Run a hydrological model (minimalistic example)

Michael Schirmer

This vignette shows how to run a hydrological model as a minimalistic example.

Preparations

Load packages

```
library(openQUARREL)
library(dplyr)
```

Create a minimum input

```
# input_data is an example input loaded with openQUARREL
minimum_input <- input_data %>%
  # filter to a single catchment, i.e. Thur at Jonschwil
  filter(HSU_ID == "2303") %>%
  # select only a few columns
  select(DatesR, P, T, E)

# convert Date to POSIXct
minimum_input$DatesR <- as.POSIXct(minimum_input$DatesR)
attr(minimum_input$DatesR, "tzone") <- "UTC"
```

The data frame `minimum_input` contains columns `P` for precipitation in mm/d, `T` for air temperature in degrees Celsius and `E` for potential evapotranspiration in mm/d. See also the great airGR get started documentation. If you need to calculate `E`, there are functions available, e.g. in airGR::PE_Oudin.

DatesR	P	T	E
1981-01-01	7.08	-0.42	0.60
1981-01-02	7.82	-2.55	0.32
1981-01-03	27.37	2.43	0.98
1981-01-04	22.56	-1.16	0.51
1981-01-05	8.68	-4.86	0.02
1981-01-06	14.99	-4.00	0.13

Choose a hydrological model

```
# default_cal_par are default calibration parameter loaded with openQUARREL
# todo: cal_par needs to be a global parameter currently, needs to be changed in future releases
cal_par <- default_cal_par
# todo: topmodel is not running right now as the required BasinInfo is not provided in example data
model <- "TUW"
```

Set model parameters

Create a parameter set, which is here the middle points of the provided intervals. See TUWmodel documentation for parameter explanation. The list `default_cal_par` is loaded with `library(openQUARREL)` and holds model and calibration specific settings.

```
param <- (cal_par[[model]]$upper + default_cal_par[[model]]$lower) / 2
print(param)
#>     SCF     DDF      Tr      Ts      Tm   LPrat      FC    BETA      k0      k1      k2    lsuz   cperc    bmax cr
#>     1.2     2.5     2.0    -1.0     0.0     0.5   300.0    10.0     1.0    16.0   140.0    50.5     4.0    15.0
```

Create input

This is done specifically for each model/package requirement, here TUWmodel.

```
# todo: the last input can be empty, will be not required in a future release
input <- create_input(model, minimum_input, list())
```

Run the model

```
sim <- simulate_model(model, param, input)
```

The output is a list with fields date, Qsim, Qobs, SWE, psolid, pliquid, melt, more_info. You can access typical components as simulated discharge `Qsim`, and if provided in the input also the observed discharge `Qobs`. If a snow module is integrated in the model or chosen to be added (see `vignette("include_snow_module")`) snow water equivalent `SWE`, the partitioned precipitation fluxes `pliquid` and `psolid`, and the snow `melt` flux. The element `more_info` is specific to the chosen model and package, in this case TUWmodel.

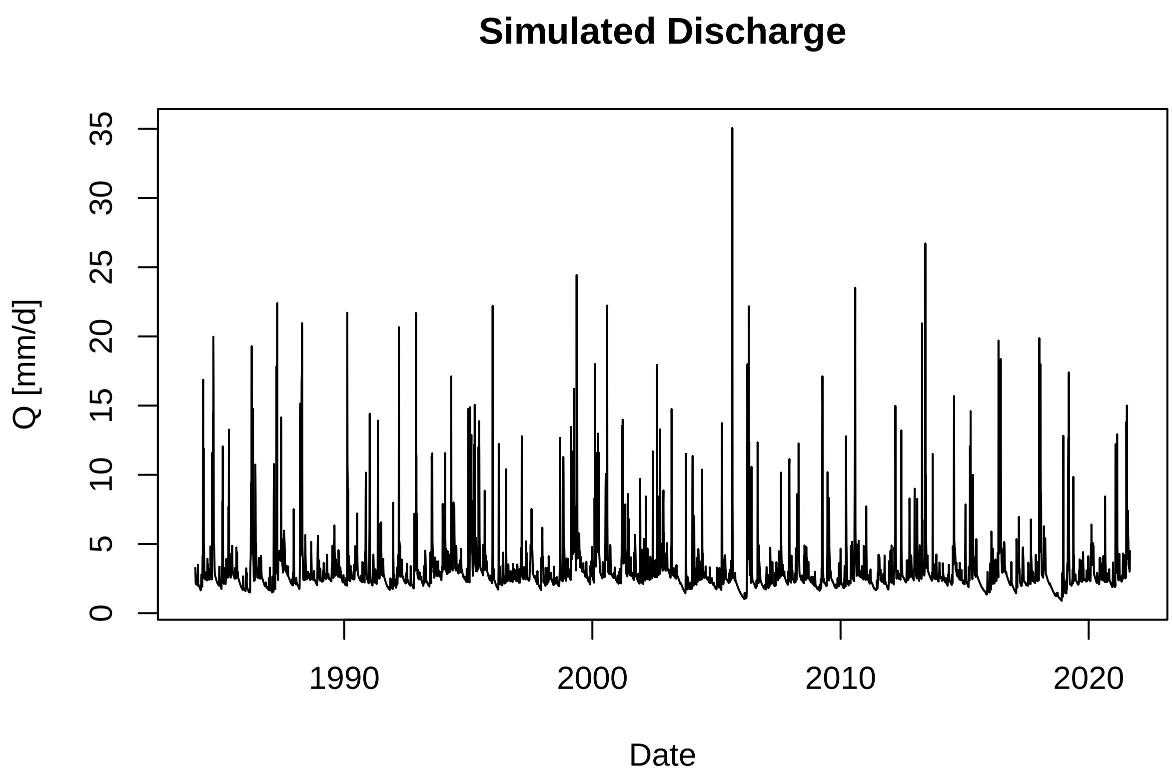
```
str(sim)
#> List of 8
#>  $ date     : Date[1:14853], format: "1981-01-01" "1981-01-02" "1981-01-03" "1981-01-04" ...
#>  $ Qsim     : num [1:14853] 0.000362 0.001447 0.003247 0.005758 0.008975 ...
#>  $ Qobs     : NULL
#>  $ SWE      : num [1:14853] 6.85 16.24 10.16 37.23 47.65 ...
#>  $ psolid   : num [1:14853] 5.71 7.82 0 22.56 8.68 ...
#>  $ pliquid  : num [1:14853] 1.37 0 27.37 0 0 ...
#>  $ melt     : num [1:14853] 0 0 6.08 0 0 ...
#>  $ more_info:List of 1
#>   ..$ output_model:List of 22
#>   .. ..$ itsteps: int 14853
#>   .. ..$ 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 [1:14853] 7.08 7.82 27.37 22.56 8.68 ...
#>   .. ..$ airt   : num [1:14853] -0.42 -2.55 2.43 -1.16 -4.86 ...
#>   .. ..$ ep     : num [1:14853] 0.6 0.32 0.98 0.51 0.02 0.13 0 0 0 0.28 ...
#>   .. ..$ output : num [1, 1:20, 1:14853] 3.62e-04 6.85 5.14e+01 1.37 5.71 ...
#>   .. ..$ qzones : num [1, 1:14853] 0.000362 0.001447 0.003247 0.005758 0.008975 ...
#>   .. ..$ q      : num [1, 1:14853] 0.000362 0.001447 0.003247 0.005758 0.008975 ...
#>   .. ..$ swe    : num [1, 1:14853] 6.85 16.24 10.16 37.23 47.65 ...
```

```
#>    .. ..$ melt   : num [1, 1:14853] 0 0 6.08 0 0 ...
#>    .. ..$ q0     : num [1, 1:14853] 0 0 0 0 0 0 0 0 0 0 ...
#>    .. ..$ q1     : num [1, 1:14853] 0 0 0 0 0 0 0 0 0 0 ...
#>    .. ..$ q2     : num [1, 1:14853] 0.0355 0.0352 0.035 0.0347 0.0345 ...
#>    .. ..$ moist  : num [1, 1:14853] 51.4 51.4 84.3 84.3 84.3 ...
#>    .. ..$ rain   : num [1, 1:14853] 1.37 0 27.37 0 0 ...
#>    .. ..$ snow   : num [1, 1:14853] 5.71 7.82 0 22.56 8.68 ...
#>    .. ..$ eta    : num [1, 1:14853] 0 0 0.554 0 0 ...
#>    .. ..$ suz    : num [1, 1:14853] 0 0 0 0 0 0 0 0 0 0 ...
#>    .. ..$ slz    : num [1, 1:14853] 4.96 4.93 4.89 4.86 4.83 ...
```

