Optimisation sans advection

2024-08-21

```
# setwd("./script")
library(generain)
library(reshape2)
library(ggplot2)
source("load_libraries.R")
```

Anciennes simulations

```
sim_BR <- function(beta1, beta2, alpha1, alpha2, x, y, z, n.BR) {
  ## Setup
  RandomFields::RFoptions(spConform=FALSE)
  lx <- length(sx <- seq_along(x))
  ly <- length(sy <- seq_along(y))
  lz <- length(sz <- seq_along(z))
  ## Model-Variogram BuhlCklu
  modelBuhlCklu <- RandomFields::RMfbm(alpha=alpha1, var=beta1, proj=1) +
                   RandomFields::RMfbm(alpha=alpha1, var=beta1, proj=2) +
                   RandomFields::RMfbm(alpha=alpha2, var=beta2, proj=3)

  ## Construct grid
  Nxy <- lx * ly
  N <- Nxy * lz
  grid <- matrix(0, nrow=N, ncol=3) # (N,3)-matrix

  for (i in sx)
    for (j in seq_len(ly*lz))
      grid[i+(j-1)*ly, 1] <- i

  for (i in sy)
    for (j in sx)
      for(k in sz)
        grid[j+lx*(i-1)+(k-1)*Nxy, 2] <- i

  for (i in sz)
    for (j in seq_len(Nxy))
      grid[j+Nxy*(i-1), 3] <- i

  ## Construct shifted variogram
  Varm1 <- vapply(seq_len(N), function(n)
      RandomFields::RFvariogram(modelBuhlCklu,
        x=sx-grid[n,1],
        y=sy-grid[n,2],
```

```
        z=sz-grid[n,3]),
        array(NA_real_, dim=c(lx, ly, lz))) ## => (lx, ly, lz, N)-array

  ## Main
  set.seed(123)
  Z <- array(, dim=c(lx, ly, lz, n.BR)) # 4d array
  E <- matrix(rexp(n.BR * N), nrow=n.BR, ncol=N)
  for (i in seq_len(n.BR)) { ## n=1
    V <- 1/E[i,1]
    W <- RandomFields::RFsimulate(modelBuhlCklu, x, y, z, n=1)
    Y <- exp(W - W[1] - Varm1[,,,1])
    Z[,,,i] <- V * Y
    ## n in {2,..,N}
    for(n in 2:N) {
      Exp <- E[i,n]
      V <- 1/Exp
      while(V > Z[N*(i-1)+n]) {
        W <- RandomFields::RFsimulate(modelBuhlCklu, x, y, z)
        Y <- exp(W - W[n] - Varm1[,,,n])
        if(all(V*Y[seq_len(n-1)] < Z[(N*(i-1)+1):(N*(i-1)+(n-1))]))
          Z[,,,i] <- pmax(V*Y, Z[,,,i])
          Exp <- Exp + rexp(1)
          V <- 1/Exp
      }
    }
  }
  ## Return
  Z
}
```

```
true_param <- c(0.4, 0.2, 1.5, 1)
ngrid <- 5
spa <- 1:ngrid
nsites <- ngrid^2 # if the grid is squared
temp <- 1:300
n.BR <- 1

# generate the simulations
# BR <- sim_BR(true_param[1] * 2, true_param[2] * 2, true_param[3], true_param[4],
#              spa, spa, temp, n.BR)
# save_simulations(BR, ngrid, n.BR,
#         folder = paste0("../data/simulations_BR/oldsim_", ngrid^2, "s_",
#                               length(temp), "t/"),
#                   file = paste0("br_", ngrid^2, "s_",
#                               length(temp), "t"), forcedind = 1)

# load the simulations
file_path <- paste0("../data/simulations_BR/oldsim_", ngrid^2, "s_",
                              length(temp), "t/br_",
                      ngrid^2, "s_", length(temp), "t_", 1, ".csv")
simu_df <- read.csv(file_path)
```

```r
nsites <- ncol(simu_df)
sites_coords <- generate_grid_coords(sqrt(nsites))
dist_mat <- get_dist_mat(sites_coords,
                         latlon = FALSE) # distance matrix
df_dist <- reshape_distances(dist_mat) # reshape the distance matrix
```

Validation du modèle de Buhl séparable

Pour la simulation avec 25 sites et 300 pas de temps et un quantile de 0.9 on obtient une bonne estimation des paramètres du modèle de Buhl séparable avec WLSE.

