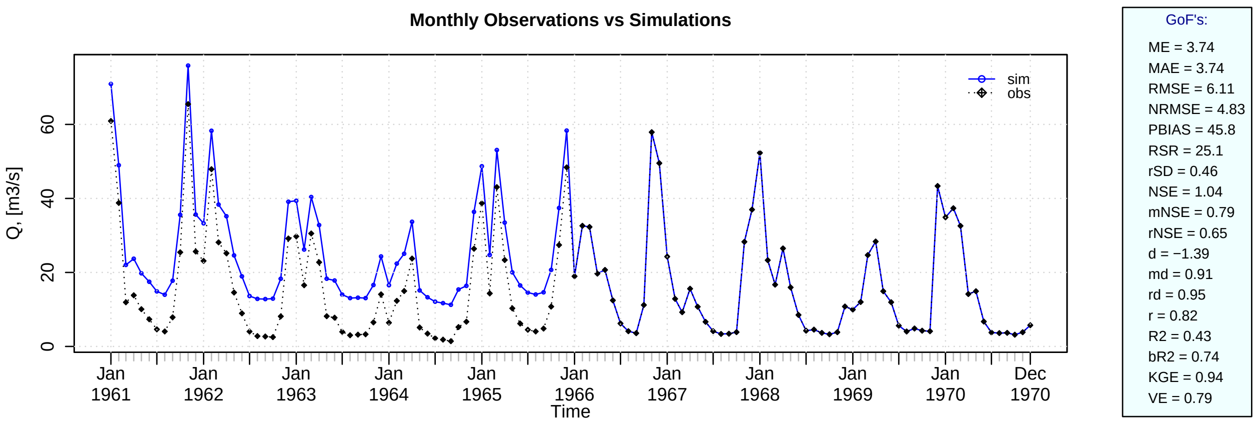


4.1 Removing warm-up period

Using the first two years (1961-1962) as warm-up period, and removing the corresponding observed and simulated values from the computation of the goodness-of-fit measures:

```r
ggof(sim=sim, obs=obs, ftype="dm", FUN=mean, cal.ini="1963-01-01")
```





Verification of the goodness-of-fit measures for the daily values after removing the warm-up period:

```r
sim <- window(sim, start="1963-01-01")
obs <- window(obs, start="1963-01-01")

gof(sim, obs)
```

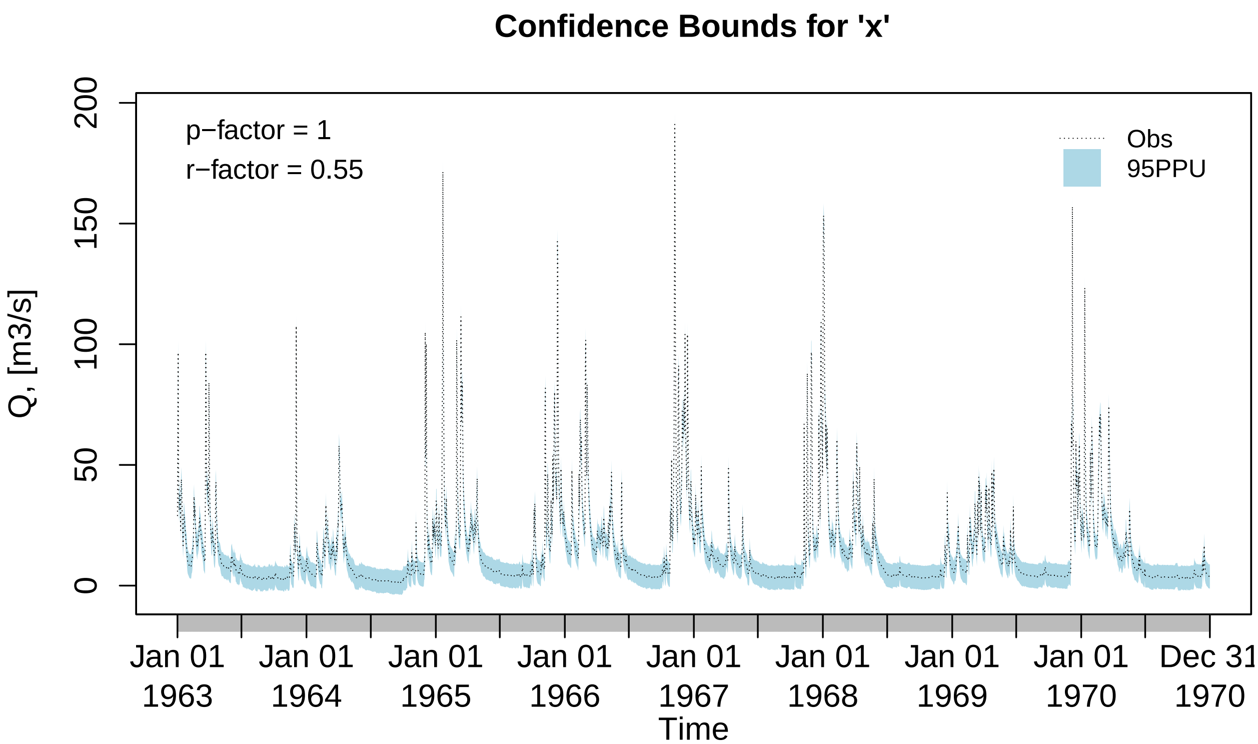
```
##           [,1]
## ME        3.74
## MAE       3.74
## MSE      37.74
## RMSE      6.14
## ubRMSE    4.87
## NRMSE % 33.70
## PBIAS % 25.20
## RSR       0.34
## rSD       1.02
## NSE       0.89
## mNSE      0.68
## rNSE     -0.47
## wNSE      0.98
## d         0.97
## dr        0.84
## md        0.84
## rd        0.64
## cp        0.52
## r         0.97
```

```
## R2       0.89
## bR2      0.81
## KGE      0.75
## KGElf    0.57
## KGEnp    0.69
## sKGE     0.70
## VE       0.75
```