Plot results

Here is a simple plot for simulated discharge with skipping the first ~3 years (warm-up):

```
plot(sim$date[3*365:length(sim$date)], sim$Qsim[3*365:length(sim$date)], type = "l", main = "Simulated I
```



Next steps

Now you do this again with a different hydrological model (see table in `vignette("model_overview")`), or proceed with `vignette("include_snow_module")`.

Model overview

Michael Schirmer

Hydrological models

Hydrological models included and tested in the openQUARREL package.

Note: There are more models in the original packages, e.g. daily and monthly models in airGR or redundant airGR models in the hydromad package.

Model	has own snow module	Package
GR4J		airGR
GR5J		airGR
GR6J		airGR
CemaNeigeGR4J	X	airGR
CemaNeigeGR5J	X	airGR
CemaNeigeGR6J	X	airGR
cmd		hydromad
snow	X	hydromad
cwi		hydromad
awbm		hydromad
sacramento		hydromad
bucket		hydromad
TUW	X	TUWmodel
topmodel		topmodel

Snow modules

Snow modules included and tested in the openQUARREL package. TUWsnow is the snow module of TUWmodel. The snow module of snow is not separately implemented yet.

Snow module	Package
CemaNeige	airGR
TUWsnow	TUWmodel

Couple a snow module to a hydrological model

This vignette shows how to couple a hydrological model with a hydrological model without a snow model. While `snow`, `TUW` and the `airGR` models come with an own snow module (the latter just with renaming the model from `GR6J` to `CemaNeigeGR6J`), this needs a bit more coding within this package (and can be done more integrated similar to the airGR package in the future). Important new parts are in *italic*.

Preparations

Load packages

```
library(openQUARREL)
library(dplyr)
```

Create a minimum input

```
minimum_input <- input_data %>%
  filter(HSU_ID == "2303") %>%
  select(DatesR, P, T, E)

# convert Date to POSIXct
minimum_input$DatesR <- as.POSIXct(minimum_input$DatesR)
attr(minimum_input$DatesR, "tzone") <- "UTC"
```

New parts are in italic:

Choose a snow module and a hydrological model (without an own snow module)

```
cal_par <- default_cal_par # todo: this needs to be changed
snow_module <- "CemaNeige"
# todo: topmodel is not running right now as the required BasinInfo is not provided in example data
model <- "sacramento"
```

Set model parameters

```
param <- (default_cal_par[[model]]$upper + default_cal_par[[model]]$lower) / 2
print(param)
#>       uztwm       uzfwm         uzk       pctim       adimp       zperc        rexp       lztwm
#>  75.5000000  75.5000000   0.3000000   0.0500005   0.2000000 125.5000000   2.5000000 250.5000000 500.
```

Set snow module parameters

```
snow_param <- (default_cal_par[[snow_module]]$upper + default_cal_par[[snow_module]]$lower) / 2
print(snow_param)
#> [1]  0.50000 54.51825
```

Create input

```
# todo: check on basin_info
input <- create_input(model, minimum_input, list()) %>%
  suppressWarnings() %>% suppressMessages()
```

Create snow input

```r
snow_input <- create_input(snow_module, minimum_input, list()) %>%
  suppressWarnings() %>% suppressMessages()
```

Run the model without the snow module

```r
sim <- simulate_model(model, param, input)
```

Run the model with the snow module

Simulate snow module

```r
# simulate snow module
snow_module_results <- simulate_snow(snow_module, snow_param, snow_input) %>%
  suppressWarnings() %>% suppressMessages()
```