En revanche, pour une simulation avec 49 sites et 300 pas de temps avec un quantile de 0.9 on obtient une mauvaise estimation notamment pour le paramètre β_1 et le choix du quantile a un impact sur l'estimation des paramètres mais aucune valeur de quantile ne permet une bonne estimation de tous les paramètres.

```r
sites_coords <- generate_grid_coords(sqrt(nsites))
df_lags <- get_lag_vectors(sites_coords, true_param,
                          hmax = sqrt(17), tau_vect = 0:10)

hmax <- sqrt(17)
q <- 0.9
chispa <- spatial_chi_alldist(df_dist, simu_df, quantile = q,
                               hmax = hmax)
spa_estim <- get_estimate_variospa(chispa, weights = "exp", summary = FALSE)
print(spa_estim)
```

```
## [1] 0.3724744 1.5253168
```

```r
q <- 0.9
tmax <- 10
chitemp <- temporal_chi(simu_df, tmax = tmax, quantile = q)
temp_estim <- get_estimate_variotemp(chitemp, tmax, npoints = ncol(simu_df),
                                      weights = "exp", summary = FALSE)
print(temp_estim)
```

```
## [1] 0.1478522 0.9514819
```

```r
df_result <- data.frame(beta1 =  spa_estim[1],
                        alpha1 = spa_estim[2],
                        beta2 = temp_estim[1],
                        alpha2 = temp_estim[2])
colnames(df_result) <- c("beta1", "alpha1", "beta2", "alpha2")

df_valid <- get_criterion(df_result, true_param)
colnames(df_valid) <- c("estim", "rmse", "mae")
print(df_valid)
```

```
##            estim       rmse        mae
## beta1  0.3724744 0.02752561 0.02752561
## beta2  0.1478522 0.05214780 0.05214780
## alpha1 1.5253168 0.02531678 0.02531678
## alpha2 0.9514819 0.04851810 0.04851810
```

