rechaRge: the HydroBudget model

Emmanuel Dubois, Yannick Marcon

2024-04-03

Contents

Welcome

This book provides users with some common workflows for performing groundwater recharge modeling. This book will teach you how to use the rechaRge R package (Dubois and Marcon, 2024). This material serves as an online companion for the manuscript "Simulation of long-term spatiotemporal variations in regional-scale groundwater recharge: contributions of a water budget approach in cold and humid climates" (Dubois et al., 2021).

HydroBudget (HB) is a spatially distributed groundwater recharge (GWR) model that computes a superficial water budget on grid cells of regional-scale watersheds with outputs aggregated into monthly time steps and with limited computational time. The model is open-source and was coded in R. This `rechaRge` R package is the result of the effort to make this code reusable and extensible.

This book is organized into different parts. In the *Introduction*, we provide practical information about the `rechaRge` R package. The *Simulation* part focuses on the **HydroBudget** model that comes with the R package, and how to use the `rechaRge`functions to run simulations, then save and visualize the results. The *Calibration and Sensitivity* part explores the usage of the `sensitivity` and of the `caRamel` R packages to find and evaluate the "best" sets of model parameters. Finally, the *Extensibility* section highlights how the generic functions of the `rechaRge` R package can be applied to a different groundwater recharge model.

Chapter 1

Introduction

1.1 Installation

The package will be submitted to the CRAN repository. In the meantime, the package installation options are:

- Using `remotes`

```r
remotes::install_github("gwrecharge/rechaRge", ref = "main")
```

- Using `pak`

```r
pak::pkg_install("gwrecharge/rechaRge@main")
```

1.2 Usage

Some ready to use data files are made available for documentation and testing purposes. This book is based on these examples, see the section 2.

Chapter 2

Simulation

2.1 HydroBudget Model

HydroBudget was developed as an accessible and computationally affordable model to simulate GWR over large areas (thousands of km2) and for long time periods (decades), in cold and humid climates. The model uses commonly available meteorological data (daily precipitation and temperature, spatialized if possible) and spatially distributed data (pedology, land cover, and slopes). It is calibrated with river flows and baseflows estimated with recursive filters. The model needs reasonable computational capacity to reach relatively short computational times. It is based on simplified representations of hydrological processes and is driven by eight parameters that need to be calibrated.

HydroBudget uses a degree-day snow model for snow accumulation and snowmelt, and a conceptual lumped reservoir to compute the soil water budget on a daily time step. For each grid cell and each time step, the calculation distributes precipitation as runoff (R), evapotranspiration (ET), and infiltration that can reach the saturated zone if geological conditions below the soil allow deep percolation. HB thus produces estimates of potential GWR. The daily results are compiled at a monthly time step.

2.2 Input data and parameters

Start with loading the `rechaRge` library.

```r
library(rechaRge)
```

Then load the input data for the simulation. In that case, the example datasets are available for download. The data input handler is based on `data.table::fread` function, then you can provide an URL like in the example or a local file path.

```
base_url <- "https://github.com/gwrecharge/rechaRge-book/raw/main/examples/input/"
input_rcn <- paste0(base_url, "rcn.csv.gz") # RCN values per RCN cell ID
input_climate <- paste0(base_url, "climate.csv.gz") # precipitation total in mm/d per
input_rcn_climate <- paste0(base_url, "rcn_climate.csv.gz") # relation between climate
```

Set the **HydroBudget** model with the parameters values (if you do not know which parameters to set, see the Calibration section):

```
HB <- rechaRge::new_hydrobudget(
  T_m = 2.1, # melting temperature (°C)
  C_m = 6.2, # melting coefficient (mm/°C/d)
  TT_F = -17.6, # Threshold temperature for soil frost (°C)
  F_T = 16.4, # Freezing time (d)
  t_API = 3.9, # Antecedent precipitation index time (d)
  f_runoff = 0.63, # Runoff factor (-)
  sw_m = 431, # Maximum soil water content (mm)
  f_inf = 0.07 # infiltration factor (-)
)
```

As the **HydroBudget** model expects some data structure (expected data, with predefined column names), set the column names mappings matching the input datasets:

```
HB$rcn_columns <- list(
  rcn_id = "cell_ID",
  RCNII = "RCNII",
  lon = "X_L93",
  lat = "Y_L93"
)
HB$climate_columns$climate_id <- "climate_cell"
HB$rcn_climate_columns <- list(
  climate_id = "climate_cell",
  rcn_id = "cell_ID"
)
```

Then define the simulation period (if not, the period will be discovered from the input data):

```
simul_period <- c(2010, 2017)
```

2.3 Simulation

Once the **HydroBudget** object is ready, compute the water budget using the model implementation:

```r
water_budget <- rechaRge::compute_recharge(
  HB,
  rcn = input_rcn,
  climate = input_climate,
  rcn_climate = input_rcn_climate,
  period = simul_period
)
```

The water budget data set is per year-month in each RCN cell:

- `vi`, the vertical inflow
- `t_mean`, the mean temperature
- `runoff`, the runoff
- `pet`, the potential evapotranspiration
- `aet`, the actual evapotranspiration
- `gwr`, the groundwater recharge
- `runoff_2`, the excess runoff

The head of this data set is:

year	month	vi	t_mean	runoff	pet	aet	gwr	runoff_2	delta_reservoir	rcn_id
2010	1	28.2	-7.3	21.5	1.6	1.6	10.2	0	-5.1	62097
2010	2	27.9	-5.7	7.9	4.6	4.6	8.8	0	6.6	62097
2010	3	83.0	1.0	30.8	19.5	19.5	16.1	0	16.6	62097
2010	4	68.2	7.7	24.9	51.5	51.5	15.1	0	-23.3	62097
2010	5	46.9	13.8	3.8	94.7	73.8	4.4	0	-35.2	62097
2010	6	107.9	17.0	33.3	114.7	81.1	0.2	0	-6.7	62097

2.4 Results

The simulation results can be reworked, summarized and visualized.