Update precipitation with snow module surface water runoff

```r
input$P <- snow_module_results$surface_water_runoff
```

Run the model with updated input*

```r
sim_update <- simulate_model(model, param, input)
```

Combine snow and runoff results (not required, just nice)

```r
sim_update <- merge_snow_runoff_sim(sim_update, snow_module_results)
```

The output contains now also the snow module results `SWE`, `psolid`, `pliquid` and `melt`, and `more_info` contains a list of 2, i.e both the complete hydrological model and the snow module results with all details directly from the original package output.

```r
str(sim_update)
#> List of 9
#>  $ date               : Date[1:14853], format: "1981-01-01" "1981-01-02" "1981-01-03" "1981-01-04"
#>  $ Qsim               : num [1:14853] 48.9 42.7 38.6 32.5 28.4 ...
#>  $ Qobs               : NULL
#>  $ SWE                : num [1:14853] 6.05 13.87 15.07 37.63 46.31 ...
#>  $ psolid             : num [1:14853] 6.05 7.82 3.9 22.56 8.68 ...
#>  $ pliquid            : num [1:14853] 1.03 0 23.47 0 0 ...
#>  $ melt               : num [1:14853] 0 0 2.7 0 0 ...
#>  $ more_info          :List of 2
#>   ..$ output_model:List of 2
#>   .. ..$ init_model  :List of 7
#>   .. .. ..$ call        : language hydromad::hydromad(DATA = NULL, sma = sma, routing = routing, uzt
#>   .. .. ..$ parlist     :List of 13
#>   .. .. .. ..$ uztwm: num [1:2] 1 150
#>   .. .. .. ..$ uzfwm: num [1:2] 1 150
#>   .. .. .. ..$ uzk  : num [1:2] 0.1 0.5
#>   .. .. .. ..$ pctim: num [1:2] 1e-06 1e-01
#>   .. .. .. ..$ adimp: num [1:2] 0 0.4
#>   .. .. .. ..$ zperc: num [1:2] 1 250
#>   .. .. .. ..$ rexp : num [1:2] 0 5
#>   .. .. .. ..$ lztwm: num [1:2] 1 500
#>   .. .. .. ..$ lzfsm: num [1:2] 1 1000
#>   .. .. .. ..$ lzfpm: num [1:2] 1 1000
#>   .. .. .. ..$ lzsk : num [1:2] 0.01 0.25
#>   .. .. .. ..$ lzpk : num [1:2] 0.0001 0.25
#>   .. .. .. ..$ pfree: num [1:2] 0 0.6
```

```
#>   .. .. ..$ last.updated: POSIXct[1:1], format: "2025-08-25 11:29:31"
#>   .. .. ..$ sma         : chr "sacramento"
#>   .. .. ..$ sma.fun     : chr "sacramento.sim"
#>   .. .. ..$ sma.formals :Dotted pair list of 24
#>   .. .. .. ..$ DATA        : symbol
#>   .. .. .. ..$ uztwm       : symbol
#>   .. .. .. ..$ uzfwm       : symbol
#>   .. .. .. ..$ uzk         : symbol
#>   .. .. .. ..$ pctim       : symbol
#>   .. .. .. ..$ adimp       : symbol
#>   .. .. .. ..$ zperc       : symbol
#>   .. .. .. ..$ rexp        : symbol
#>   .. .. .. ..$ lztwm       : symbol
#>   .. .. .. ..$ lzfsm       : symbol
#>   .. .. .. ..$ lzfpm       : symbol
#>   .. .. .. ..$ lzsk        : symbol
#>   .. .. .. ..$ lzpk        : symbol
#>   .. .. .. ..$ pfree       : symbol
#>   .. .. .. ..$ etmult      : num 1
#>   .. .. .. ..$ dt          : num 1
#>   .. .. .. ..$ uztwc_0     : num 0.5
#>   .. .. .. ..$ uzfwc_0     : num 0.5
#>   .. .. .. ..$ lztwc_0     : num 0.5
#>   .. .. .. ..$ lzfsc_0     : num 0.5
#>   .. .. .. ..$ lzfpc_0     : num 0.5
#>   .. .. .. ..$ adimc_0     : num 0.5
#>   .. .. .. ..$ min_ninc    : num 20
#>   .. .. .. ..$ return_state: logi FALSE
#>   .. .. ..$ warmup      : num 100
#>   .. .. ..- attr(*, "class")= chr [1:2] "hydromad.sacramento" "hydromad"
#>   .. ..$ fitted_model:List of 9
#>   .. .. ..$ call        : language hydromad::hydromad(DATA = (zoo::read.zoo(input[ind, ]))(), sma = 
#>   .. .. ..$ parlist     :List of 13
#>   .. .. .. ..$ uztwm: num 75.5
#>   .. .. .. ..$ uzfwm: num 75.5
#>   .. .. .. ..$ uzk  : num 0.3
#>   .. .. .. ..$ pctim: num 0.05
#>   .. .. .. ..$ adimp: num 0.2
#>   .. .. .. ..$ zperc: num 126
#>   .. .. .. ..$ rexp : num 2.5
#>   .. .. .. ..$ lztwm: num 250
#>   .. .. .. ..$ lzfsm: num 500
#>   .. .. .. ..$ lzfpm: num 500
#>   .. .. .. ..$ lzsk : num 0.13
#>   .. .. .. ..$ lzpk : num 0.125
#>   .. .. .. ..$ pfree: num 0.3
#>   .. .. ..$ last.updated: POSIXct[1:1], format: "2025-08-25 11:29:31"
#>   .. .. ..$ sma         : chr "sacramento"
#>   .. .. ..$ sma.fun     : chr "sacramento.sim"
#>   .. .. ..$ sma.formals :Dotted pair list of 24
#>   .. .. .. ..$ DATA        : symbol
#>   .. .. .. ..$ uztwm       : symbol
#>   .. .. .. ..$ uzfwm       : symbol
```