Optimisation

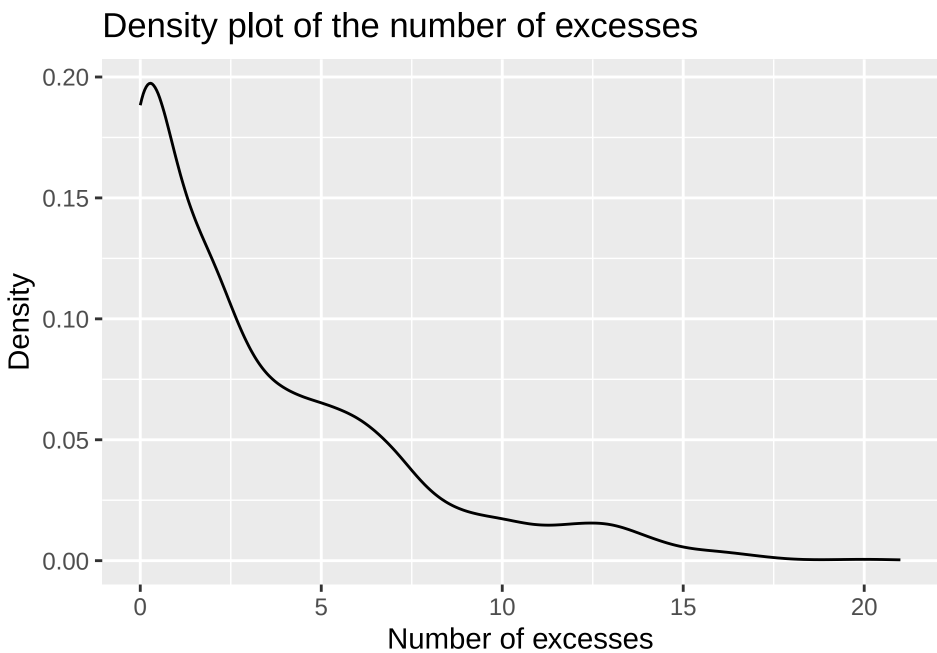
Get excesses

```r
empirical_excesses <- function(data_rain, quantile, df_lags) {
  excesses <- df_lags # copy the dataframe
  unique_tau <- unique(df_lags$tau) # unique temporal lags

  for (t in unique_tau) { # loop over temporal lags
    df_h_t <- df_lags[df_lags$tau == t, ] # get the dataframe for each lag

    for (i in seq_len(nrow(df_h_t))) { # loop over each pair of sites
      # get the indices of the sites
      ind_s2 <- as.numeric(as.character(df_h_t$s2[i]))
      ind_s1 <- df_h_t$s1[i]

      # get the data for the pair of sites
      rain_cp <- data_rain[, c(ind_s1, ind_s2), drop = FALSE]
      rain_cp <- na.omit(rain_cp)
      colnames(rain_cp) <- c("s1", "s2")

      Tmax <- nrow(rain_cp) # number of time steps
      rain_nolag <- rain_cp$s1[1:(Tmax - t)] # get the data without lag
      rain_lag <- rain_cp$s2[(1 + t):Tmax] # get the data with lag

      n <- length(rain_nolag) # number of observations
      # transform the data in uniform data
      rain_unif <- cbind(rank(rain_nolag) / (n + 1), rank(rain_lag) / (n + 1))
      # get the conditional excesses on s2
      cp_cond <- rain_unif[rain_unif[, 2] > quantile, , drop = FALSE]
      joint_excesses <- sum(cp_cond[, 1] > quantile) # number of excesses for s1
                                                      # given those of s2
      marginal_excesses <- nrow(cp_cond) # number excesses for s2

      # store the number of excesses
      excesses$nj[excesses$s1 == ind_s1
                    & excesses$s2 == ind_s2
                    & excesses$tau == t] <- marginal_excesses
      excesses$kij[excesses$s1 == ind_s1
                  & excesses$s2 == ind_s2
                  & excesses$tau == t] <- joint_excesses
    }
  }
  return(excesses)
}

q <- 0.93
excesses <- empirical_excesses(simu_df, quantile = q, df_lags = df_lags)
print(head(excesses))
```

```
##   s1 s2 h1 h2 tau hnorm nj kij
## 1  1  2  0  1   0     1 21  17
## 2  1  2  0  1   1     1 20  15
```

```
## 3   1   2   0   1    2      1  20   14
## 4   1   2   0   1    3      1  20   13
## 5   1   2   0   1    4      1  20   12
## 6   1   2   0   1    5      1  20   12
```

```
# density plot of the number of excesses
ggplot(excesses, aes(x = kij)) +
  geom_density() +
  labs(title = "Density plot of the number of excesses",
       x = "Number of excesses", y = "Density")
```




```
neg_ll <- function(params, simu, df_lags, locations,
                   latlon = FALSE, quantile = 0.9,
                   simu_exp = FALSE, excesses = NULL) {
  hmax <- max(df_lags$hnorm)
  tau <- unique(df_lags$tau)
  # print(params)

  if (is.null(excesses)) {
    excesses <- empirical_excesses(simu, quantile, df_lags)
  }

  lower.bound <- c(1e-6, 1e-6, 1e-6, 1e-6)
  upper.bound <- c(Inf, Inf, 1.999, 1.999)
  if (length(params) == 6) {
    lower.bound <- c(lower.bound, -1e-6, -1e-6)
    upper.bound <- c(upper.bound, Inf, Inf)
  }

  # Check if the parameters are in the bounds
  if (any(params < lower.bound) || any(params > upper.bound)) {
    message("out of bounds")
```

```r
    return(1e9)
  }

  if (length(params) == 6) { # if we have the advection parameters
    df_lags <- get_lag_vectors(locations, params, tau = tau, hmax = hmax)
    excesses <- empirical_excesses(simu, quantile, df_lags)
  }

  nj <- excesses$nj # number of marginal excesses
  kij <- excesses$kij # number of joint excesses
  chi <- theorical_chi(params, df_lags) # get chi matrix
  # transform in chi vector
  chi_vect <- as.vector(chi$chi)
  chi_vect <- ifelse(chi_vect <= 0, 0.000001, chi_vect) # avoid log(0)

  non_excesses <- nj - kij # number of non-excesses
  # log-likelihood vector
  ll_vect <- kij * log(chi_vect) + non_excesses * log(1 - chi_vect)

  # final negative log-likelihood
  nll <- -sum(ll_vect, na.rm = TRUE)
  return(nll)
}
```

Pour la simulation avec 25 sites et 300 pas de temps avec un quantile de 0.9 on obtient une bonne estimation des paramètres du modèle de Buhl séparable avec l'optimisation de la vraisemblance composite. En changeant le quantile, on garde des résultats similaires pour $q > 0.9$ et proche de 0.9. Pour $q <= 0.9$, on obtient des résultats moins bons pour le paramètre α_1.

