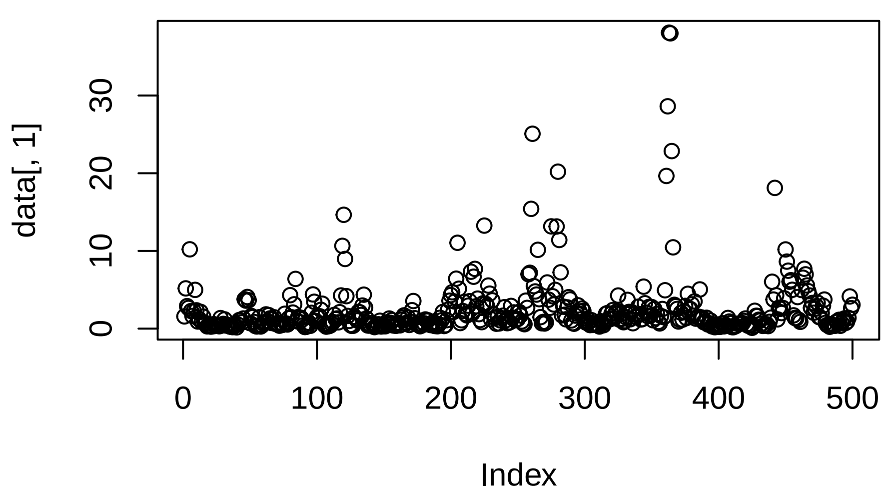
Optimisation

2024-08-15

Data from simulations

Dans un premier temps, j'ai pris des simulations de Brown-Resnick avec moins de sites mais plus de pas de temps. J'ai pris des simulations avec 4 sites et 500 pas de temps.

```
# spatial and temporal structures
ngrid <- 2
spa <- 1:ngrid
nsites <- ngrid^2 # if the grid is squared
temp <- 1:500

# Simulation
num_iterations <- 5
list_BR <- list()
for (i in 1:num_iterations) {
  file_path <- paste0("../data/simulations_BR/sim_", ngrid^2, "s_",
                      length(temp), "t/br_", ngrid^2, "s_", length(temp), "t_",
                      i, ".csv")
  df <- read.csv(file_path)
  list_BR[[i]] <- df
}

simu_df <- list_BR[[1]] # first simulation
nsites <- ncol(simu_df) # number of sites
data <- simu_df
plot(data[, 1])
```



```r
# get grid coordinates
sites_coords <- generate_grid_coords(sqrt(nsites))
# get distance matrix
distance_matrix <- as.matrix(dist(sites_coords))
rownames(distance_matrix) <- colnames(distance_matrix) <- paste0("S",
                                                           1:nrow(sites_coords))

print(distance_matrix)
```

```
##          S1       S2       S3       S4
## S1 0.000000 1.000000 1.000000 1.414214
## S2 1.000000 0.000000 1.414214 1.000000
## S3 1.000000 1.414214 0.000000 1.000000
## S4 1.414214 1.000000 1.000000 0.000000
```

```r
# Function to calculate excess indicators for each site
excesses_indicators <- function(data, q) {
  # Assume each row represents a moment in time
  data$time <- 1:nrow(data)

  # Convert data to long format
  data_long <- melt(data, id.vars = "time", variable.name = "site",
                    value.name = "value")
  colnames(data_long) <- c("time", "site", "value")

  # Calculate threshold for each site
  thresholds <- aggregate(value ~ site, data_long, function(x) quantile(x, q))

  # Add a column for excesses
  data_long <- merge(data_long, thresholds, by = "site",
                      suffixes = c("", "_threshold"))
  data_long$excess <- ifelse(data_long$value > data_long$value_threshold, 1, 0)

  return(data_long)
}

q <- 0.7 # quantile
data_long <- excesses_indicators(data, q)
print(head(data_long))
```

```
##   site time     value value_threshold excess
## 1   S1    1  1.572784        2.151103      0
## 2   S1    2  5.180775        2.151103      1
## 3   S1    3  2.897169        2.151103      1
## 4   S1    4  2.662405        2.151103      1
## 5   S1    5 10.207054        2.151103      1
## 6   S1    6  2.266449        2.151103      1
```