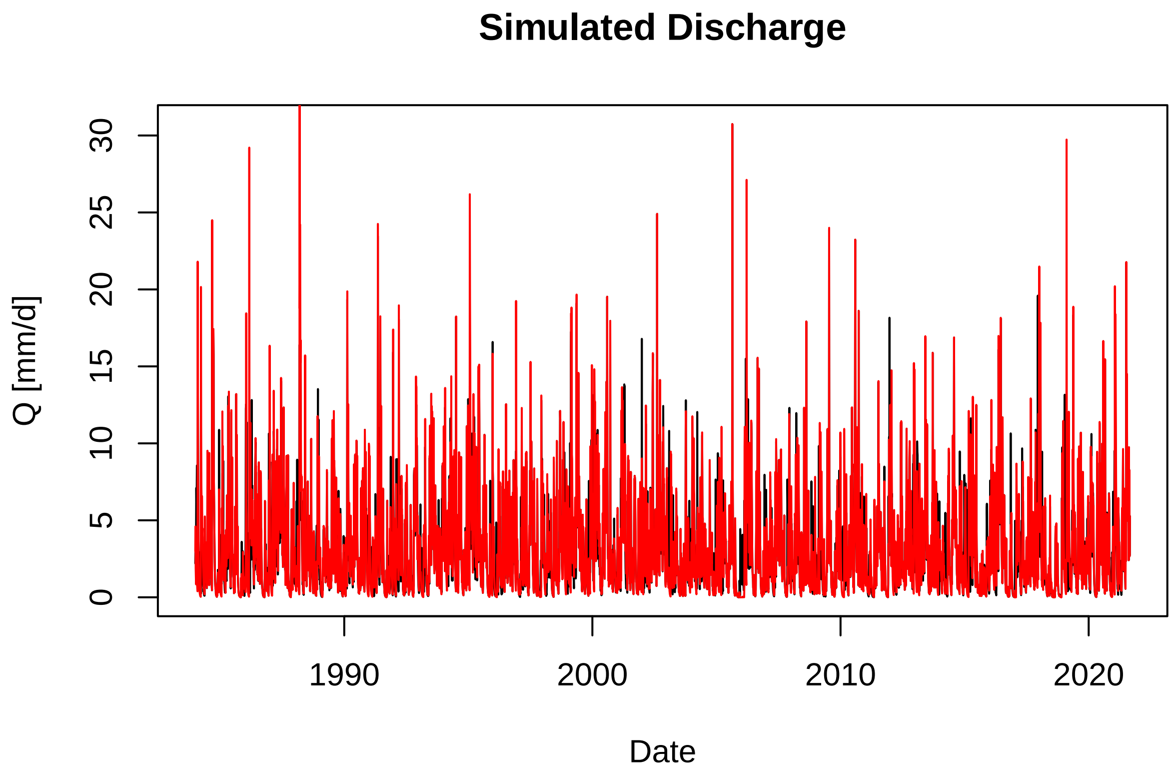
```
#>   .. .. .. ..$ uzk         : symbol
#>   .. .. .. ..$ pctim       : symbol
#>   .. .. .. ..$ adimp       : symbol
#>   .. .. .. ..$ zperc       : symbol
#>   .. .. .. ..$ rexp        : symbol
#>   .. .. .. ..$ lztwm       : symbol
#>   .. .. .. ..$ lzfsm       : symbol
#>   .. .. .. ..$ lzfpm       : symbol
#>   .. .. .. ..$ lzsk        : symbol
#>   .. .. .. ..$ lzpk        : symbol
#>   .. .. .. ..$ pfree       : symbol
#>   .. .. .. ..$ etmult      : num 1
#>   .. .. .. ..$ dt          : num 1
#>   .. .. .. ..$ uztwc_0     : num 0.5
#>   .. .. .. ..$ uzfwc_0     : num 0.5
#>   .. .. .. ..$ lztwc_0     : num 0.5
#>   .. .. .. ..$ lzfsc_0     : num 0.5
#>   .. .. .. ..$ lzfpc_0     : num 0.5
#>   .. .. .. ..$ adimc_0     : num 0.5
#>   .. .. .. ..$ min_ninc    : num 20
#>   .. .. .. ..$ return_state: logi FALSE
#>   .. .. ..$ warmup      : num 0
#>   .. .. ..$ data        :'zooreg' series from 1981-01-01 to 2021-08-31
#>   Data: num [1:14853, 1:3] 1.03 0 26.17 0 0 ...
#>   .. .. .. ..- attr(*, "dimnames")=List of 2
#>   .. .. .. .. ..$ : NULL
#>   .. .. .. .. ..$ : chr [1:3] "P" "E" "T"
#>   Index:  POSIXct[1:14853], format: "1981-01-01" "1981-01-02" "1981-01-03" "1981-01-04" ...
#>   Frequency: 1.15740740740741e-05
#>   .. .. ..$ U           :'zooreg' series from 1981-01-01 to 2021-08-31
#>   Data: num [1:14853] 48.9 42.7 38.6 32.5 28.4 ...
#>   Index:  POSIXct[1:14853], format: "1981-01-01" "1981-01-02" "1981-01-03" "1981-01-04" ...
#>   Frequency: 1.15740740740741e-05
#>   .. .. ..- attr(*, "class")= chr [1:2] "hydromad.sacramento" "hydromad"
#>   ..$ snow_module :List of 3
#>   .. ..$ DatesR         : POSIXlt[1:14853], format: "1981-01-01" "1981-01-02" "1981-01-03" "1981-01-
#>   .. ..$ CemaNeigeLayers:List of 1
#>   .. .. ..$ Layer01:List of 11
#>   .. .. .. ..$ Pliq        : num [1:14853] 1.03 0 23.47 0 0 ...
#>   .. .. .. ..$ Psol        : num [1:14853] 6.05 7.82 3.9 22.56 8.68 ...
#>   .. .. .. ..$ SnowPack    : num [1:14853] 6.05 13.87 15.07 37.63 46.31 ...
#>   .. .. .. ..$ ThermalState: num [1:14853] -0.21 -1.38 0 -0.58 -2.72 ...
#>   .. .. .. ..$ Gratio      : num [1:14853] 0.0196 0.045 0.0489 0.1221 0.1502 ...
#>   .. .. .. ..$ PotMelt     : num [1:14853] 0 0 17.8 0 0 ...
#>   .. .. .. ..$ Melt        : num [1:14853] 0 0 2.7 0 0 ...
#>   .. .. .. ..$ PliqAndMelt : num [1:14853] 1.03 0 26.17 0 0 ...
#>   .. .. .. ..$ Temp        : num [1:14853] -0.42 -2.55 2.43 -1.16 -4.86 ...
#>   .. .. .. ..$ Gthreshold  : num [1:14853] 308 308 308 308 308 ...
#>   .. .. .. ..$ Glocalmax   : num [1:14853] -1000 -1000 -1000 -1000 -1000 ...
#>   .. ..$ StateEnd       :List of 3
#>   .. .. ..$ Store          :List of 4
#>   .. .. .. ..$ Prod: num NA
#>   .. .. .. ..$ Rout: num NA
```

```
#>   .. .. .. ..$ Exp : num NA
#>   .. .. .. ..$ Int : num NA
#>   .. .. ..$ UH             :List of 2
#>   .. .. .. ..$ UH1: num [1:20] NA NA NA NA NA NA NA NA NA NA ...
#>   .. .. .. ..$ UH2: num [1:40] NA NA NA NA NA NA NA NA NA NA ...
#>   .. .. ..$ CemaNeigeLayers:List of 4
#>   .. .. .. ..$ G      : num 1.13e-05
#>   .. .. .. ..$ eTG    : num 0
#>   .. .. .. ..$ Gthr   : num NA
#>   .. .. .. ..$ Glocmax: num NA
#>   .. .. ..- attr(*, "class")= chr [1:3] "IniStates" "CemaNeige" "daily"
#>   .. ..- attr(*, "class")= chr [1:3] "OutputsModel" "daily" "CemaNeige"
#>  $ surface_water_runoff: num [1:14853] 1.03 0 26.17 0 0 ...
```

