Simulations

2024-09-09

Simulation Brown-Resnick

Sans l'advection

```r
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
}
```

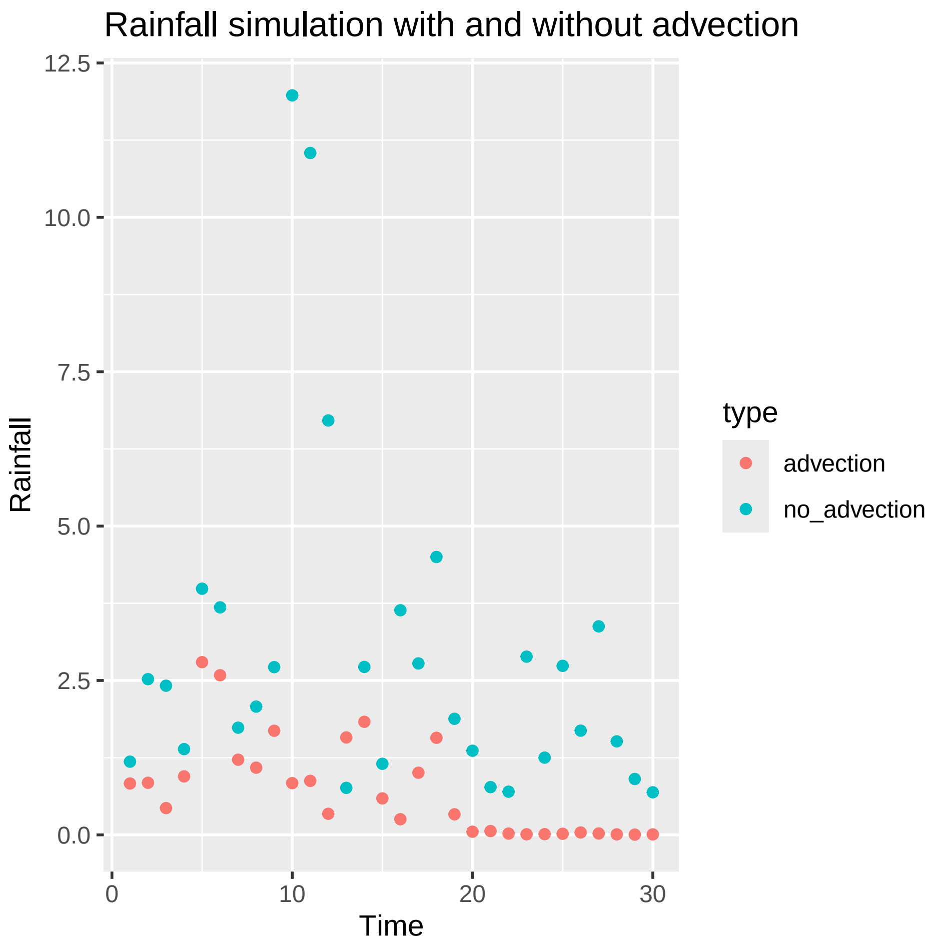
Avec l'advection

```
sim_BR_adv <- function(beta1, beta2, alpha1, alpha2, x, y, z, n.BR,
                       adv) {
  ## 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)
```