Pour la simulation avec 49 sites et 300 pas de temps avec un quantile de 0.9 on obtient une mauvaise estimation des paramètres du modèle avec l'optimisation de la vraisemblance composite. Les alphas sont fortement sous-estimés.

Peu importe la méthode d'optimisation, on obtient des résultats similaires.

```r
result <- optim(par = c(true_param), fn = neg_ll,
                        simu = simu_df,
                        quantile = q,
                        excesses = excesses,
                        df_lags = df_lags,
                        locations = sites_coords,
                        method = "CG",
                        control = list(parscale = c(1, 1, 1, 1),
                                       maxit = 10000))
print(result$convergence) # 0 if it has converged
```

```
## [1] 0
```

```r
print(result$par)
```

```
## [1] 0.3296619 0.1608786 1.5400254 1.0345042
```

```r
rmse <- sqrt((result$par - true_param)^2)
df_rmse <- data.frame(estim = result$par, rmse = rmse)
rownames(df_rmse) <- c("beta1", "beta2", "alpha1", "alpha2")
print(t(df_rmse))
```

```
##            beta1      beta2     alpha1     alpha2
## estim 0.32966187 0.16087856 1.54002539 1.03450425
## rmse  0.07033813 0.03912144 0.04002539 0.03450425
```

Ajout de l'advection dans la simulation

```r
x <- 1:3
y <- 1:3
z <- 1:2

lx <- length(sx <- seq_along(x))
ly <- length(sy <- seq_along(y))
lz <- length(sz <- seq_along(z))

## Construct grid
Nxy <- lx * ly
N <- Nxy * lz
grid <- matrix(0, nrow = N, ncol = 3) # (N,3)-matrix

# Spatial x coordinates
for (i in sx) {
  for (j in seq_len(ly * lz)) {
    grid[i + (j - 1) * ly, 1] <- i
  }
}

# Spatial y coordinates
for (i in sy) {
  for (j in sx) {
    for (k in sz) {
      grid[j + lx * (i - 1) + (k - 1) * Nxy, 2] <- i
    }
  }
}

# Spatial z coordinates
for (i in sz) {
  for (j in seq_len(Nxy)) {
    grid[j + Nxy * (i - 1), 3] <- i
  }
}

print(head(grid))
```

```
##      [,1] [,2] [,3]
## [1,]    1    1    1
```

```
## [2,]    2    1    1
## [3,]    3    1    1
## [4,]    1    2    1
## [5,]    2    2    1
## [6,]    3    2    1
```

```
# Advection vector
Vx <- 0.5
Vy <- 0.5
V <- c(Vx, Vy)

grid_adv <- grid
# Construct shifted grid
grid_adv[, 1:2] <- grid_adv[, 1:2] - grid_adv[, 3] * V


print(head(grid_adv))
```

```
##      [,1] [,2] [,3]
## [1,]  0.5  0.5    1
## [2,]  1.5  0.5    1
## [3,]  2.5  0.5    1
## [4,]  0.5  1.5    1
## [5,]  1.5  1.5    1
## [6,]  2.5  1.5    1
```

```
sim_BR_adv <- function(beta1, beta2, alpha1, alpha2, x, y, z, n.BR,
                       adv = c(0, 0)) {
  ## Setup
  RandomFields::RFoptions(spConform = FALSE)
  lx <- length(sx <- seq_along(x))
  ly <- length(sy <- seq_along(y))
  lz <- length(sz <- seq_along(z))

  ## Model-Variogram BuhlCklu
  modelBuhlCklu <- RandomFields::RMfbm(alpha = alpha1, var = beta1, proj = 1) +
                   RandomFields::RMfbm(alpha = alpha1, var = beta1, proj = 2) +
                   RandomFields::RMfbm(alpha = alpha2, var = beta2, proj = 3)

  ## Construct grid
  Nxy <- lx * ly
  N <- Nxy * lz
  grid <- matrix(0, nrow = N, ncol = 3) # (N,3)-matrix

  for (i in sx)
    for (j in seq_len(ly * lz))
      grid[i + (j - 1) * ly, 1] <- i

  for (i in sy)
    for (j in sx)
      for (k in sz)
        grid[j + lx * (i - 1) + (k - 1) * Nxy, 2] <- i
```