Plot results

Here is a simple plot for simulated discharge with (red) and without (black) the snow module, skipping the first ~3 years (warm-up):

```
plot(sim$date[3*365:length(sim$date)], sim$Qsim[3*365:length(sim$date)], type = "l", main = "Simulated D
lines(sim_update$date[3*365:length(sim$date)], sim_update$Qsim[3*365:length(sim$date)], col="red")
```



Next steps

Now you do this again with a different hydrological model or snow module (see table in `vignette("model_overview")`), or proceed with `vignette("calibrate_validate")`.

Calibration methods overview

Michael Schirmer

The first column represents the `cal_fn` input string used in many functions.

Calibration option	Description	Reference	Package
Calibration_Michel	Combination of a grid search with prior knowledge of parameter combination and a steepest descent local search	Calibration_Michel	airGR
steepest_descent	Variant of Calibration_Michel with a monte carlo sampling with 100 trials instead of a grid search		openQUARREL
montecarlo___random_1000	Monte carlo sampling with X trials (X is a variable, here 1000)	runif	stats
montecarlo___lhs_1000	Monte carlo sampling using a latin hyper cube	randomLHS	lhs
montecarlo___sobol_1000	Monte carlo sampling using sobol sampling	sobol	randtoolbox
nlminb	Optimization using PORT routines	nlminb	stats
optim	General-purpose Optimization	optim	stats
nlminb___random_100	Optimization using PORT routines starting with Monte carlo resampling	nlminb	stats
optim___random_100	General-purpose Optimization starting with Monte carlo resampling	optim	stats
DEoptim	Evolutionary global optimization via the Differential Evolution algorithm	DEoptim	DEoptim
hydroPSO	Enhanced Particle Swarm Optimisation	hydroPSO	hydroPSO
malschains	Optimization with the MA-LS-Chains algorithm	malschains	Rmalschains

Calibration and validation

Michael Schirmer

2025-08-25

This vignette shows how to calibrate and validate a hydrological model

Preparations

Do `vignette("run_model_minimalistic")` for data loading.

We now need additionally the hydroGOF package.

Note: the warning of deprecated packages can be ignored, openQUARREL is using only non-affected parts of hydroGOF.

```
library(hydroGOF)
#> Lade nötiges Paket: zoo
#>
#> Attache Paket: 'zoo'
#> Die folgenden Objekte sind maskiert von 'package:base':
#>
#>     as.Date, as.Date.numeric
#> The legacy packages maptools, rgdal, and rgeos, underpinning the sp package,
#> which was just loaded, will retire in October 2023.
#> Please refer to R-spatial evolution reports for details, especially
#> https://r-spatial.org/r/2023/05/15/evolution4.html.
#> It may be desirable to make the sf package available;
#> package maintainers should consider adding sf to Suggests:.
#> The sp package is now running under evolution status 2
#>      (status 2 uses the sf package in place of rgdal)
#> Please note that 'maptools' will be retired during October 2023,
#> plan transition at your earliest convenience (see
#> https://r-spatial.org/r/2023/05/15/evolution4.html and earlier blogs
#> for guidance);some functionality will be moved to 'sp'.
#>  Checking rgeos availability: FALSE
```

… and observed streamflow (column Qmm) from the example data set.

```
minimum_input <- input_data %>%
  filter(HSU_ID == "2303") %>%
  select(DatesR, Qmm) %>%
  inner_join(minimum_input, join_by(DatesR))
```

Define periods for warm-up, calibration and validation:

```
# split data set
split_indices <- split_data_set(
  minimum_input,
  c("1981-01-01", "1982-12-31", # warm up
    "1983-01-01", "2000-12-31", # calibration
```

```
    "2001-01-01", "2020-12-31") # validation
)
```

Calibration

Choose the calibration function, the error criterion and streamflow transformation applied (here `KGE` and `none` separated by a `__`), and whether you want to transform parameters to a hypercube.

```
cal_fn <- "steepest_descent"
error_crit_transfo <- "KGE__none"
do_transfo_param <- TRUE
```