Calculate joint excesses

$k_{ij,\tau} = \sum_t 1_{\{}X_{s_i,t} > q, X_{s_j,t+\tau} > q\}$

```r
# Function to calculate k_ij,tau for a pair of sites
calculate_kij_tau <- function(data, site_i, site_j, tau) {
  data_i <- data[data$site == site_i, ]
  data_j <- data[data$site == site_j, ]

  if (tau >= 0) {
    data_i_n <- data_i[1:(nrow(data_i) - tau), ]
    data_j_n <- data_j[(tau + 1):nrow(data_j), ]
  } else {
    data_i_n <- data_i[(-tau + 1):nrow(data_i), ]
    data_j_n <- data_j[1:(nrow(data_j) + tau), ]
  }

  sum(data_i_n$excess & data_j_n$excess)
}

site_i <- "S1"
site_j <- "S4"
tau <- 10

kij_tau <- calculate_kij_tau(data_long, site_i, site_j, tau)
print(kij_tau)
```

```
## [1] 52
```

```r
# verification
data_s1_lag <- data$S1[1:(nrow(data) - tau)]
data_s4_lag <- data$S4[(1 + tau):(nrow(data))]
kij_tau_verif <- sum(data_s1_lag > quantile(data$S1, q)
          & data_s4_lag > quantile(data$S4, q))
print(kij_tau_verif)
```

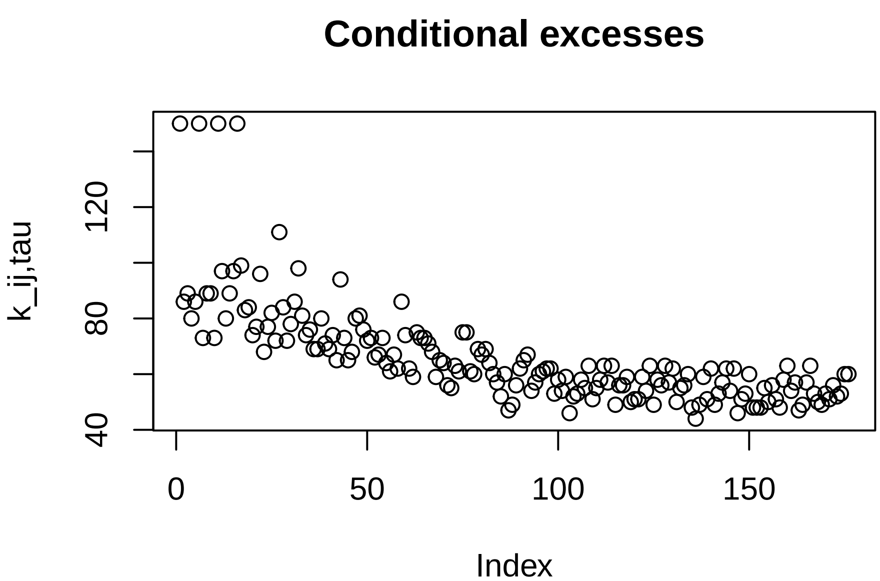
```
## [1] 52
```

```r
# Function to calculate k_ij,tau for different tau values
calculate_kij_tau_list <- function(data_long, taus) {
  kij_tau_list <- list()
  sites <- unique(data_long$site)

  for (tau in taus) {
    # Initialize matrix
    kij_matrix <- matrix(0, nrow = length(sites), ncol = length(sites),
                         dimnames = list(sites, sites))
    for (i in 1:length(sites)) { # for each pair of sites
      for (j in 1:length(sites)) {
        # if (i != j) {
          kij_matrix[i, j] <- calculate_kij_tau(data_long, sites[i], sites[j],
                                                 tau)
        # }
      }
    }
    kij_tau_list[[as.character(tau)]] <- kij_matrix # for each tau
```

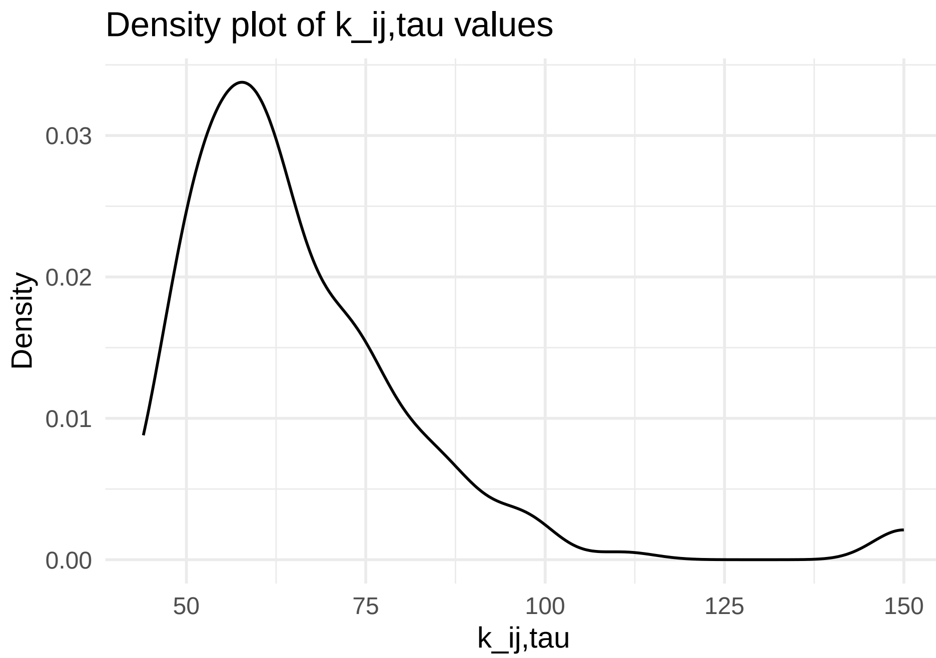
```
  }

  return(kij_tau_list)
}

taus <- 0:10 # temporal lags
kij_tau_list <- calculate_kij_tau_list(data_long, taus) # conditional excesses
print(kij_tau_list$`0`) # no temporal lag
```

```
##      S1  S2  S3  S4
## S1  150  86  89  80
## S2   86 150  73  89
## S3   89  73 150  97
## S4   80  89  97 150
```

```
# get all values in kij_tau_list
all_values <- unlist(kij_tau_list)
# remove 0
all_values <- all_values[all_values != 0]
plot(all_values, main = "Conditional excesses",
                ylab = "k_ij,tau")
```




```
# density plot
ggplot(data.frame(x = all_values), aes(x = x)) +
  geom_density() +
  labs(title = "Density plot of k_ij,tau values",
       x = "k_ij,tau",
       y = "Density") +
  theme_minimal()
```