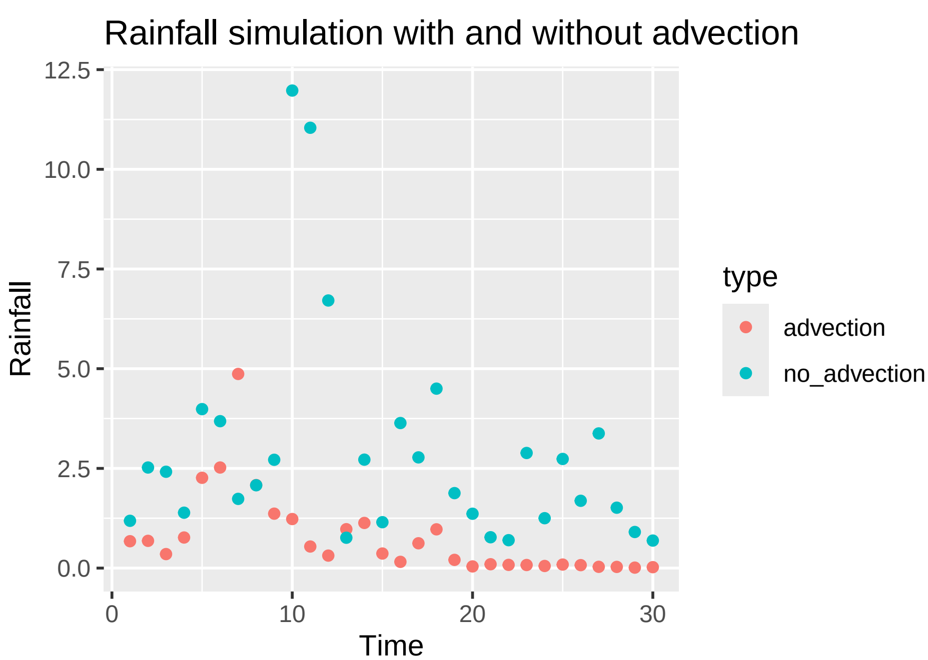
```r
  for (i in sz)
    for (j in seq_len(Nxy))
      grid[j + Nxy * (i - 1), 3] <- i

  # Construct shifted grid with advected coordinates
  grid[, 1:2] <- grid[, 1:2] - grid[, 3] * adv

  ## Construct shifted variogram
  Varm1 <- vapply(seq_len(N), function(n)
      RandomFields::RFvariogram(modelBuhlCklu,
        x = sx - grid[n, 1],
        y = sy - grid[n, 2],
        z = sz - grid[n, 3]),
        array(NA_real_, dim = c(lx, ly, lz))) ## => (lx, ly, lz, N)-array

  ## Main
  set.seed(123)
  Z <- array(, dim = c(lx, ly, lz, n.BR)) # 4d array
  E <- matrix(rexp(n.BR * N), nrow = n.BR, ncol = N)
  for (i in seq_len(n.BR)) { ## n=1
    V <- 1 / E[i, 1]
    W <- RandomFields::RFsimulate(modelBuhlCklu, x, y, z, n = 1)
    Y <- exp(W - W[1] - Varm1[, , , 1])
    Z[, , , i] <- V * Y
    ## n in {2,..,N}
    for (n in 2:N) {
      Exp <- E[i, n]
      V <- 1 / Exp
      while(V > Z[N * (i - 1) + n]) {
        W <- RandomFields::RFsimulate(modelBuhlCklu, x, y, z)
        Y <- exp(W - W[n] - Varm1[, , , n])
        if(all(V * Y[seq_len(n-1)] < Z[(N*(i-1)+1):(N*(i-1)+(n-1))]))
          Z[, , , i] <- pmax(V * Y, Z[, , , i])
          Exp <- Exp + rexp(1)
          V <- 1 / Exp
      }
    }
  }
  ## Return
  Z
}
```

Simulations verifications

Comparaison avec et sans advection (seed fixée):

```r
adv <- c(0.5, 0.5)
true_param <- c(0.4, 0.2, 1.5, 1, adv)
ngrid <- 2
spa <- 1:ngrid
nsites <- ngrid^2 # if the grid is squared
temp <- 1:30
```