Put basin information in a list. (todo: some redundancy with input, needs to be solved in future releases)

```
hydro_data <- list()
hydro_data$BasinObs <- minimum_input
# basin_data is an example data loaded with openQUARREL
minimum_basin_info <- basin_data[["2303"]]
# delete HypsoData not needed here
minimum_basin_info$HypsoData <- NULL
hydro_data$BasinInfo <- minimum_basin_info
```

Calibrate the model.

```
# calibrate the model
calibration_results <- calibrate_model(
  hydro_data, split_indices, model, input,
  cal_fn = cal_fn, do_transfo_param = do_transfo_param
) %>% suppressWarnings() %>% suppressMessages()
#> Random sampling with method steepest_descent finished in 1.514 secs with best value 0.75...
```

Show calibration results:

Calibration finished in 10.395 sec with best criterion 0.8220945 with the model parameters:

```
# get the parameter names and print
names(calibration_results$model_param) <- names(default_cal_par[[model]]$lower)
print(calibration_results$model_param)
#>         SCF         DDF          Tr          Ts          Tm       LPrat          FC        BETA
#>   1.4940000   0.7845411   1.2987184  -3.0000000  -1.1497423   1.0000000 113.1220231   8.5734205   2.
```

There is more information after calibration, `calibration_results` is a list of storing also the used calibration settings `cal_par` and with `more_info` the output of the chosen calibration function.

```
str(calibration_results)
#> List of 12
#>  $ model                 : chr "TUW"
#>  $ snow_module           : NULL
#>  $ model_param           : Named num [1:15] 1.494 0.785 1.299 -3 -1.15 ...
#>   ..- attr(*, "names")= chr [1:15] "SCF" "DDF" "Tr" "Ts" ...
#>  $ preset_snow_parameters: logi FALSE
#>  $ cal_fn                : chr "steepest_descent"
#>  $ error_crit_transfo    : chr "KGE__none"
#>  $ error_crit_val        : num 0.822
#>  $ cal_maximize          : logi TRUE
#>  $ do_transfo_param      : logi TRUE
#>  $ duration              : 'difftime' num 10.395
```

```
#>   ..- attr(*, "units")= chr "secs"
#>  $ cal_par               :List of 7
#>   ..$ lower           : Named num [1:15] 0.9 0 1 -3 -2 0 0 0 0 2 ...
#>   .. ..- attr(*, "names")= chr [1:15] "SCF" "DDF" "Tr" "Ts" ...
#>   ..$ upper           : Named num [1:15] 1.5 5 3 1 2 1 600 20 2 30 ...
#>   .. ..- attr(*, "names")= chr [1:15] "SCF" "DDF" "Tr" "Ts" ...
#>   ..$ nof_param       : int 15
#>   ..$ has_snow_module: logi TRUE
#>   ..$ DEoptim         :List of 2
#>   .. ..$ NP     : num 300
#>   .. ..$ itermax: num 200
#>   ..$ malschains      :List of 1
#>   .. ..$ maxEvals: num 2000
#>   ..$ hydroPSO        :List of 1
#>   .. ..$ control:List of 5
#>   .. .. ..$ write2disk: logi FALSE
#>   .. .. ..$ verbose   : logi FALSE
#>   .. .. ..$ npart     : num 80
#>   .. .. ..$ maxit     : num 50
#>   .. .. ..$ reltol    : num 1e-10
#>  $ more_info             :List of 8
#>   ..$ ParamFinalR  : num [1:15] 0.99 0.157 0.149 0 0.213 ...
#>   ..$ CritFinal    : num -0.822
#>   ..$ NIter        : num 41
#>   ..$ NRuns        : num 936
#>   ..$ HistParamR   : num [1:41, 1:15] 0.743 0.743 0.743 0.743 0.743 ...
#>   .. ..- attr(*, "dimnames")=List of 2
#>   .. .. ..$ : NULL
#>   .. .. ..$ : chr [1:15] "Param1" "Param2" "Param3" "Param4" ...
#>   ..$ HistCrit     : num [1:41, 1] -0.749 -0.765 -0.79 -0.795 -0.797 ...
#>   ..$ CritName     : chr "KGE_none"
#>   ..$ CritBestValue: NULL
```

Simulate model, similar to `vignette("include_snow_module")`, but now applying the calibrated parameters.

```
# simulate snow, if an external snow module is needed (not here, but when you change it)
# todo: this update process can be put in one function
if (exists("snow_module")) {
  if (!is.null(snow_module)) {

    # create input
    snow_input <- create_input(snow_module, minimum_input, list()) %>%
      suppressWarnings() %>% suppressMessages()

    # simulate snow
    snow_module_results <- simulate_snow(snow_module, snow_param, snow_input) %>%
      suppressWarnings() %>% suppressMessages()

    # update precip
    input$P <- snow_module_results$surface_water_runoff

  }
}
```

```
# run model
# Note: we put Qobs as input to have it available for airGR plots
sim <- simulate_model(model, calibration_results$model_param, input, Qobs = hydro_data$BasinObs$Qmm)

# merge snow module results with hydro model results
if (exists("snow_module_results")) {
  sim <- merge_snow_runoff_sim(sim, snow_module_results)
}
```