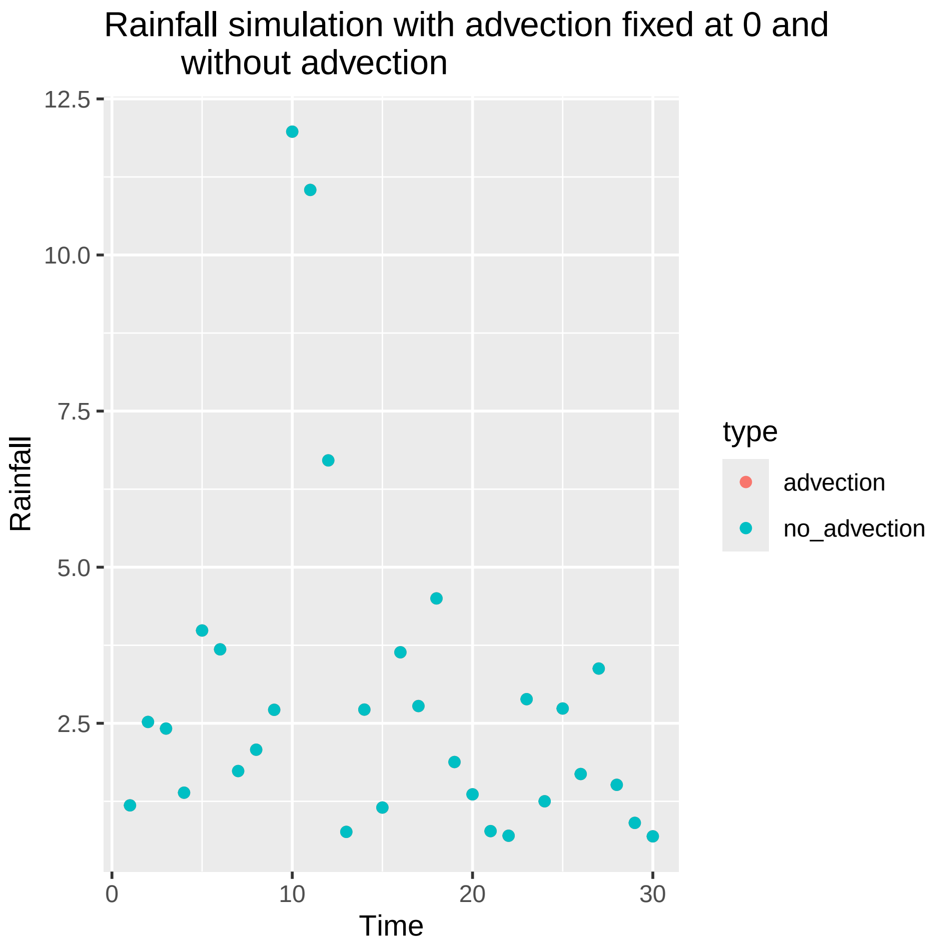
```
    for (j in sx)
      for (k in sz)
        grid[j + lx * (i - 1) + (k - 1) * Nxy, 2] <- i

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

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

  ## 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):



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