4.2 Plotting uncertainty bands

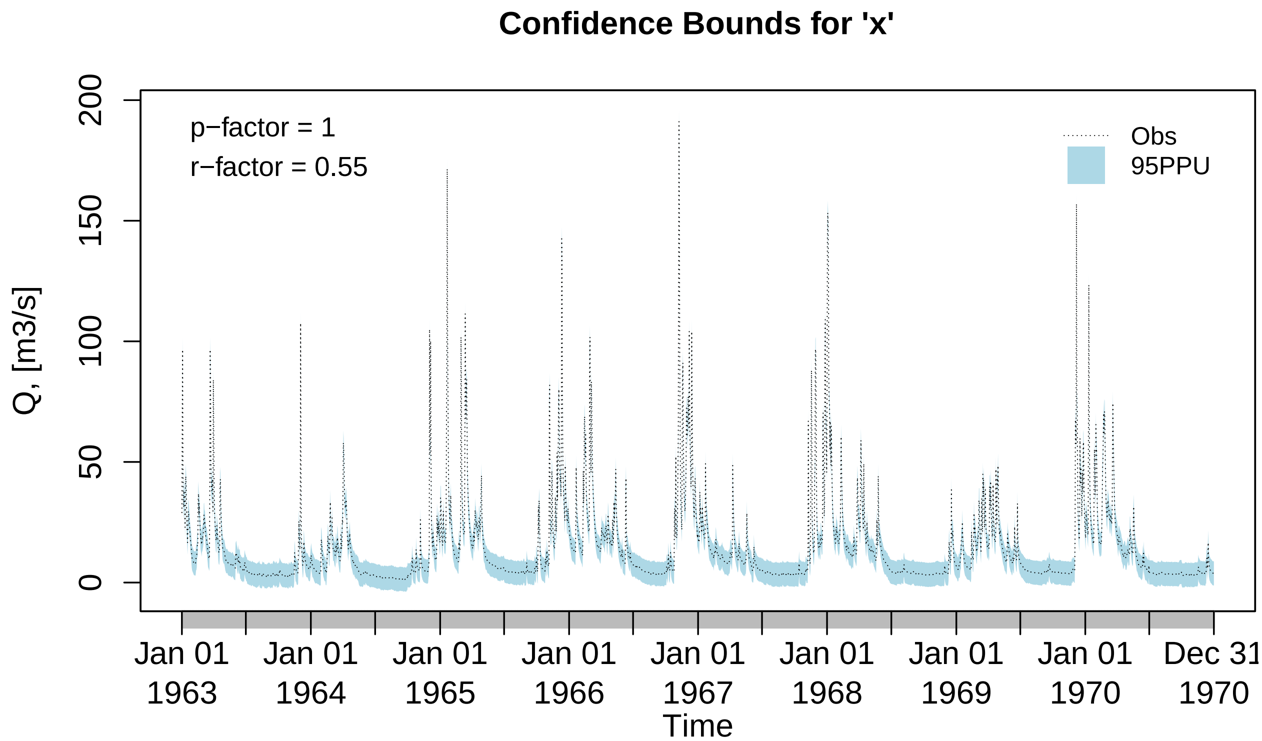
Generating fictitious lower and upper uncertainty bounds:

```
lband <- obs - 5
uband <- obs + 5
plotbands(obs, lband, uband)
```