2.4.1 Save results in files

Start with defining the output folder:

```r
sim_dir <- file.path(tempdir(), paste0("simulation_HydroBudget_", format(Sys.time(), "
```

Then write the resulting water budget in different formats in this output folder:

- CSV

```r
# CSV
rechaRge::write_recharge_results(HB, water_budget, output_dir = sim_dir)
```

- NetCDF

```r
# NetCDF
rechaRge::write_recharge_results(HB, water_budget, output_dir = sim_dir, format = "nc"
  "lon" = list(
    longname = "Qc lambert NAD83 epsg32198 Est",
    unit = "m"
  ),
  "lat" = list(
    longname = "Qc lambert NAD83 epsg32198 North",
    unit = "m"
  )
))
```

- Rasters

```r
# Rasters
rechaRge::write_recharge_rasters(
  HB,
  water_budget = water_budget,
  input_rcn = input_rcn,
  crs = "+proj=lcc +lat_1=60 +lat_2=46 +lat_0=44 +lon_0=-68.5 +x_0=0 +y_0=0 +ellps=GRS
  output_dir = sim_dir
)
```

The output folder content should look like this:

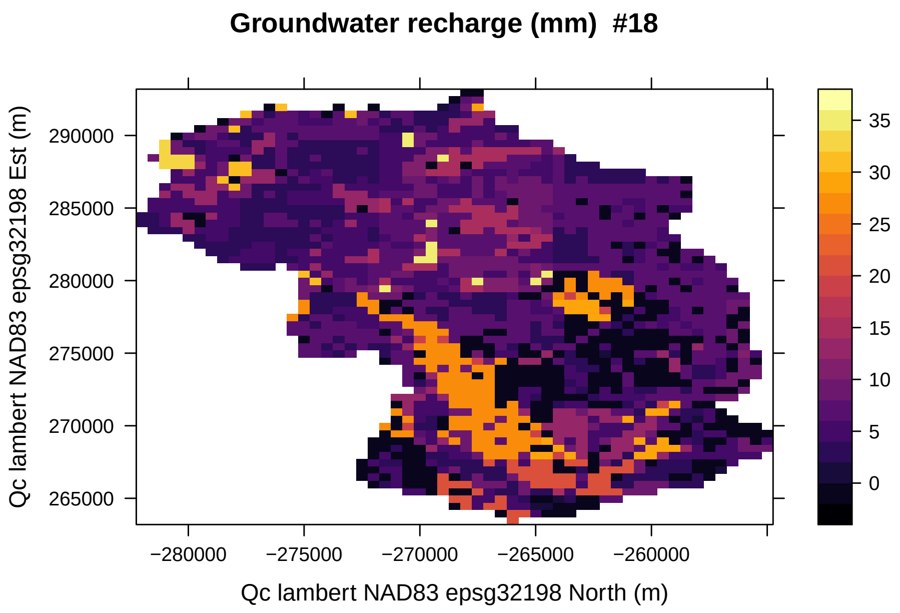
```r
# List simulation output files
list.files(sim_dir)
```

```
[1] "bilan_spat_month.csv"          "bilan_unspat_month.csv"
[3] "interannual_aet_NAD83.tif"     "interannual_gwr_NAD83.tif"
[5] "interannual_runoff_NAD83.tif" "water_budget.nc"
```

2.4.2 Data visualization

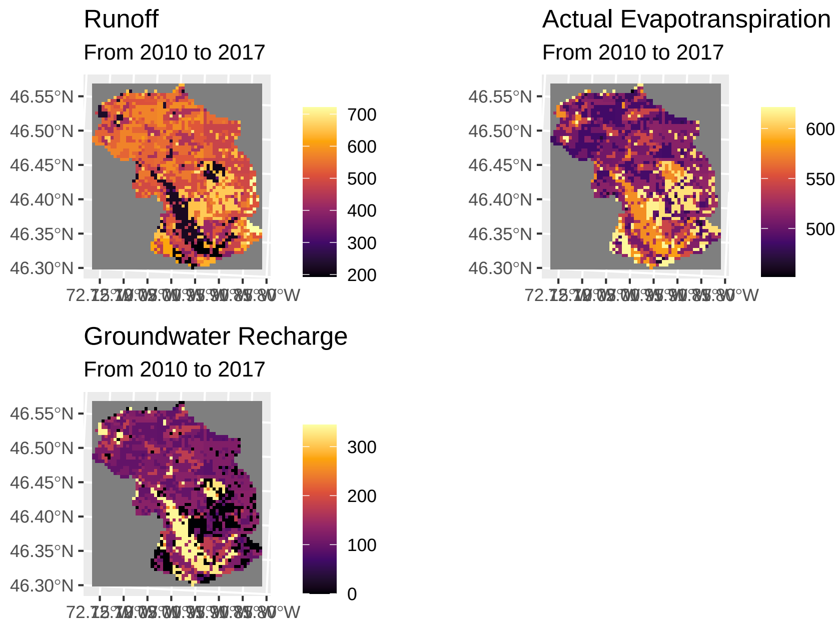
Visualize the saved NetCDF file:

```r
library(ncdf4)
library(lattice)
library(viridisLite)

# Extract GWR data
nc <- nc_open(file.path(sim_dir, "water_budget.nc"))
gwr <- ncvar_get(nc, "gwr")
gwratt <- ncatt_get(nc, "gwr")
lon <- ncvar_get(nc, "lon")
lonatt <- ncatt_get(nc, "lon")
lat <- ncvar_get(nc, "lat")
latatt <- ncatt_get(nc, "lat")
time <- ncvar_get(nc, "time")
nc_close(nc)

# Render the 18th month
month <- 18
gwr1 <- gwr[,,month]
grid <- expand.grid(lon=lon, lat=lat)
title <- paste0(gwratt$long_name, " (", gwratt$units, ") ", " #", month)
xlab <- paste0(latatt$long_name, " (", latatt$units, ")")
ylab <- paste0(lonatt$long_name, " (", lonatt$units, ")")
levelplot(gwr1 ~ lon * lat, data=grid, pretty=T, col.regions=inferno(100),
          main=title, xlab=xlab, ylab=ylab)
```