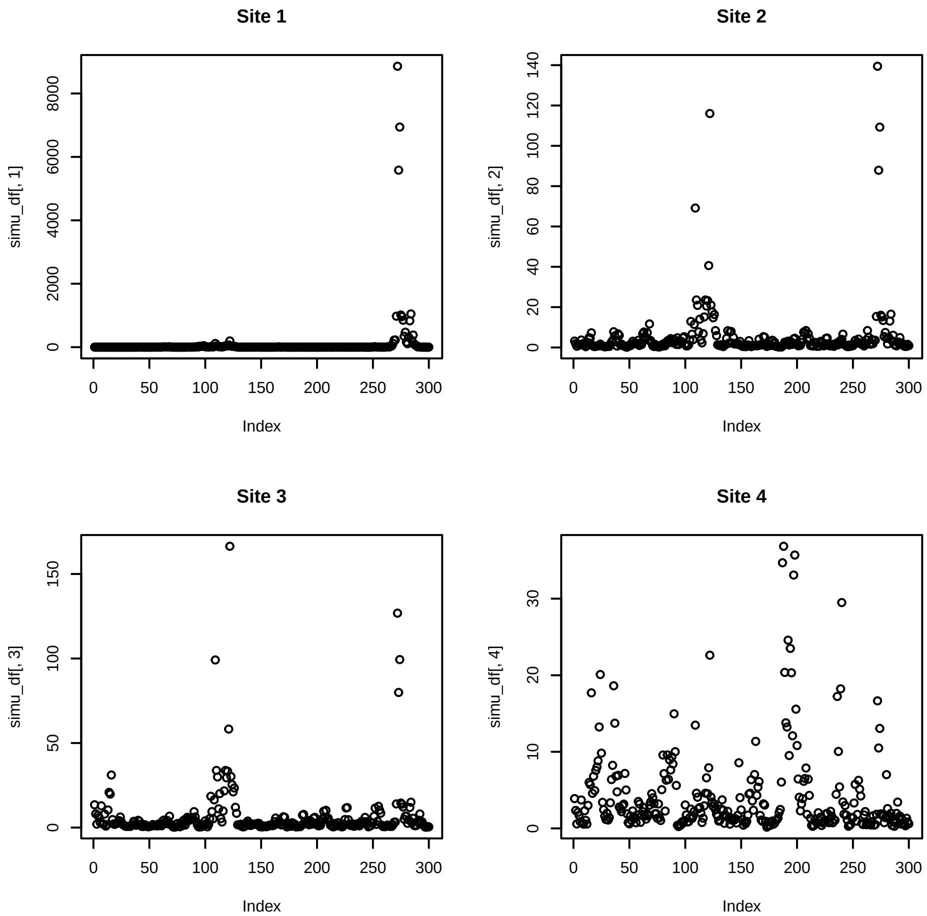
Donc je vais utiliser `sim_BR_adv` pour les simulations.

Simulation avec 25 sites et 300 pas de temps sans advection

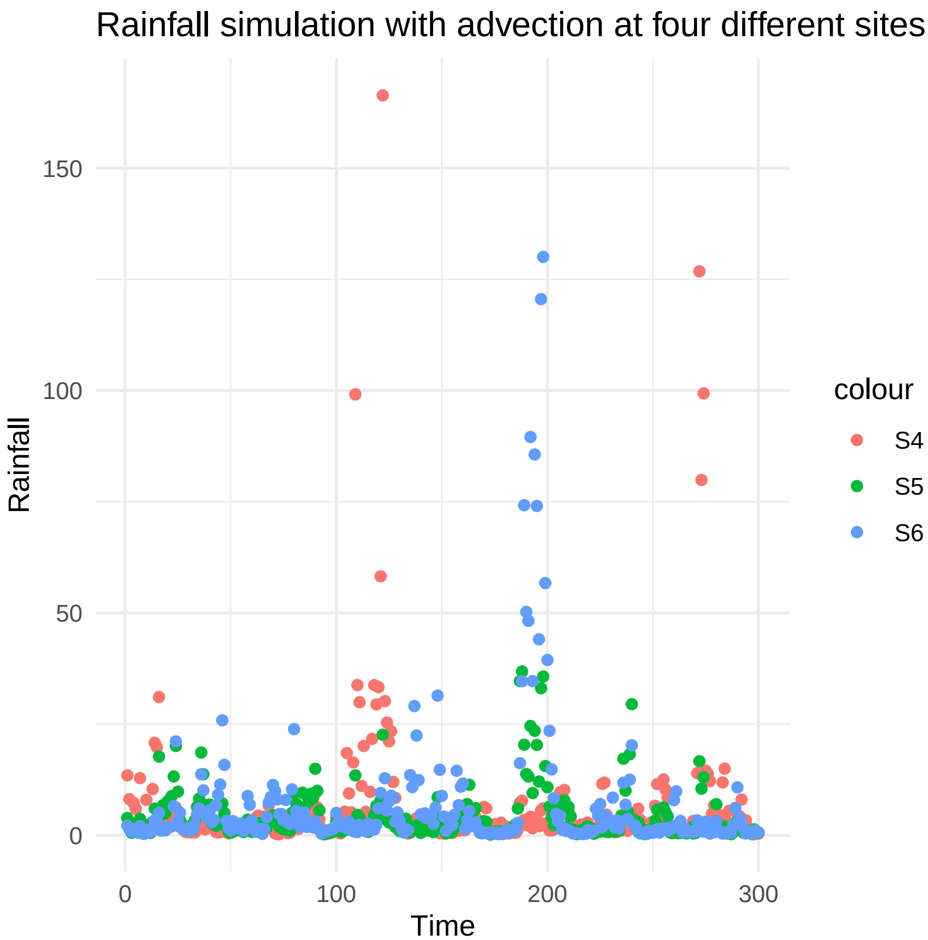
```
adv <- c(0, 0)
true_param <- c(0.4, 0.2, 1.5, 1, adv)
ngrid <- 5
spa <- 1:ngrid
nsites <- ngrid^2 # if the grid is squared
temp <- 1:300
n.BR <- 1