Plotting the previously generated uncertainty bands:

```
plotbands(obs, lband, uband)
```

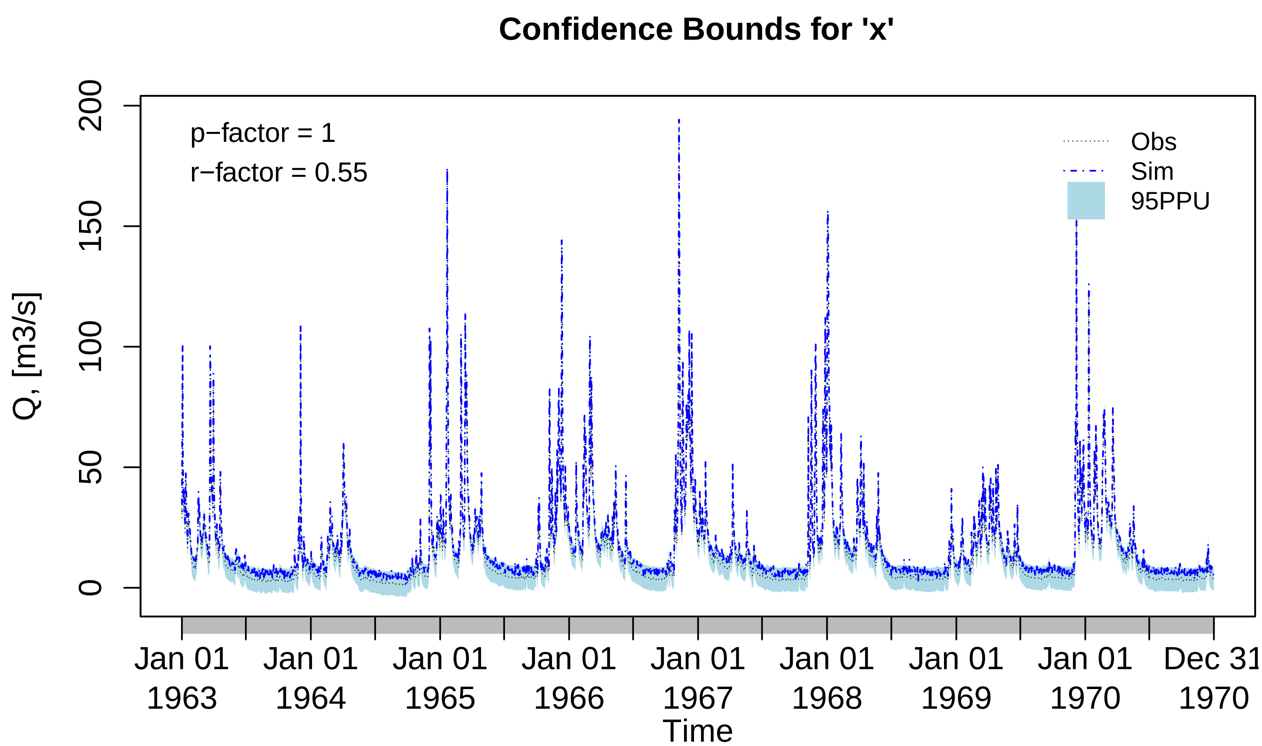


Randomly generating a simulated time series:

```
sim <- obs + rnorm(length(obs), mean=3)
```

Plotting the previously generated simualted time series along the obsertations and the uncertainty bounds:

```
plotbands(obs, lband, uband, sim)
```



4.3 Analysis of the residuals

Computing the daily residuals (even if this is a dummy example, it is enough for illustrating the capability)