Groundwater recharge (mm) #18
Qc lambert NAD83 epsg32198 Est (m)
290000
285000
280000
275000
270000
265000
−280000
−275000
−270000
−265000
−260000
Qc lambert NAD83 epsg32198 North (m)
35
30
25
20
15
10
5
0

Visualize the saved raster files:

```r
library(tidyterra)
library(terra)
library(ggplot2)
library(cowplot)
subtitle <- ifelse(simul_period[1] == simul_period[2],
  paste0("In ", simul_period[1]),
  paste0("From ", simul_period[1], " to ", simul_period[2])
)
runoff <- terra::rast(file.path(sim_dir, "interannual_runoff_NAD83.tif"))
runoffplot <- ggplot() +
  geom_spatraster(data = runoff) +
  scale_fill_viridis_c(option = "inferno") +
  labs(
    fill = "",
    title = "Runoff",
    subtitle = subtitle
  )
aet <- terra::rast(file.path(sim_dir, "interannual_aet_NAD83.tif"))
aetplot <- ggplot() +
  geom_spatraster(data = aet) +
  scale_fill_viridis_c(option = "inferno") +
  labs(
    fill = "",
    title = "Actual Evapotranspiration",
    subtitle = subtitle
  )
gwr <- terra::rast(file.path(sim_dir, "interannual_gwr_NAD83.tif"))
gwrplot <- ggplot() +
  geom_spatraster(data = gwr) +
  scale_fill_viridis_c(option = "inferno") +
  labs(
    fill = "",
    title = "Groundwater Recharge",
    subtitle = subtitle
  )
cowplot::plot_grid(runoffplot, aetplot, gwrplot)
```



Chapter 3

Calibration and Sensitity

We will make use of `recharge` API to perform calibration and sensitivity analysis with different tools.

3.1 Quality assessment

Following the previous example, we will need to load observations datasets:

```
 # relation between gaugins station and RCN cell IDs
input_rcn_gauging <- paste0(base_url, "rcn_gauging.csv.gz")
 # flow rates in mm/d
input_observed_flow <- paste0(base_url, "observed_flow.csv.gz")
input_alpha_lyne_hollick <- paste0(base_url, "alpha_lyne_hollick.csv.gz")
```

And we also need in this case to update the settings of the **HydroBudget** model object, so that column names match with the expected ones:

```
HB$rcn_gauging_columns <- list(
  rcn_id = "cell_ID",
  station_id = "gauging_stat"
)
HB$alpha_lyne_hollick_columns$station_id <- "station"
```

Then we can process the river flow observations and assess simulation quality:

```
quality <- recharge::evaluate_simulation_quality(
  HB,
  water_budget = water_budget,
  rcn_gauging = input_rcn_gauging,
  observed_flow = input_observed_flow,
  alpha_lyne_hollick = input_alpha_lyne_hollick,
  period = simul_period
)
```

The `rechaRge` package proposes an model-free implementation of the Kling-Gupta Efficiency algorithm, that can be used for quality evaluation. In the case of our example the quality measurements of interest are:

```
list(
  KGE_qtot_cal_mean = mean(quality$simulation_metadata$KGE_qtot_cal),
  KGE_qbase_cal_mean = mean(quality$simulation_metadata$KGE_qbase_cal))
```

```
$KGE_qtot_cal_mean
[1] 0.8549218

$KGE_qbase_cal_mean
[1] 0.7203224
```

3.2 Using sensitivity

The sensitivity R package can perform various sensitivity analysis.

3.2.1 Define model function

```
library(rechaRge)
library(data.table)

# Preload input data
# Quiet download
options(datatable.showProgress = FALSE)
# use input example files provided by the package
base_url <- "https://github.com/gwrecharge/rechaRge-book/raw/main/examples/input/"
input_rcn <- fread(paste0(base_url, "rcn.csv.gz"))
input_climate <- fread(paste0(base_url, "climate.csv.gz"))
input_rcn_climate <- fread(paste0(base_url, "rcn_climate.csv.gz"))
input_rcn_gauging <- fread(paste0(base_url, "rcn_gauging.csv.gz"))
input_observed_flow <- fread(paste0(base_url, "observed_flow.csv.gz"))
input_alpha_lyne_hollick <- fread(paste0(base_url, "alpha_lyne_hollick.csv.gz"))
# Simulation period
```

```r
simul_period <- c(2017, 2017)

hydrobudget_eval <- function(i) {
  # Calibration parameters
  HB <- recharge::new_hydrobudget(
    T_m = i[1],
    # melting temperature (°C)
    C_m = i[2],
    # melting coefficient (mm/°C/d)
    TT_F = i[3],
    # Threshold temperature for soil frost (°C)
    F_T = i[4],
    # Freezing time (d)
    t_API = i[5],
    # Antecedent precipitation index time (d)
    f_runoff = i[6],
    # Runoff factor (-)
    sw_m = i[7],
    # Maximum soil water content (mm)
    f_inf = i[8] # infiltration factor (-)
  )
  # Input data specific settings
  HB$rcn_columns <- list(
    rcn_id = "cell_ID",
    RCNII = "RCNII",
    lon = "X_L93",
    lat = "Y_L93"
  )
  HB$climate_columns$climate_id <- "climate_cell"
  HB$rcn_climate_columns <- list(climate_id = "climate_cell",
                                 rcn_id = "cell_ID")
  HB$rcn_gauging_columns <- list(rcn_id = "cell_ID",
                                 station_id = "gauging_stat")
  HB$alpha_lyne_hollick_columns$station_id <- "station"

  # Simulation with the HydroBudget model
  water_budget <- recharge::compute_recharge(
    HB,
    rcn = input_rcn,
    climate = input_climate,
    rcn_climate = input_rcn_climate,
    period = simul_period,
    workers = 1
  )
  # Evaluate simulation quality
  result <- rechaRge::evaluate_simulation_quality(
    HB,
    water_budget = water_budget,
    rcn_gauging = input_rcn_gauging,
    observed_flow = input_observed_flow,
    alpha_lyne_hollick = input_alpha_lyne_hollick,
    period = simul_period
  )

  return(c(
    mean(result$simulation_metadata$KGE_qtot_cal),
    mean(result$simulation_metadata$KGE_qbase_cal)
  ))
}
```