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

foldername <- paste0("../data/simulations_BR/sim_", ngrid^2, "s_",
                                length(temp), "t/")
if (!dir.exists(foldername)) {
  dir.create(foldername, recursive = TRUE)
}

save_simulations(BR, ngrid, n.BR, folder = foldername,
```

```
       file = paste0("br_", ngrid^2, "s_", length(temp), "t"), forcedind = 1)

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

```
# simulation gif
# create_simu_gif(simu_df, true_param, type = "br", forcedtemp = 50)
```




```
# plot the simulation at four different sites on 100 time steps
df_plot <- data.frame(time = 1:300,
                      S3 = simu_df[, 3],
                      S4 = simu_df[, 4],
                      S5 = simu_df[, 5])

ggplot(df_plot, aes(x = time)) +
    geom_point(aes(y = S3, color = "S4")) +
    geom_point(aes(y = S4, color = "S5")) +
    geom_point(aes(y = S5, color = "S6")) +
    labs(title = "Rainfall simulation with advection at four different sites",
         x = "Time", y = "Rainfall") +
    theme_minimal()
```



Verif marginales

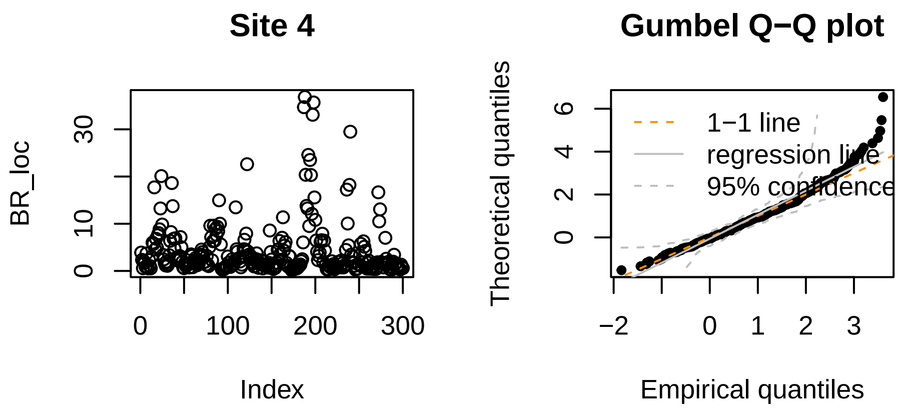
```
# qq-plots margins
par(mfrow = c(2, 2))

BR_loc <- simu_df$S4
plot(BR_loc, main = "Site 4")
BR_loc_log <- log(BR_loc)
gumbel.fit <- gum.fit(BR_loc_log)
```

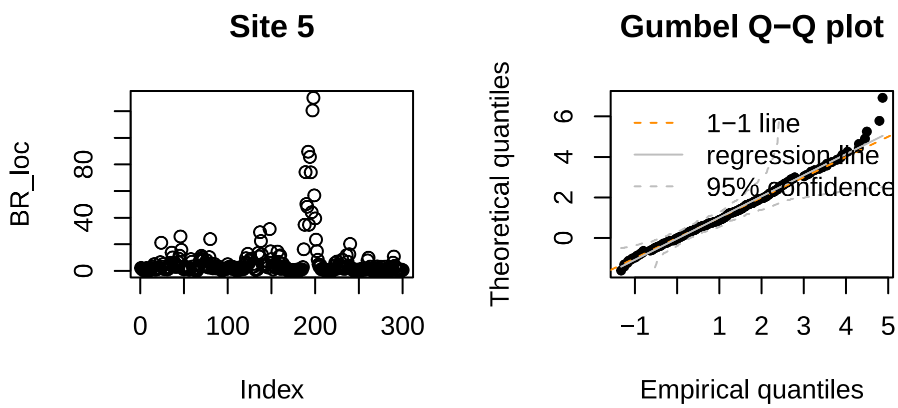
```
## $conv
## [1] 0
##
## $nllh
## [1] 456.0032
##
## $mle
## [1] 0.2863054 0.9784276
##
## $se
## [1] 0.05975817 0.04252630
```

```r
mu <- gumbel.fit$mle[1]
sigma <- gumbel.fit$mle[2]

theorical_qgum <- qgumbel(ppoints(BR_loc_log), mu, sigma)

qqplot(BR_loc_log, theorical_qgum, main = "Gumbel Q-Q plot",
       xlab = "Empirical quantiles",
       ylab = "Theoretical quantiles")


BR_loc <- simu_df$S5
plot(BR_loc, main = "Site 5")
BR_loc_log <- log(BR_loc)
gumbel.fit <- gum.fit(BR_loc_log)
```

```
## $conv
## [1] 0
##
## $nllh
## [1] 480.268
##
## $mle
## [1] 0.3115968 1.0328937
##
## $se
## [1] 0.06291003 0.04645451
```

```r
mu <- gumbel.fit$mle[1]
sigma <- gumbel.fit$mle[2]

theorical_qgum <- qgumbel(ppoints(BR_loc_log), mu, sigma)

qqplot(BR_loc_log, theorical_qgum, main = "Gumbel Q-Q plot",
       xlab = "Empirical quantiles",
       ylab = "Theoretical quantiles")
```





Simulation avec 25 sites et 100 pas de temps avec advection

```
adv <- c(0.05, 0.02)
true_param <- c(0.1, 0.05, 1.5, 1, adv)
ngrid <- 5
spa <- 1:ngrid
nsites <- ngrid^2 # if the grid is squared
temp <- 1:100
n.BR <- 1

start_time <- Sys.time()
BR <- sim_BR_adv(true_param[1] * 2, true_param[2] * 2, true_param[3],
                 true_param[4], spa, spa, temp, n.BR, adv)
end_time <- Sys.time()
print(end_time - start_time)
```