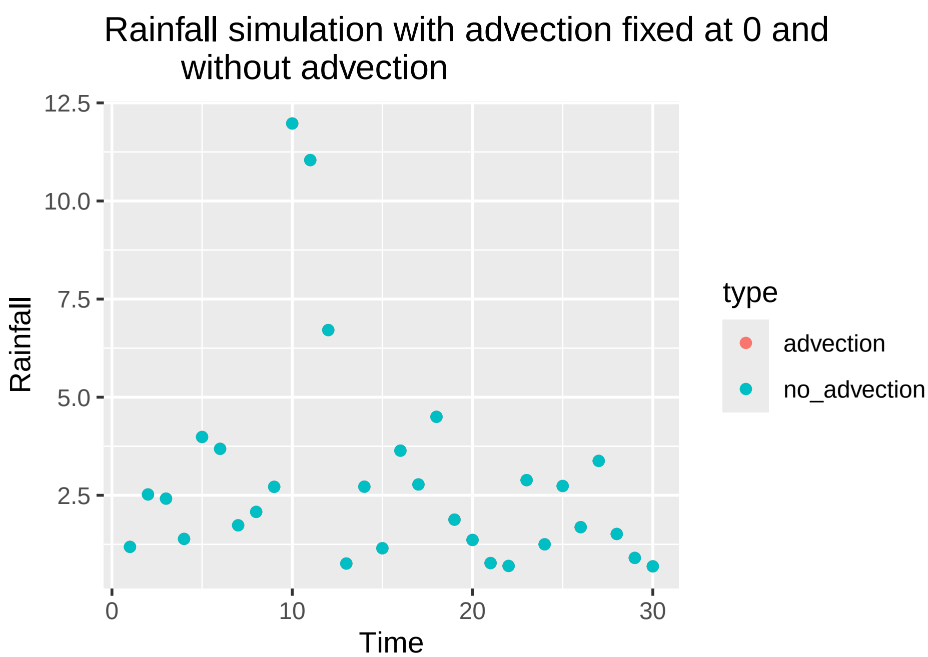
```r
n.BR <- 1

# generate the simulations
BR_adv <- sim_BR_adv(true_param[1] * 2, true_param[2] * 2, true_param[3],
                    true_param[4], spa, spa, temp, n.BR, adv)
# plot(BR_adv[1, 1, , ], main = "Rainfall simulation with advection")

BR_noadv <- sim_BR(true_param[1] * 2, true_param[2] * 2, true_param[3],
                  true_param[4], spa, spa, temp, n.BR)
# plot(BR_noadv[1, 1, , ], main = "Rainfall simulation without advection")

# plot on the same graph with ggplot
df_adv <- data.frame(time = temp,
                     value = BR_adv[1, 1, , ],
                     type = "advection")
df_noadv <- data.frame(time = temp,
                       value = BR_noadv[1, 1, , ],
                       type = "no_advection")

df_plot <- rbind(df_adv, df_noadv)

ggplot(df_plot, aes(x = time, y = value, color = type)) +
  geom_point() +
  labs(title = "Rainfall simulation with and without advection",
       x = "Time", y = "Rainfall")
```



Comparaison sans advection avec les différents codes, on retrouve bien la meme chose, sachant que l'on fixe la seed:

```r
adv <- c(0, 0)
true_param <- c(0.4, 0.2, 1.5, 1, adv)
ngrid <- 2
```