3.2.2 Run sensitivity analysis

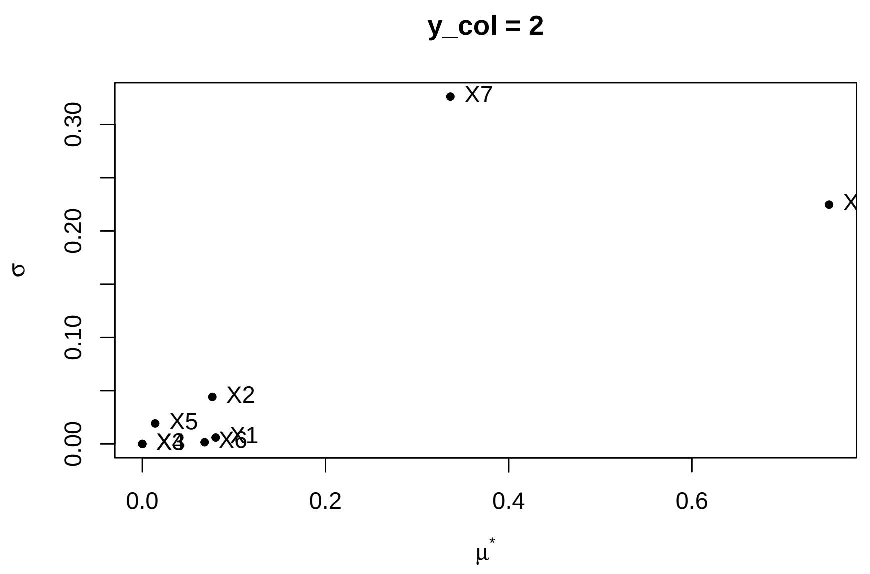
```r
library(sensitivity)
# Use future package to parallel
library(future.apply)

hydrobudget_sens <- function(X) {
  kge_hb <- as.matrix(t(
    future_apply(X, MARGIN = 1, FUN = hydrobudget_eval, future.seed = TRUE)))
  return(kge_hb)
}

# Number of variables
nvar <- 8
# Range of the parameters
binf <- c(1, 4, -20, 5, 3.05, 0.5, 160, 0.01)
bsup <- c(2.5, 6.5, -12, 30, 4.8, 0.6, 720, 0.05)

# parallel computation setting
plan(multisession, workers = 3)
#plan(sequential) # non parallel
sensitivity_results <- morris(
  model = hydrobudget_sens,
  factors = nvar,
  r = 2,
  design = list(type = "oat", levels = 5, grid.jump = 3),
  binf = binf,
  bsup = bsup)
```

3.2.3 Handle sensitivity results

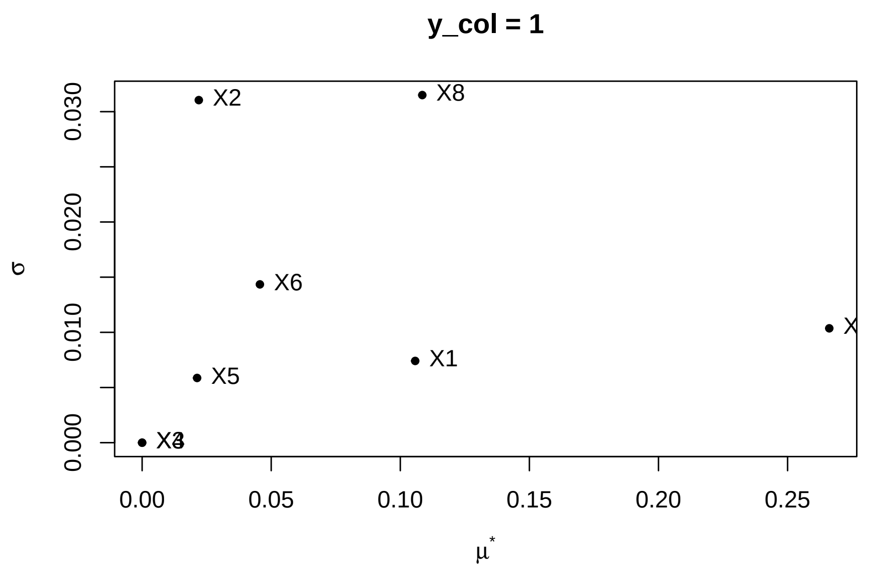
```
# Variable of interest
mu <- apply(sensitivity_results$ee, 3, function(M){
  apply(M, 2, mean)
})
mu.star <- apply(abs(sensitivity_results$ee), 3, function(M){
  apply(M, 2, mean)
})
sigma <- apply(sensitivity_results$ee, 3, function(M){
  apply(M, 2, sd)
})
sensitivity_eval <- data.table(mu = mu, mu.star = mu.star, sigma = sigma)
```

mu.ycol1	mu.ycol2	mu.star.ycol1	mu.star.ycol2	sigma.ycol1	sigma.ycol2
0.1057706	-0.0800860	0.1057706	0.0800860	0.0074096	0.0059664
-0.0169729	0.0764310	0.0219527	0.0764310	0.0310458	0.0441381
0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
0.0000000	0.0000000	0.0000000	0.0000000	0.0000000	0.0000000
0.0212918	-0.0140694	0.0212918	0.0140694	0.0058654	0.0192185
0.0456199	-0.0680606	0.0456199	0.0680606	0.0143449	0.0015621
-0.2661576	-0.3363263	0.2661576	0.3363263	0.0103630	0.3262131
0.1085154	0.7497127	0.1085154	0.7497127	0.0315015	0.2247081

```
# Plot
plot(sensitivity_results, y_col = 2)
title(main = "y_col = 2")
```



```r
plot.new()
plot(sensitivity_results, y_col = 1)
title(main = "y_col = 1")
```



3.3 Using caRamel

The caRamel R package can perform both calibration and sensitivity analysis.