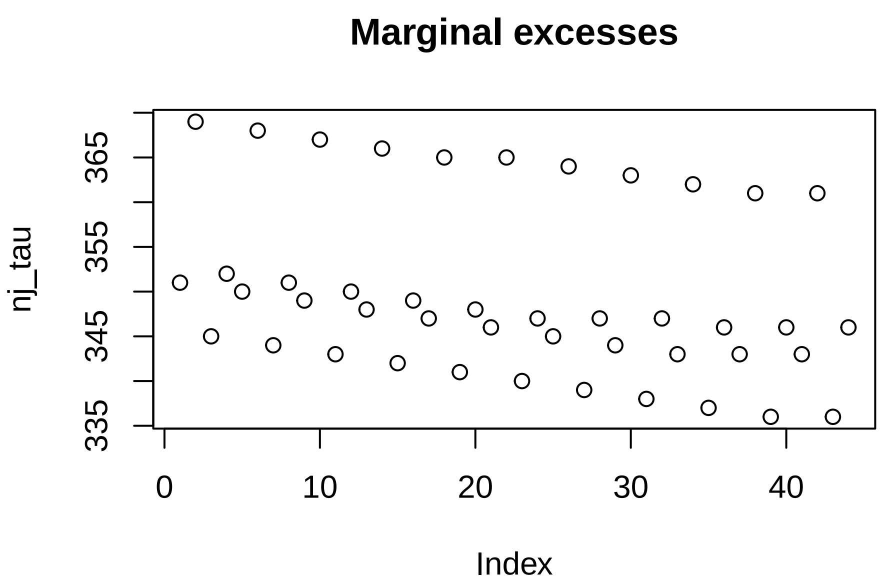
Calculate marginal excesses

$n_{j,\tau} = \sum_t 1_{\{X_{s_j,t+\tau}\}>q}$

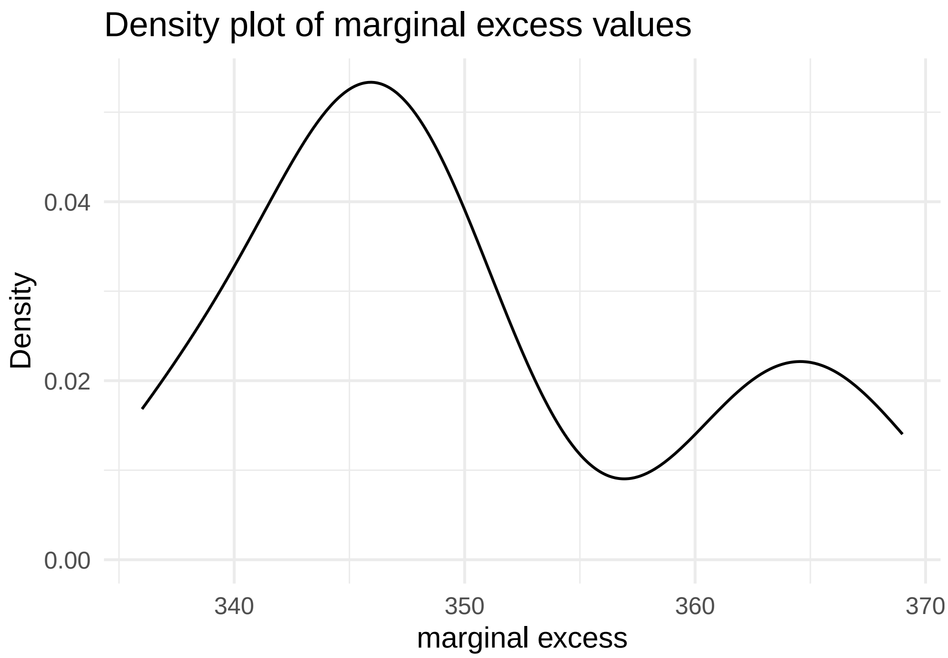
Et ainsi on peut dire que $k_{ij,\tau}|n_{j,\tau} \sim \text{Bin}(n_{j,\tau}, \chi_{ij,\tau})$. Où $\chi_{ij,\tau} = 2(1 - \Phi(\sqrt{0.5\gamma_{ij,\tau}}))$ ou encore $\chi_{ij,\tau} = P(X_{s_i,t} > q | X_{s_j,t+\tau} > q)$ avec $\gamma_{ij,\tau} = 2(\beta_1 ||h_{ij}||^{\alpha_1} + \beta_2 |\tau|^{\alpha_2})$.

```
calculate_marginal_excesses <- function(data_long, taus, q) {
  results <- list()

  for (tau in taus) {
    shifted_data <- data_long

    # Create a new column for shifted values
    shifted_data$value_shifted <- NA

    # Shift the values for the given tau
    for (site in unique(data_long$site)) {
      site_data <- data_long[data_long$site == site, ]
      if (tau == 0) {
        shifted_data$value_shifted[shifted_data$site == site] <- site_data$value
      } else {
        shifted_data$value_shifted[shifted_data$site == site] <- c(rep(NA, tau),
                                        site_data$value[1:(nrow(site_data) - tau)])
      }
    }

    # Calculate the excesses for the shifted values
    shifted_data$excess <- shifted_data$value_shifted > q

    # Aggregate to calculate the sum of excesses for each site
    nj_tau <- aggregate(excess ~ site, shifted_data, sum, na.rm = TRUE)
```

```r
    # Store the results in a list
    results[[as.character(tau)]] <- nj_tau
  }

  return(results)
}

nj_tau_list <- calculate_marginal_excesses(data_long, taus, q)

# get all excess values
all_values <- unlist(lapply(nj_tau_list, function(x) x$excess))

plot(all_values, main = "Marginal excesses",
                ylab = "nj_tau")
```




```r
# density plot
ggplot(data.frame(x = all_values), aes(x = x)) +
  geom_density() +
  labs(title = "Density plot of marginal excess values",