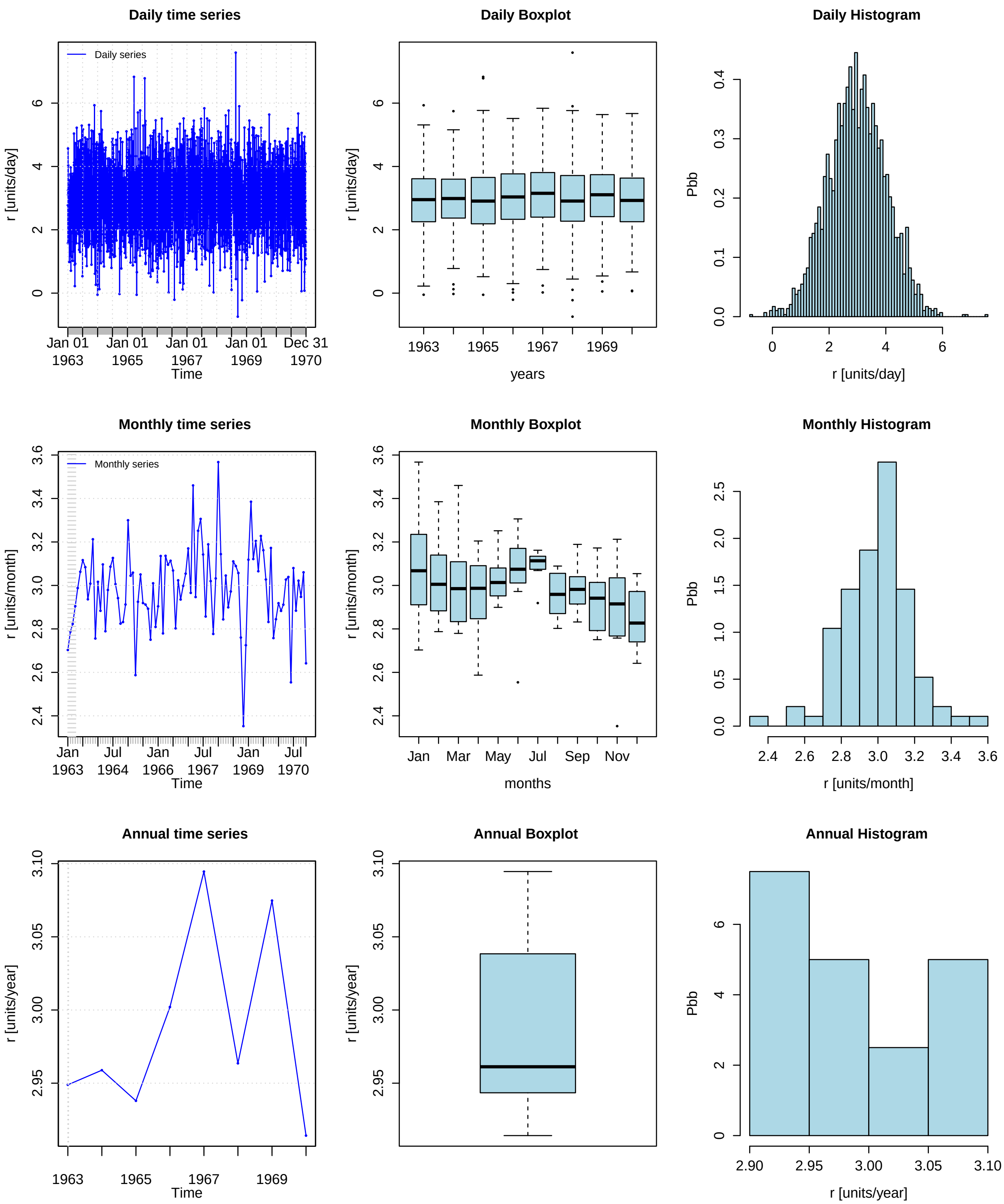
```
r <- sim-obs
```

Summarizing and plotting the residuals (it requires the hydroTSM package):

```
library(hydroTSM)
smry(r)
```