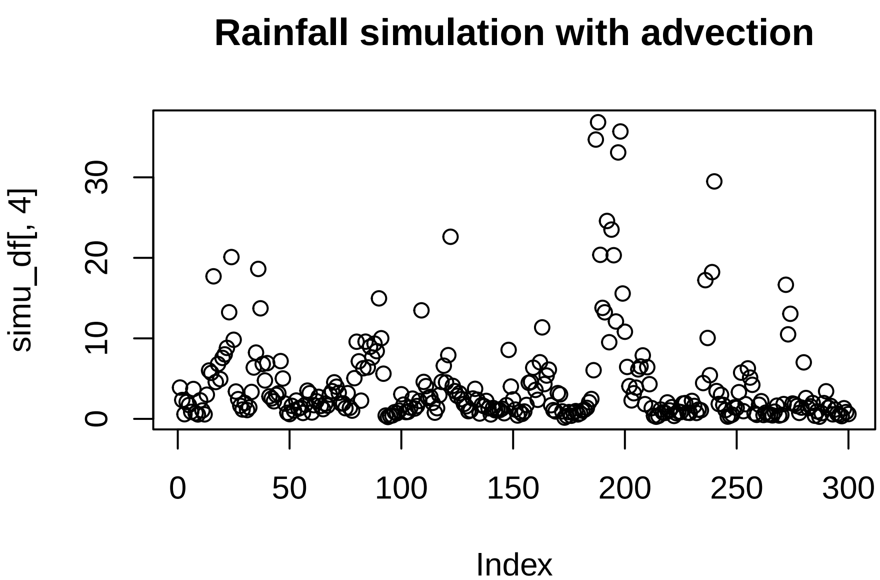
```r
spa <- 1:ngrid
nsites <- ngrid^2 # if the grid is squared
temp <- 1:30
n.BR <- 1

# generate the simulations
BR_adv <- sim_BR_adv(true_param[1] * 2, true_param[2] * 2, true_param[3],
                    true_param[4], spa, spa, temp, n.BR, adv)

BR_noadv <- sim_BR(true_param[1] * 2, true_param[2] * 2, true_param[3],
                  true_param[4], spa, spa, temp, n.BR)

# plot on the same graph with ggplot
df_adv <- data.frame(time = temp,
                     value = BR_adv[1, 1, , ],
                     type = "advection")
df_noadv <- data.frame(time = temp,
                      value = BR_noadv[1, 1, , ],
                      type = "no_advection")

df_plot <- rbind(df_adv, df_noadv)

ggplot(df_plot, aes(x = time, y = value, color = type)) +
  geom_point() +
  labs(title = "Rainfall simulation with advection fixed at 0 and
        without advection",
       x = "Time", y = "Rainfall")
```




```r
adv <- c(0.5, 0.5)
true_param <- c(0.4, 0.2, 1.5, 1, adv)
ngrid <- 5
spa <- 1:ngrid
```

```r
nsites <- ngrid^2 # if the grid is squared
temp <- 1:300
n.BR <- 1

# BR <- sim_BR_adv(true_param[1] * 2, true_param[2] * 2, true_param[3],
#                        true_param[4], spa, spa, temp, n.BR)


# save_simulations(BR, ngrid, n.BR,
#         folder = paste0("../data/simulations_BR/oldsim_adv_", ngrid^2, "s_",
#                                 length(temp), "t/"),
#                   file = paste0("br_", ngrid^2, "s_",
#                                 length(temp), "t"), forcedind = 1)

# load the simulations
file_path <- paste0("../data/simulations_BR/oldsim_adv_", ngrid^2, "s_",
                              length(temp), "t/br_",
                        ngrid^2, "s_", length(temp), "t_", 1, ".csv")
simu_df <- read.csv(file_path)
```

```r
plot(simu_df[, 4], main = "Rainfall simulation with advection")
```




```r
nsites <- ncol(simu_df)
sites_coords <- generate_grid_coords(sqrt(nsites))
dist_mat <- get_dist_mat(sites_coords,
                         latlon = FALSE) # distance matrix
df_dist <- reshape_distances(dist_mat) # reshape the distance matrix
```

```r
sites_coords <- generate_grid_coords(sqrt(nsites))
df_lags <- get_lag_vectors(sites_coords, true_param,
                          hmax = sqrt(17), tau_vect = 0:10)

hmax <- sqrt(17)
```

```r
q <- 0.9

# excesses <- empirical_excesses(simu_df, quantile = q, df_lags = df_lags)

# result <- optim(par = c(true_param), fn = neg_ll,
#                          simu = simu_df,
#                          quantile = q,
#                          df_lags = df_lags,
#                          locations = sites_coords,
#                          method = "CG",
#                          control = list(parscale = c(1, 1, 0.1, 0.1, 1, 1),
#                                              maxit = 10000))

# print(result$convergence) # 0 if it has converged
# print(result$par)

# if (result$convergence == 0) {
#   rmse <- sqrt((result$par - true_param)^2)
#   df_rmse <- data.frame(estim = result$par, rmse = rmse)
#   rownames(df_rmse) <- c("beta1", "beta2", "alpha1", "alpha2", "Vx", "Vy")
#   # print(t(df_rmse))
#   # save the results
#   write.csv(t(df_rmse), file = "../data/optim_results_95.csv")
# } else {
#   print("No convergence")
# }

# get result from csv
df_rmse <- read.csv("../data/optim_results_80.csv")
print(df_rmse)
```

```
##       X     beta1      beta2      alpha1      alpha2         Vx           Vy
## 1 estim 0.2022172 0.12002196 1.497439422 0.998345525 0.06532307 0.0007833163
## 2  rmse 0.1977828 0.07997804 0.002560578 0.001654475 0.43467693 0.4992166837
```

```r
df_rmse <- read.csv("../data/optim_results_85.csv")
print(df_rmse)
```

```
##       X     beta1      beta2      alpha1      alpha2         Vx           Vy
## 1 estim 0.2519524 0.14466452 1.495955886 0.997330498 0.04565102 0.0002049355
## 2  rmse 0.1480476 0.05533548 0.004044114 0.002669502 0.45434898 0.4997950645
```

```r
df_rmse <- read.csv("../data/optim_results_90.csv")
print(df_rmse)
```

```
##       X      beta1      beta2     alpha1     alpha2         Vx           Vy
## 1 estim 0.33673233 0.17390599 1.46478128 0.97766504 0.04402112 0.0003720267
## 2  rmse 0.06326767 0.02609401 0.03521872 0.02233496 0.45597888 0.4996279733
```