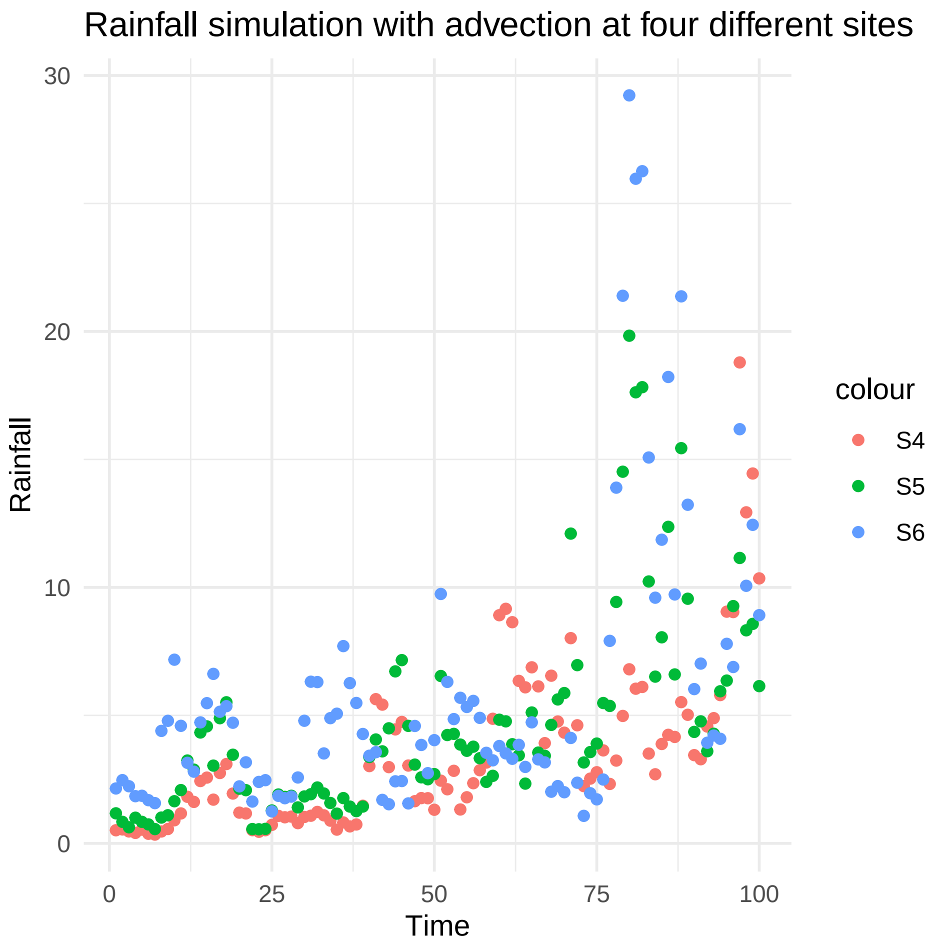
```
## Time difference of 5.5749 secs
```

```
foldername <- paste0("../data/simulations_BR/sim_adv_", ngrid^2, "s_",
                                length(temp), "t/")
if (!dir.exists(foldername)) {
  dir.create(foldername, recursive = TRUE)
}
```

```r
save_simulations(BR, ngrid, n.BR, folder = foldername,
        file = paste0("br_", ngrid^2, "s_", length(temp), "t"), forcedind = 1)

# load the simulations
file_path <- paste0(foldername, "br_", ngrid^2, "s_", length(temp),
                "t_", 1, ".csv")
simu_df_adv <- read.csv(file_path)
```

```r
# simulation gif
# create_simu_gif(simu_df_adv, true_param, type = "br_adv", forcedtemp = 50)
```

```r
# plot the simulation at four different sites on 100 time steps
df_plot <- data.frame(time = 1:100,
                      S3 = simu_df_adv[, 3],
                      S4 = simu_df_adv[, 4],
                      S5 = simu_df_adv[, 5])

ggplot(df_plot, aes(x = time)) +
    geom_point(aes(y = S3, color = "S4")) +
    geom_point(aes(y = S4, color = "S5")) +
    geom_point(aes(y = S5, color = "S6")) +
    labs(title = "Rainfall simulation with advection at four different sites",
         x = "Time", y = "Rainfall") +
    theme_minimal()
```