3.3.1 Define objective function

We will start by defining the objective function to optimize: this function will run a simulation and quality evaluation and its returned values will be used by `caRamel` to measure the quality of the injected parameters.

```r
make_hydrobudget_eval <- function() {
  # Preload input data
  # Quiet download
  options(datatable.showProgress = FALSE)
  # use input example files provided by the package
  base_url <- "https://github.com/gwrecharge/rechaRge-book/raw/main/examples/input/"
  input_rcn <- fread(paste0(base_url, "rcn.csv.gz"))
  input_climate <- fread(paste0(base_url, "climate.csv.gz"))
  input_rcn_climate <- fread(paste0(base_url, "rcn_climate.csv.gz"))
  input_rcn_gauging <- fread(paste0(base_url, "rcn_gauging.csv.gz"))
  input_observed_flow <- fread(paste0(base_url, "observed_flow.csv.gz"))
  input_alpha_lyne_hollick <- fread(paste0(base_url, "alpha_lyne_hollick.csv.gz"))
  # Simulation period
  simul_period <- c(2017, 2017)

  hydrobudget_eval <- function(i) {
    # Calibration parameters
    HB <- rechaRge::new_hydrobudget(
      T_m = x[i, 1],
      # melting temperature (°C)
      C_m = x[i, 2],
      # melting coefficient (mm/°C/d)
      TT_F = x[i, 3],
      # Threshold temperature for soil frost (°C)
      F_T = x[i, 4],
      # Freezing time (d)
      t_API = x[i, 5],
      # Antecedent precipitation index time (d)
      f_runoff = x[i, 6],
      # Runoff factor (-)
      sw_m = x[i, 7],
      # Maximum soil water content (mm)
      f_inf = x[i, 8] # infiltration factor (-)
    )
    # Input data specific settings
    HB$rcn_columns <- list(
      rcn_id = "cell_ID",
      RCNII = "RCNII",
      lon = "X_L93",
      lat = "Y_L93"
    )
    HB$climate_columns$climate_id <- "climate_cell"
    HB$rcn_climate_columns <- list(climate_id = "climate_cell",
                                   rcn_id = "cell_ID")
    HB$rcn_gauging_columns <- list(rcn_id = "cell_ID",
                                   station_id = "gauging_stat")
    HB$alpha_lyne_hollick_columns$station_id <- "station"

    # Simulation with the HydroBudget model
    rechaRge::with_verbose(FALSE)
    water_budget <- rechaRge::compute_recharge(
      HB,
      rcn = input_rcn,
      climate = input_climate,
      rcn_climate = input_rcn_climate,
      period = simul_period,
      workers = 1 # do not parallelize, caRamel will do it
    )

    # Evaluate simulation quality
    quality <- rechaRge::evaluate_simulation_quality(
      HB,
      water_budget = water_budget,
      rcn_gauging = input_rcn_gauging,
      observed_flow = input_observed_flow,
      alpha_lyne_hollick = input_alpha_lyne_hollick,
      period = simul_period
    )

    return(c(
      mean(quality$simulation_metadata$KGE_qtot_cal),
      mean(quality$simulation_metadata$KGE_qbase_cal)
    ))
  }

  return(hydrobudget_eval)
}
```

3.3.2 Run calibration analysis

Then perform calibration with sensitivity:

```
library(caRamel)

# Number of objectives
nobj <- 2
# Number of variables
nvar <- 8
# All the objectives are to be maximized
minmax <- c(TRUE, TRUE)
# Ranges of the parameters
bounds <- matrix(nrow = nvar, ncol = 2)
bounds[, 1] <- c(1, 4, -20, 5, 3.05, 0.5, 160, 0.01)
bounds[, 2] <- c(2.5, 6.5, -12, 30, 4.8, 0.6, 720, 0.05)

calibration_results <- caRamel(
  nobj = nobj,
  nvar = nvar,
  minmax =  minmax,
  bounds = bounds,
  func = make_hydrobudget_eval(),
  prec = matrix(0.01, nrow = 1, ncol = nobj),
  sensitivity = FALSE, # you can include sensitivity analysis
  archsize = 100,# adjust to relevant value
  popsize = 20,  # adjust to relevant value
  maxrun = 20,   # adjust to relevant value
  carallel = 1,  # do parallel ...
  numcores = 2   # ... on 2 cores
)
```

3.3.3 Handle calibration results

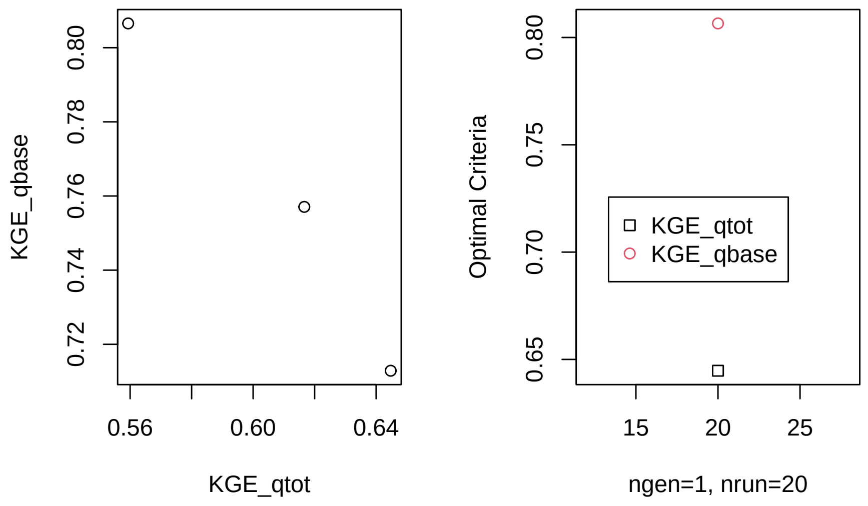
Make use of calibration results, by merging simulation outputs with front objectives, to get more readable parameters, ordered by “best fit” score:

```
output_front <- data.table(cbind(
  calibration_results$parameters, calibration_results$objectives))
colnames(output_front) <-
  c("T_m", "C_m", "TT_F", "F_T", "t_API", "f_runoff", "sw_m", "f_inf", "KGE_qtot", "KGE_qbase")
y <- 0.6 # choose your KGE weight criteria
output_front[, `:=`(KGE_score = (KGE_qtot * (1 - y) + KGE_qbase * y))]
output_front <- output_front[order(KGE_score, decreasing = TRUE)]
```