Validation

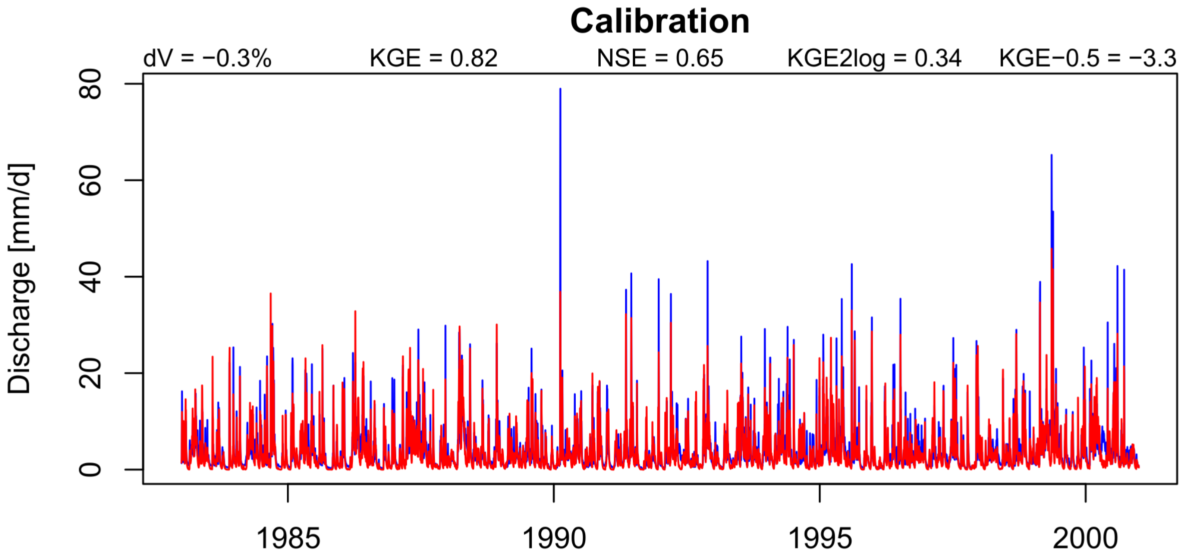
Plots

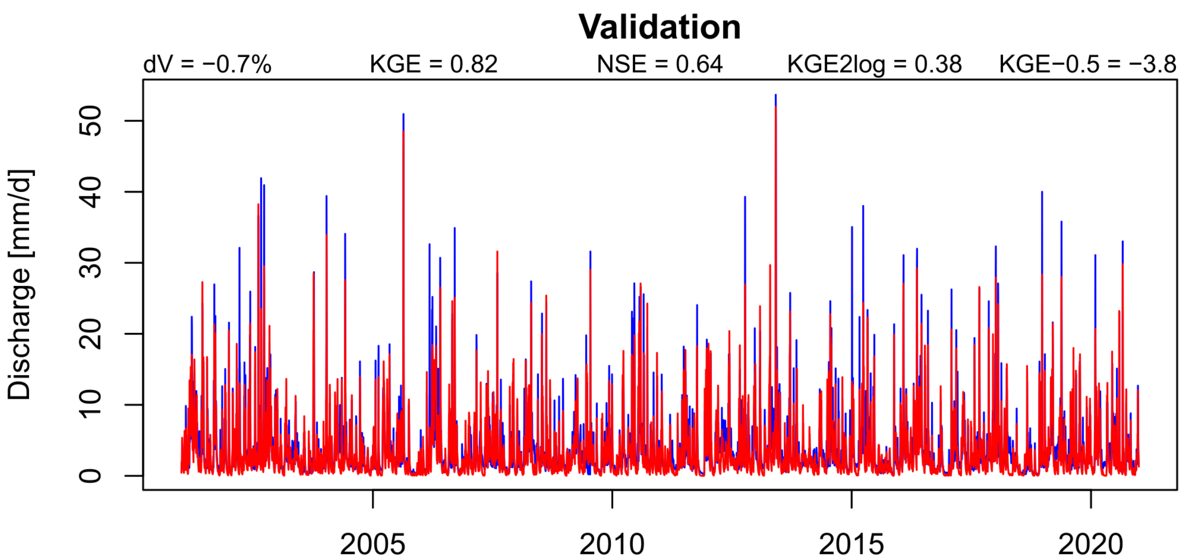
For validation there are two plots available which can be right now only only saved to disc as pdf (which will be changed in a future release).

The first is a simple plot showing simulated and observed streamflow with some indicators for both the calibration and validation period, simulated streamflow is red.

```
# define and create a output folder
output_folder <- "output"
dir.create(output_folder)

save_cal_val_plot(file.path(output_folder, "cal_val_plot.pdf"), hydro_data$BasinObs, sim$Qsim, split_inc
```