Verif marginales

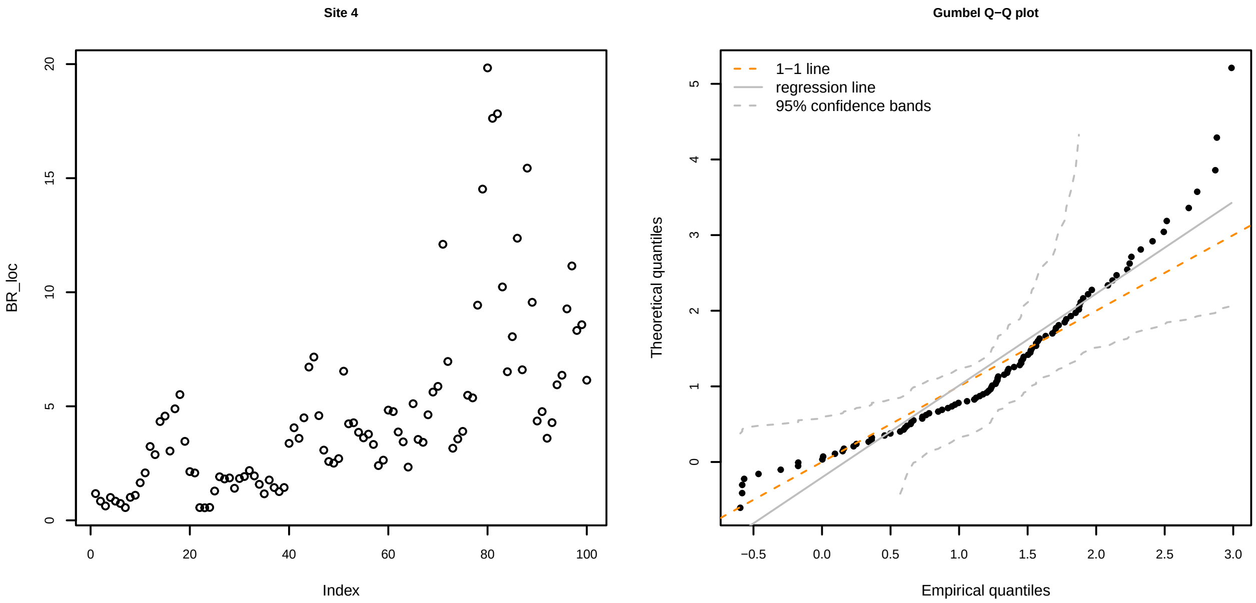
```
# qq-plots margins
par(mfrow = c(2, 2), cex = 0.5, cex.main = 0.8, cex.axis = 0.8)

BR_loc <- simu_df_adv$S4
plot(BR_loc, main = "Site 4")
BR_loc_log <- log(BR_loc)
gumbel.fit <- gum.fit(BR_loc_log)
```

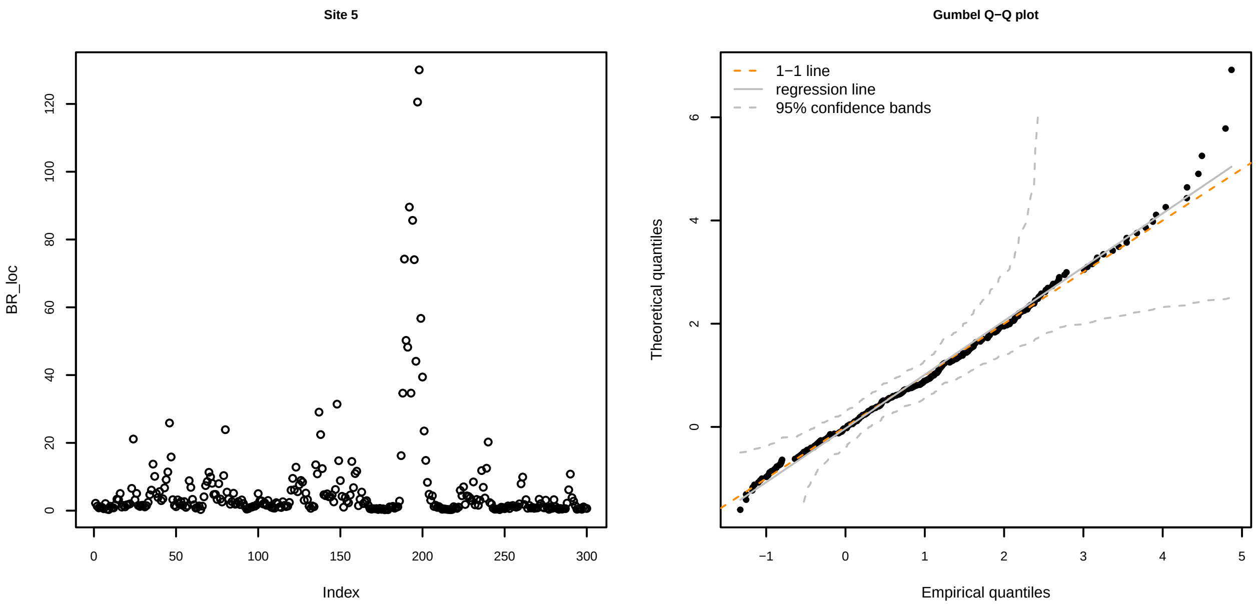
```
## $conv
## [1] 0
##
## $nllh
## [1] 132.6661
##
## $mle
## [1] 0.7874979 0.8352338
##
## $se
## [1] 0.08870781 0.06056060
```

```r
mu <- gumbel.fit$mle[1]
sigma <- gumbel.fit$mle[2]

theorical_qgum <- qgumbel(ppoints(BR_loc_log), mu, sigma)

qqplot(BR_loc_log, theorical_qgum, main = "Gumbel Q-Q plot",
       xlab = "Empirical quantiles",
       ylab = "Theoretical quantiles")


BR_loc <- simu_df$S5
plot(BR_loc, main = "Site 5")
BR_loc_log <- log(BR_loc)
gumbel.fit <- gum.fit(BR_loc_log)
```

```
## $conv
## [1] 0
##
## $nllh
## [1] 480.268
##
## $mle
## [1] 0.3115968 1.0328937
##
## $se
## [1] 0.06291003 0.04645451
```

```r
mu <- gumbel.fit$mle[1]
sigma <- gumbel.fit$mle[2]

theorical_qgum <- qgumbel(ppoints(BR_loc_log), mu, sigma)

qqplot(BR_loc_log, theorical_qgum, main = "Gumbel Q-Q plot",
       xlab = "Empirical quantiles",
       ylab = "Theoretical quantiles")
```