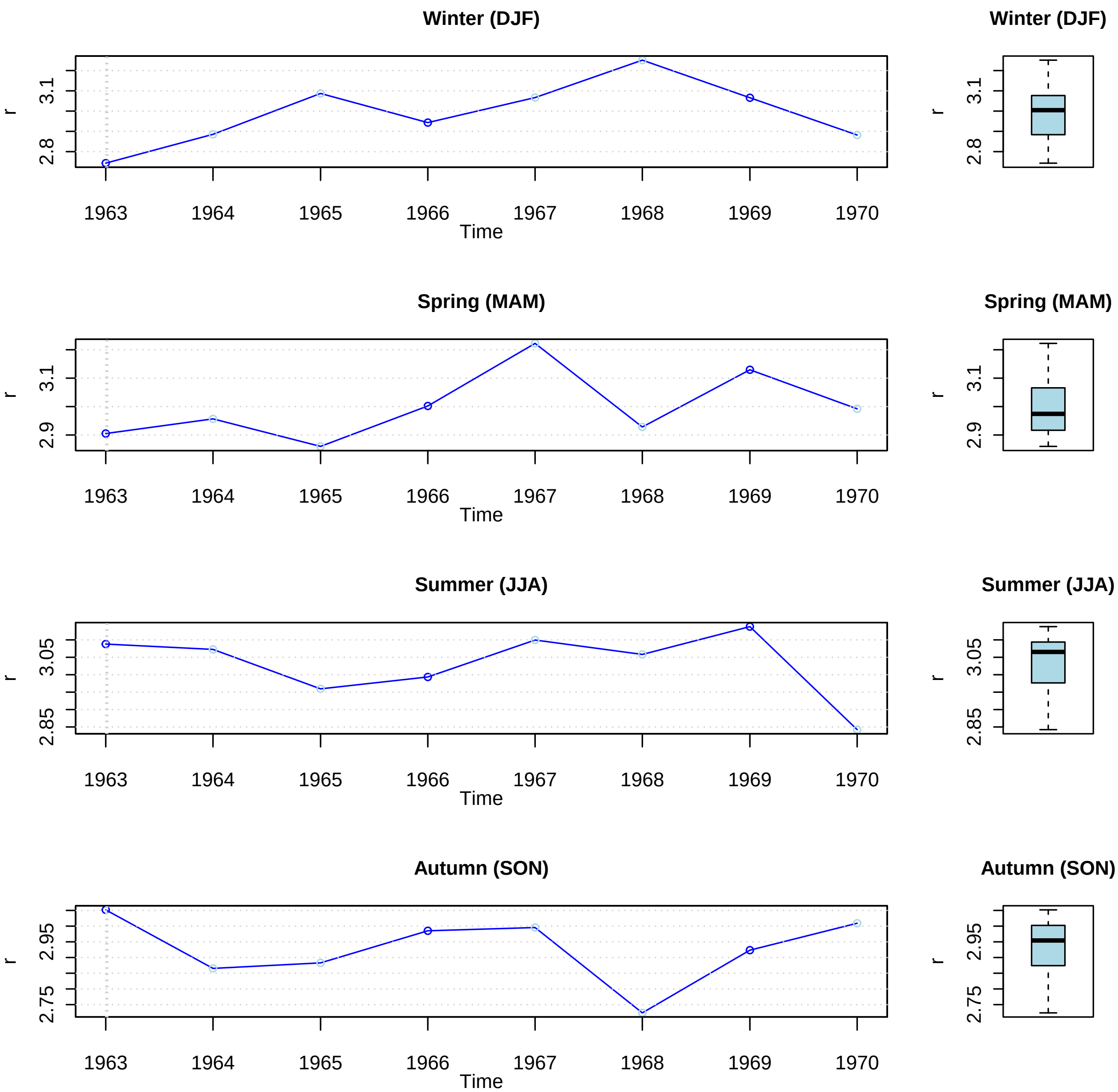
```
##                Index         r
## Min.     1963-01-01   -0.7433
## 1st Qu.  1964-12-31    2.3110
## Median   1966-12-31    2.9780
## Mean     1966-12-31    2.9870
## 3rd Qu.  1968-12-30    3.6790
## Max.     1970-12-31    7.5950
## IQR            <NA>    1.3686
## sd             <NA>    1.0212
## cv             <NA>    0.3419
## Skewness       <NA>    0.0035
## Kurtosis       <NA>    0.0755
## NA's           <NA>    2.0000
## n              <NA> 2922.0000
```

```
# daily, monthly and annual plots, boxplots and histograms
hydroplot(r, FUN=mean)
```



Seasonal plots and boxplots

```
# daily, monthly and annual plots, boxplots and histograms
hydroplot(r, FUN=mean, pfreq="seasonal")
```



5 Software details

This tutorial was built under:

```
## [1] "aarch64-apple-darwin20 (64-bit)"
```

```
## [1] "R version 4.3.2 (2023-10-31)"
```

```
## [1] "hydroGOF 0.4-15"
```

6 Version history

- v0.3: Jan-2024
- v0.2: Mar-2020
- v0.1: Aug 2011