Plus petite advection

```
adv <- c(0.05, 0.05) # pixel by time?
true_param <- c(0.4, 0.2, 1.5, 1, adv)
ngrid <- 5
spa <- 1:ngrid
nsites <- ngrid^2 # if the grid is squared
temp <- 1:300
n.BR <- 1

# BR <- sim_BR_adv(true_param[1] * 2, true_param[2] * 2, true_param[3],
#                     true_param[4], spa, spa, temp, n.BR)


# save_simulations(BR, ngrid, n.BR,
#         folder = paste0("../data/simulations_BR/oldsim_adv_2_", ngrid^2, "s_",
#                              length(temp), "t/"),
#                   file = paste0("br_", ngrid^2, "s_",
#                              length(temp), "t"), forcedind = 1)

# load the simulations
file_path <- paste0("../data/simulations_BR/oldsim_adv_2_", ngrid^2, "s_",
                           length(temp), "t/br_",
                      ngrid^2, "s_", length(temp), "t_", 1, ".csv")
simu_df <- read.csv(file_path)
```

```
nsites <- ncol(simu_df)
sites_coords <- generate_grid_coords(sqrt(nsites))
dist_mat <- get_dist_mat(sites_coords,
                         latlon = FALSE) # distance matrix
df_dist <- reshape_distances(dist_mat) # reshape the distance matrix
```

```
sites_coords <- generate_grid_coords(sqrt(nsites))
df_lags <- get_lag_vectors(sites_coords, true_param,
                          hmax = sqrt(17), tau_vect = 0:10)

hmax <- sqrt(17)
q <- 0.85

# excesses <- empirical_excesses(simu_df, quantile = q, df_lags = df_lags)

# result <- optim(par = c(true_param), fn = neg_ll,
#                          simu = simu_df,
#                          quantile = q,
#                          df_lags = df_lags,
#                          locations = sites_coords,
#                          method = "CG",
#                          control = list(parscale = c(1, 1, 0.1, 0.1, 1, 1),
#                                        maxit = 10000))

# print(result$convergence) # 0 if it has converged
# print(result$par)
```

```r
# if (result$convergence == 0) {
#   rmse <- sqrt((result$par - true_param)^2)
#   df_rmse <- data.frame(estim = result$par, rmse = rmse)
#   rownames(df_rmse) <- c("beta1", "beta2", "alpha1", "alpha2", "Vx", "Vy")
#   # print(t(df_rmse))
#   # save the results
#   write.csv(t(df_rmse), file = "../data/optim_results_2_85.csv")
# } else {
#   print("No convergence")
# }

# get result from csv
df_rmse <- read.csv("../data/optim_results_2_85.csv")
print(df_rmse)
```

```
##       X     beta1      beta2       alpha1       alpha2          Vx           Vy
## 1 estim 0.2794549 0.14334178 1.4994688066 0.9996596452 0.055497791 0.0005206783
## 2  rmse 0.1205451 0.05665822 0.0005311934 0.0003403548 0.005497791 0.0494793217
```