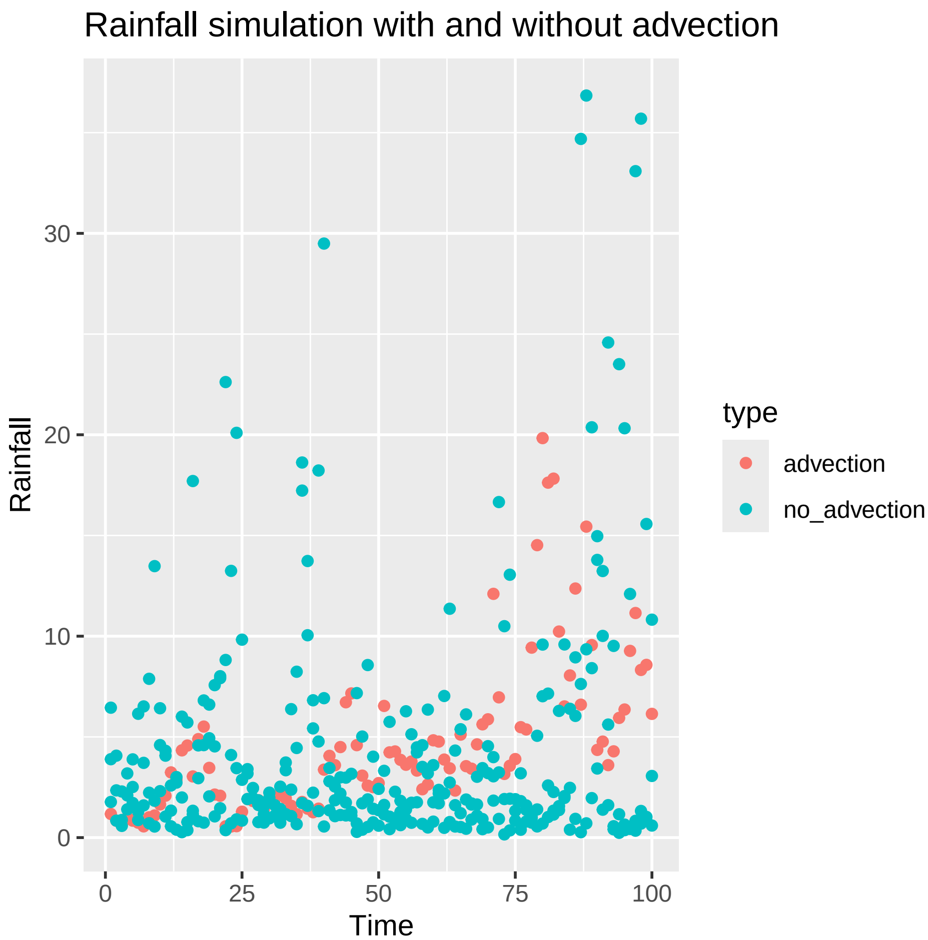




Comparaison avec et sans advection

```
# plot on the same graph with ggplot
df_adv <- data.frame(time = temp,
                     value = simu_df_adv[, 4],
                     type = "advection")
df_noadv <- data.frame(time = temp,
                      value = simu_df[, 4],
                      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")
```



Probleme simulation avec advection

Quand le nombre de pas de temps est trop élevé, cela prends beaucoup trop de temps ie le code a tourner 7 jours pour 300 pas de temps et 25 sites et je l'ai arreté.