T_m	C_m	TT_F	F_T	t_API	f_runoff	sw_m	f_inf	KGE_
1.233346	5.623252	-15.49015	29.301143	3.811963	0.5961505	269.4135	0.0321405	0.559
1.492015	4.958859	-13.23603	5.717497	4.078846	0.5521782	253.7823	0.0358564	0.616
1.352516	4.913283	-17.39437	9.707676	3.164533	0.5974775	182.5930	0.0162182	0.644

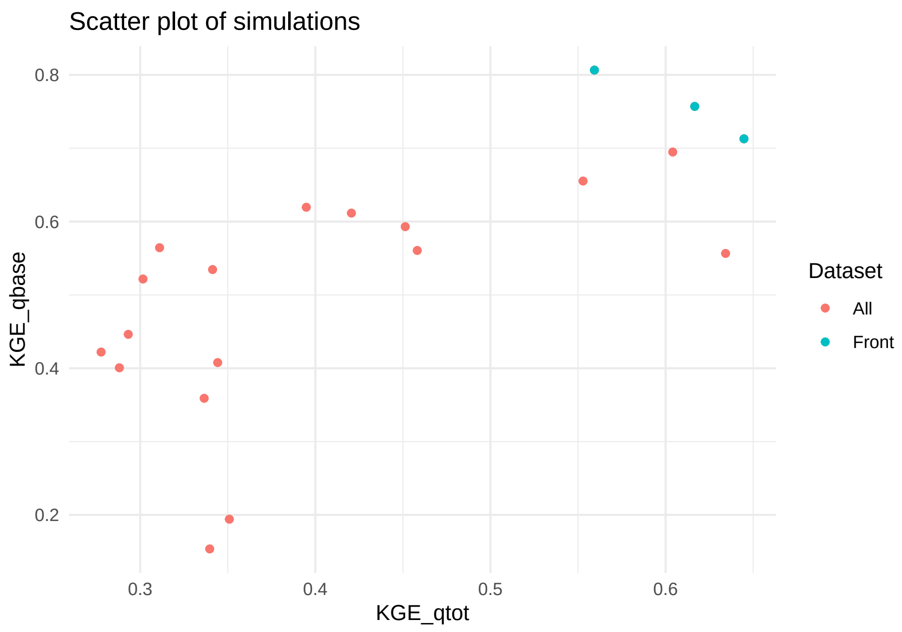
Display the results with `caRamel`'s plotting feature:

```r
# Plot using caRamel
plot_caramel(calibration_results, objnames = c("KGE_qtot", "KGE_qbase"))
```



Scatter plot the calibration's simulations, with the resulting Pareto front:

```r
# Plot all using ggplot
library(ggplot2)
front <- data.table(calibration_results$objectives)
colnames(front) <- c("KGE_qtot", "KGE_qbase")
all <- data.table(calibration_results$total_pop[, 9:10])
colnames(all) <- c("KGE_qtot", "KGE_qbase")
combined_data <- rbind(all, front)
combined_data$group <- c(rep("All", nrow(all)), rep("Front", nrow(front)))
ggplot(combined_data, aes(x = KGE_qtot, y = KGE_qbase, color = group)) +
  geom_point() +
  labs(title = "Scatter plot of simulations",
       x = "KGE_qtot", y = "KGE_qbase",
       color = "Dataset") +
  theme_minimal()
```



3.3.4 Evaluate uncertainty

For each set of parameters proposed by `caRamel`, run a simulation and then evaluate the uncertainty of the set of parameters identified as being the "best fit".

```r
# for each proposed set of parameters, run the water budget simulation
param_ids <- as.numeric(rownames(output_front))

# run simulations
water_budgets <- lapply(param_ids, FUN = function(i) {
  x <- output_front
  HB <- rechaRge::new_hydrobudget(
    T_m = as.numeric(x[i, 1]),
    C_m = as.numeric(x[i, 2]),
    TT_F = as.numeric(x[i, 3]),
    F_T = as.numeric(x[i, 4]),
    t_API = as.numeric(x[i, 5]),
    f_runoff = as.numeric(x[i, 6]),
    sw_m = as.numeric(x[i, 7]),
    f_inf = as.numeric(x[i, 8])
  )
  # Input data specific settings
  HB$rcn_columns <- list(
    rcn_id = "cell_ID",
    RCNII = "RCNII",
    lon = "X_L93",
    lat = "Y_L93"
  )
  HB$climate_columns$climate_id <- "climate_cell"
  HB$rcn_climate_columns <- list(climate_id = "climate_cell",
                                 rcn_id = "cell_ID")

  # Simulation with the HydroBudget model
  rechaRge::compute_recharge(
    HB,
    rcn = input_rcn,
    climate = input_climate,
    rcn_climate = input_rcn_climate,
    period = simul_period,
    workers = 2
  )
})
```

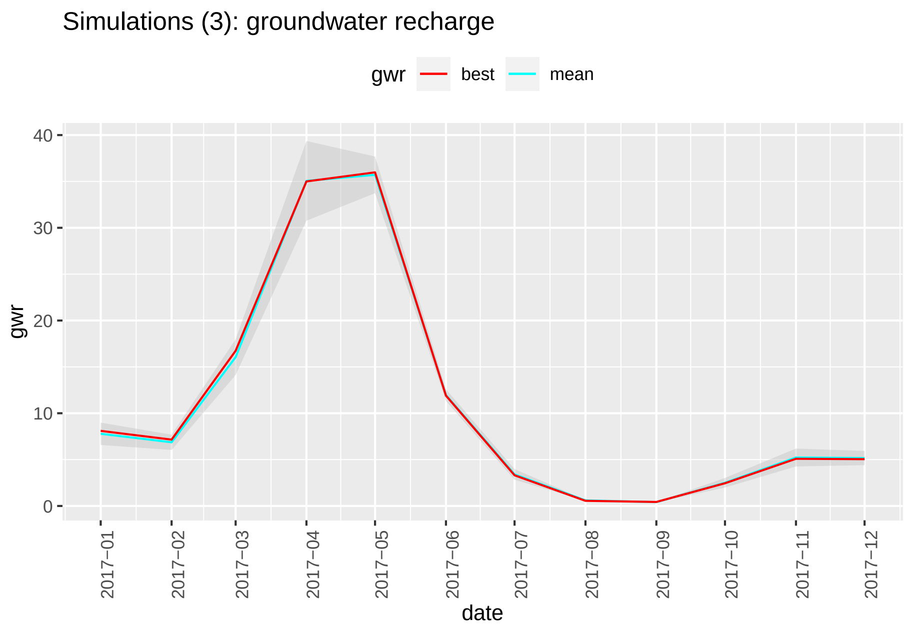
The following function will, for a given metric (e.g. `gwr`, `runoff` etc.):