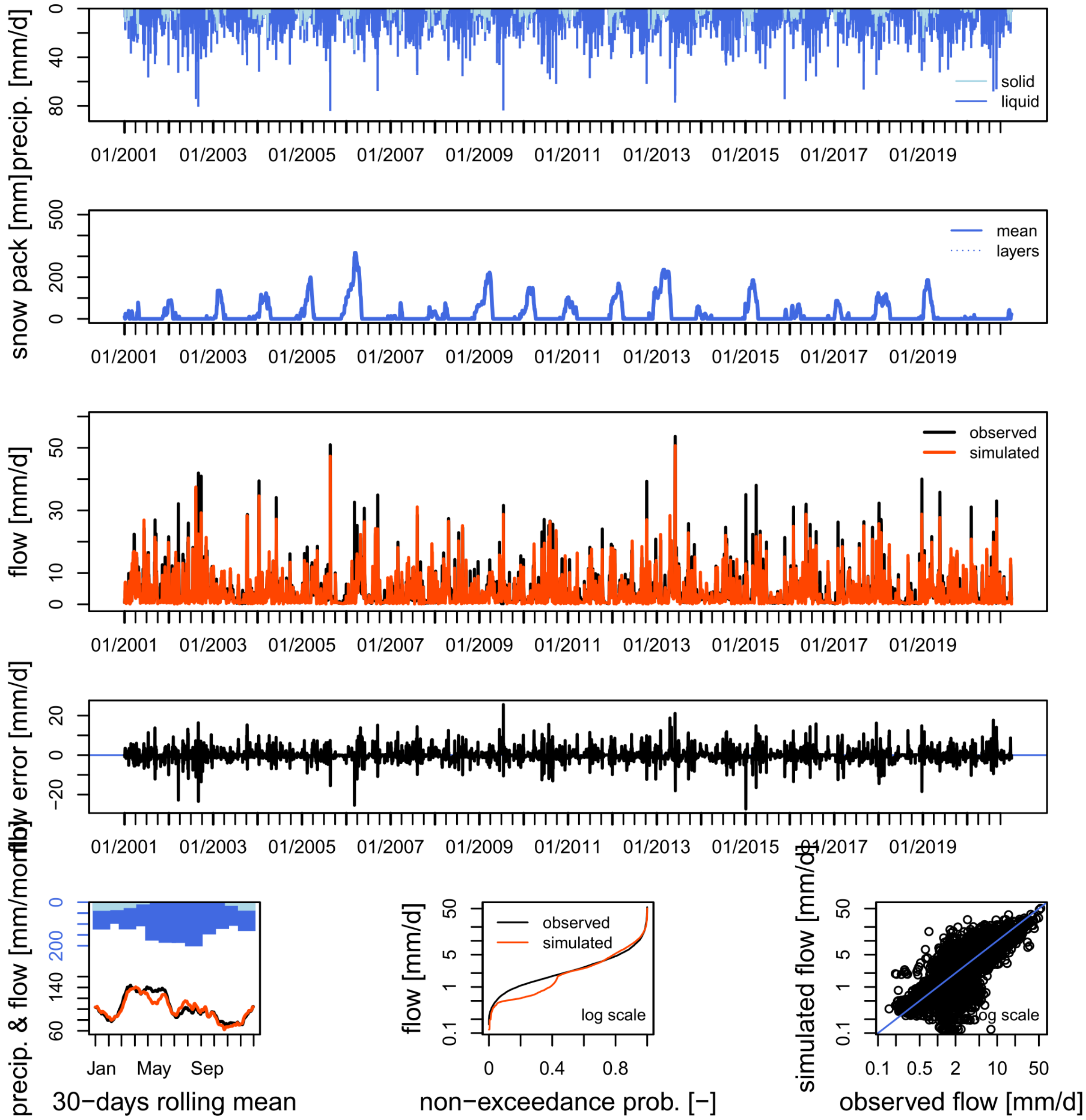


The second plot is the great airGR::plot adapted to models from other packages. We plot it for the calibration and validation period separately.

```
# airGR plots for validation and calibration time period

# calibration period
save_airGR_plot(file.path(output_folder, "airGR_cal.pdf"), model, sim, split_indices$ind_cal, hydro_data
  suppressWarnings() %>% suppressMessages()
# validation period
save_airGR_plot(file.path(output_folder, "airGR_val.pdf"), model, sim, split_indices$ind_val, hydro_data
  suppressWarnings() %>% suppressMessages()
```

Here is the validation plot shown:



It seems that this model has difficulties in spring with snow melt, quite probably because of a missing snow covered fraction parameterisation. Try this with the model `CemaNeigeGR4J` as this snow module has such a parameterisation implemented.

Calculation of (sub)seasonal performance metrics

Define metrics with streamflow transformations, separated by `__`. For example `mae__power__-0.5` is the mean absolute error calculated with hydroGOF::mae, and using a power transformation with exponent -0.5.

Note: All these (and other) string combinations can also be used for calibration with applying it to `error_crit_transfo`.

```
# validation settings
val_crit_transfo <- c("KGE__none", "NSE__none", "VE__none", "pbias__none", "mae__none", "mse__none",
                      "KGE__power__0.2",  "NSE__power__0.2", "mae__power__0.2", "mse__power__0.2",
```

```
                      "KGE__boxcoxsantos", "NSE__boxcoxsantos", "mae__boxcoxsantos", "mse__boxcoxsantos
                      "KGEtang__log", "NSE__log", "mae__log", "mse__log",
                      "KGE__power__-0.5", "NSE__power__-0.5", "mae__power__-0.5", "mse__power__-0.5")
```

Define the subseasons for which the above defined metrics are calculated for:

```
# a list with names and arrays of two digits describing months used to calculate
# subseasonal validation metrics
val_subseason <- list(spring = c("02", "03", "04", "05"),
                      summer = c("06", "07", "08", "09"))
```

Calculate (sub)seasonal metrics. It will automatically calculate performance metrics for the whole year (all).

```
# calculate performance metrics for calibration period
perf_cal <- calc_subseasonal_validation_results(val_subseason, hydro_data$BasinObs$DatesR,
                                               split_indices$ind_cal, "calibration",
                                               col_name = "period",
                                               sim$Qsim, hydro_data$BasinObs$Qmm, val_crit_transfo
)

# calculate performance metrics for calibration period
perf_val <- calc_subseasonal_validation_results(val_subseason, hydro_data$BasinObs$DatesR,
                                               split_indices$ind_val, "validation",
                                               col_name = "period",
                                               sim$Qsim, hydro_data$BasinObs$Qmm, val_crit_transfo
)

# combine periods in one data frame
perf_df <- dplyr::bind_rows(perf_cal, perf_val)
```

Show the results which is a tibble. Lambda is the exponent if a power transformation is used ...

```
knitr::kable(head(perf_df))
```

crit	transfo	lambda	value	season	period
KGE	none	NA	0.8009880	spring	calibration
NSE	none	NA	0.6224816	spring	calibration
VE	none	NA	0.5858467	spring	calibration
pbias	none	NA	3.5000000	spring	calibration
mae	none	NA	1.8717143	spring	calibration
mse	none	NA	9.5795278	spring	calibration

This result can be stored as ascii file:

```
# write ascii results overview
write_ascii(
  file.path(output_folder, "perf_ascii.txt"),
  # to include also the parameters
  calibration_results,
  perf_df
)
```

All results can be stored as a binary

```
# for this split sim in calibration and validation periods
sim_list <- list()
```

```r
sim_list$cal <- subset_simulations(split_indices$ind_cal, sim)
sim_list$val <- subset_simulations(split_indices$ind_val, sim)

saveRDS(
  list(
    calibration = calibration_results,
    simulation_val = sim_list$val,
    simulation_cal = sim_list$cal,
    sim_more_info = sim_list$more_info,
    validation = perf_df
  ),
  file.path(output_folder, "results_binary.rds")
)
```

Next steps

Choose a different model, or a snow module/hydrological model combination, or a different calibration function (see `vignette("calibration_methods_overview")` for a complete list), or a different streamflow transformation for calibration (e.g. `error_crit_transfo <- "NSE__power__0.2"`) or do it in a different subbasin with filtering the example data to a different HSU_ID.