Est ce que c'est parce que la fonction ne passer pas bien avec la parallélisation??

Avec 200 pas de temps, cela fonctionne:

```
adv <- c(0.05, 0.02)
true_param <- c(0.1, 0.05, 1.5, 1, adv)
ngrid <- 5
spa <- 1:ngrid
nsites <- ngrid^2 # if the grid is squared
temp <- 1:200
n.BR <- 1
```

```r
start_time <- Sys.time()
BR <- sim_BR_adv(true_param[1] * 2, true_param[2] * 2, true_param[3],
                    true_param[4], spa, spa, temp, n.BR, adv)
end_time <- Sys.time()
print(end_time - start_time)
```

```
## Time difference of 11.69549 secs
```

```r
foldername <- paste0("../data/simulations_BR/sim_adv_", ngrid^2, "s_",
                                length(temp), "t/")
if (!dir.exists(foldername)) {
  dir.create(foldername, recursive = TRUE)
}

save_simulations(BR, ngrid, n.BR, folder = foldername,
        file = paste0("br_", ngrid^2, "s_", length(temp), "t"), forcedind = 1)

# load the simulations
file_path <- paste0(foldername, "br_", ngrid^2, "s_", length(temp),
                "t_", 1, ".csv")
simu_df_adv <- read.csv(file_path)
```

Avec 300 pas de temps et une faible advection, cela fonctionne:

```r
adv <- c(0.05, 0.02)
true_param <- c(0.1, 0.05, 1.5, 1, adv)
ngrid <- 5
spa <- 1:ngrid
nsites <- ngrid^2 # if the grid is squared
temp <- 1:300
n.BR <- 1

start_time <- Sys.time()
BR <- sim_BR_adv(true_param[1] * 2, true_param[2] * 2, true_param[3],
                    true_param[4], spa, spa, temp, n.BR, adv)
end_time <- Sys.time()
print(end_time - start_time)
```

```
## Time difference of 2.449328 mins
```

```r
foldername <- paste0("../data/simulations_BR/sim_adv2_", ngrid^2, "s_",
                                length(temp), "t/")
if (!dir.exists(foldername)) {
  dir.create(foldername, recursive = TRUE)
}

save_simulations(BR, ngrid, n.BR, folder = foldername,
        file = paste0("br_", ngrid^2, "s_", length(temp), "t"), forcedind = 1)

# load the simulations
file_path <- paste0(foldername, "br_", ngrid^2, "s_", length(temp),
                "t_", 1, ".csv")
simu_df_adv <- read.csv(file_path)
```

Avec 300 pas de temps et une plus forte advection, c'est ultra long car les valeurs des coordonnées explosent et cela explose dans le variogramme qui va écrasé les valeurs de la simulation dans le $Y = exp(W - W[1] - Varm1[,,,1])$ qui va donner des valeurs très proches de 0 et donc $Z = V * Y$ va donner des valeurs très proches de 0 et cela ne va pas trouver de maximum.

```
adv <- c(0.05, 0.02)
true_param <- c(0.1, 0.05, 1.5, 1, adv)
ngrid <- 5
spa <- 1:ngrid
nsites <- ngrid^2 # if the grid is squared
temp <- 1:300
n.BR <- 1

start_time <- Sys.time()
BR <- sim_BR_adv(true_param[1] * 2, true_param[2] * 2, true_param[3],
                    true_param[4], spa, spa, temp, n.BR, adv)
end_time <- Sys.time()
print(end_time - start_time)
```

```
## Time difference of 2.51908 mins
```

```
foldername <- paste0("../data/simulations_BR/sim_adv2_", ngrid^2, "s_",
                                length(temp), "t/")
if (!dir.exists(foldername)) {
  dir.create(foldername, recursive = TRUE)
}

save_simulations(BR, ngrid, n.BR, folder = foldername,
        file = paste0("br_", ngrid^2, "s_", length(temp), "t"), forcedind = 1)

# load the simulations
file_path <- paste0(foldername, "br_", ngrid^2, "s_", length(temp),
                "t_", 1, ".csv")
simu_df_adv <- read.csv(file_path)
```