- make one row per year-month, one column for the value of the "best fit" simulation, one for the mean of all the simulated values, and one for the standard deviation between all the simulated values.
- plot the corresponding time series, showing how the "best fit" compares with the mean and the standard deviation range.

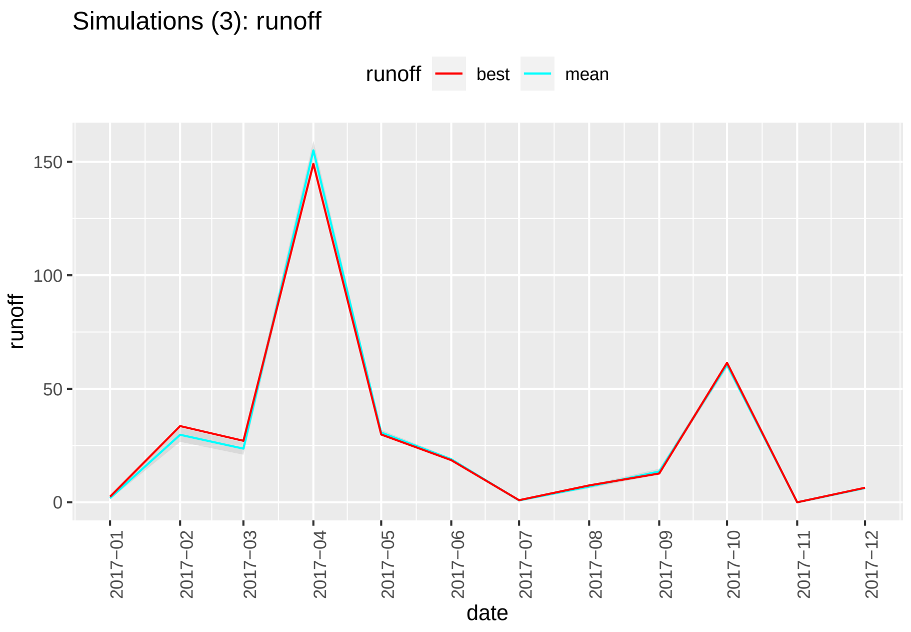
```
# make one row per year-month and one column per simulated measure
plot_metric <- function(water_budgets, metric, title = NULL) {
  # spatialized: add metric values starting from best fit
  metrics <- data.table(year = water_budgets[[1]]$year,
                        month = water_budgets[[1]]$month)
  for (i in param_ids) {
    param_id <- paste0(metric, i)
    set(metrics, j = param_id, value = water_budgets[[i]][[metric]])
  }

  # non-spatialized: calculate mean, group by year-month
  metrics_monthly <- unique(metrics, by = c("year", "month"))[, c("year", "month")]
  for (i in param_ids) {
    param_id <- paste0(metric, i)
    metrics_id <- metrics[ , .(mean = mean(get(param_id))), by = c("year", "month")]
    set(metrics_monthly, j = param_id, value = metrics_id$mean)
  }
  # add mean and sd for each year-month row
  ym_cols <- c("year", "month")
  set(metrics_monthly, j = "mean", value = apply(metrics_monthly[, !..ym_cols], 1, mean))
  set(metrics_monthly, j = "sd", value = apply(metrics_monthly[, !..ym_cols], 1, sd))
  set(metrics_monthly, j = "date",
      value = as.POSIXct(paste(metrics_monthly$year, metrics_monthly$month, "1", sep="-")))
  set(metrics_monthly, j = "min", value = metrics_monthly$mean - metrics_monthly$sd)
  set(metrics_monthly, j = "max", value = metrics_monthly$mean + metrics_monthly$sd)
  colnames(metrics_monthly)[[3]] <- "best"

  # plot uncertainty
  library(ggplot2)
  library(scales)
  ggplot(data = metrics_monthly, aes(x = date)) +
    geom_ribbon(aes(ymin = min, ymax = max), fill = "gray", alpha = 0.4) +
    geom_line(aes(y = mean, color = "mean")) +
    geom_line(aes(y = best, color = "best")) +
    labs(title = title, color = metric, x = "date", y = metric) +
    scale_color_manual(values = c(best = "red", mean = "cyan")) +
    scale_x_datetime(date_labels = "%Y-%m", breaks = date_breaks("months")) +
    theme(axis.text.x = element_text(angle = 90), legend.position = "top")
}
```

We can now visualize the uncertainty for different metrics:

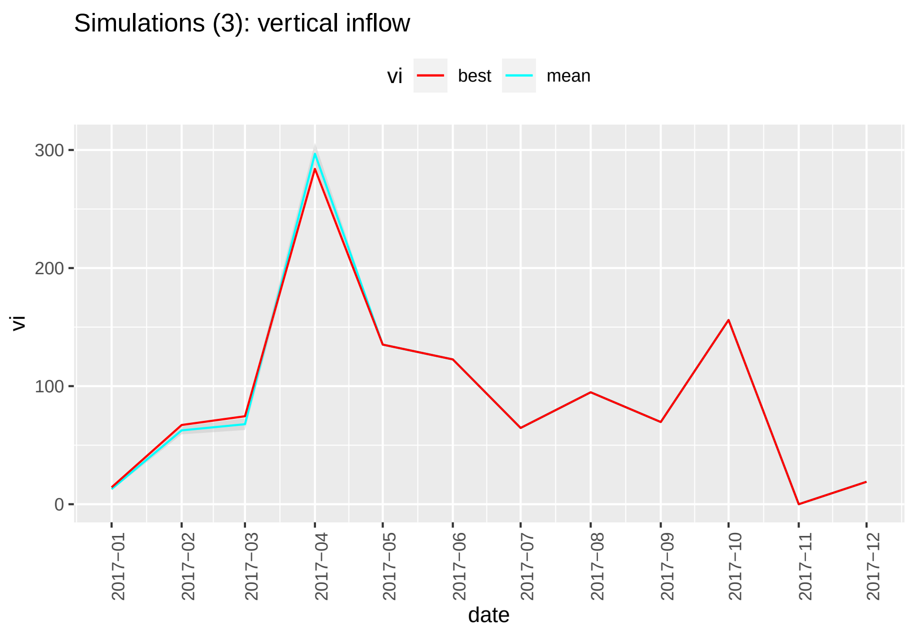
```
plot_metric(water_budgets, "gwr",
            title = paste0("Simulations (", length(param_ids),"): groundwater recharge"))
```



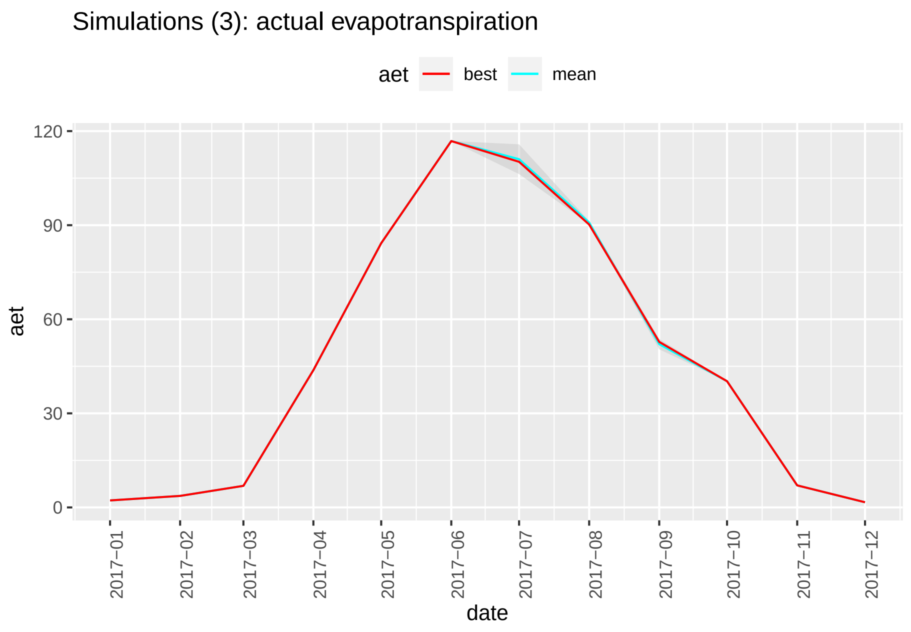
```
plot_metric(water_budgets, "runoff",
            title = paste0("Simulations (", length(param_ids),"): runoff"))
```



```r
plot_metric(water_budgets, "vi",
            title = paste0("Simulations (", length(param_ids),"): vertical inflow"))
```



```r
plot_metric(water_budgets, "aet",
            title = paste0("Simulations (", length(param_ids),"): actual evapotranspiration"))
```


Simulations (3): actual evapotranspiration
aet
best
mean
120
90
60
30
0
aet
2017−01
2017−02
2017−03
2017−04
2017−05
2017−06
2017−07
2017−08
2017−09
2017−10
2017−11
2017−12
date

Chapter 4

Extensibility

4.1 Generic functions

The `rechaRge` package exposes the following S3 generic functions:

- `compute_recharge`, is the main function that performs the simulation computations according to the provided model parameters and input data. The result of this function is an input of the following functions.
- `write_recharge_results`, will write the simulation results into data files, summarized results can be included and various output file formats can be supported, depending on the model.
- `write_recharge_rasters`, will write the simulation results into raster files.
- `evaluate_simulation_quality`, is evaluating the quality of simulation output. This function should be used when making calibration and sensitivity analysis.

You can use the reference implementation of **HydroBudget** model to extend this API with your own model. For this you will need to start with the definition of a model specific object. In the case of **HydroBudget**, this object holds the parameters of the model and some helper information about the input source format. See the `new_hydrobudget` function.

4.2 Other languages

If you have a model designed in Python, you could consider using the `reticulate` R package that facilitates the binding between the R and the Python execution environments.

Chapter 5

About R

R is a free, open-source software and programming language developed in 1995 at the University of Auckland as an environment for statistical computing and graphics (Ikaha and Gentleman, 1996). Since then R has become one of the dominant software environments for data analysis and is used by a variety of scientific disciplines. CRAN Tasks provides an excellent overview of existing R packages for a given discipline (see for instance Envirometrics Task View; Spatial Task View).

R offers numerous advantages, such as:

1. Free and Open source
2. Reproducible Research
    - repeatable:
        - code + output in a single document
        - easier the re-analyses
    - scalable: applicable to small or large datasets
    - extensible: several
3. Getting help
    - Numerous Discipline Specific R Groups
    - Numerous Local R User Groups (including R-Ladies Groups)
    - Stack Overflow
4. Learning Resources
    - R books
    - (Free Online) R Books

Bibliography

Dubois, E., Larocque, M., Gagné, S., and Meyzonnat, G. (2021). Simulation of long-term spatiotemporal variations in regional-scale groundwater recharge: contributions of a water budget approach in cold and humid climates. *Hydrology and Earth System Sciences*, 25(12):6567–6589.

Dubois, E. and Marcon, Y. (2024). *rechaRge: Groundwater Recharge Model*. R package version 0.10, https://gwrecharge.github.io/rechaRge/.