       x = "marginal excess",
       y = "Density") +
  theme_minimal()
```



Calculate the log-likelihood

$$\ell(\theta) = \sum_{\tau} \sum_{i} \sum_{j} k_{ij,\tau} \log(\chi_{ij,\tau}) + (n_{j,\tau} - k_{ij,\tau}) \log(1 - \chi_{ij,\tau})$$

```
# Function to calculate gamma_theta
gamma_theta <- function(h, tau, beta1, beta2, alpha1, alpha2) {
  gamma <- 2 * (beta1 * h^alpha1 + beta2 * abs(tau)^alpha2)

  if (gamma < 0) {
    gamma <- 0
  }
  return(gamma)
}

# Function to calculate chi_ij with distance matrix
chi_ij <- function(site_i, site_j, tau, theta, distance_matrix) {
  h <- distance_matrix[site_i, site_j]
  beta1 <- theta[1]
  beta2 <- theta[2]
  alpha1 <- theta[3]
  alpha2 <- theta[4]
  gamma <- gamma_theta(h, tau, beta1, beta2, alpha1, alpha2)
  return(2 * (1 - pnorm(sqrt(0.5 * gamma))))
}

log_likelihood <- function(theta, data, distance_matrix, taus, kij_tau_list,
                           nj_tau_list) {
  # print(theta)
  ll <- 0
  sites <- unique(data$site)
```

```r
  lower.bound <- c(1e-6, 1e-6, 1e-6, 1e-6)
  upper.bound <- c(Inf, Inf, 1.999, 1.999)

  # Check if the parameters are in the bounds
  if (any(theta < lower.bound) || any(theta > upper.bound)) {
    message("out of bounds")
    return(1e8)
  }

  for (tau in taus) {
    kij_matrix <- kij_tau_list[[as.character(tau)]]
    nj_df <- nj_tau_list[[as.character(tau)]]
    for (i in 1:length(sites)) {
      for (j in 1:length(sites)) {
        if (i != j) {
          site_i <- sites[i]
          site_j <- sites[j]
          nj <- nj_df$excess[nj_df$site == site_j]
          kij <- kij_matrix[site_i, site_j]

          chi <- chi_ij(site_i, site_j, tau, theta, distance_matrix)

          ll <- ll + kij * log(chi) + (nj - kij) * log(1 - chi)
        }
      }
    }
  }

  return(-ll)  # Negative log likelihood
}
```

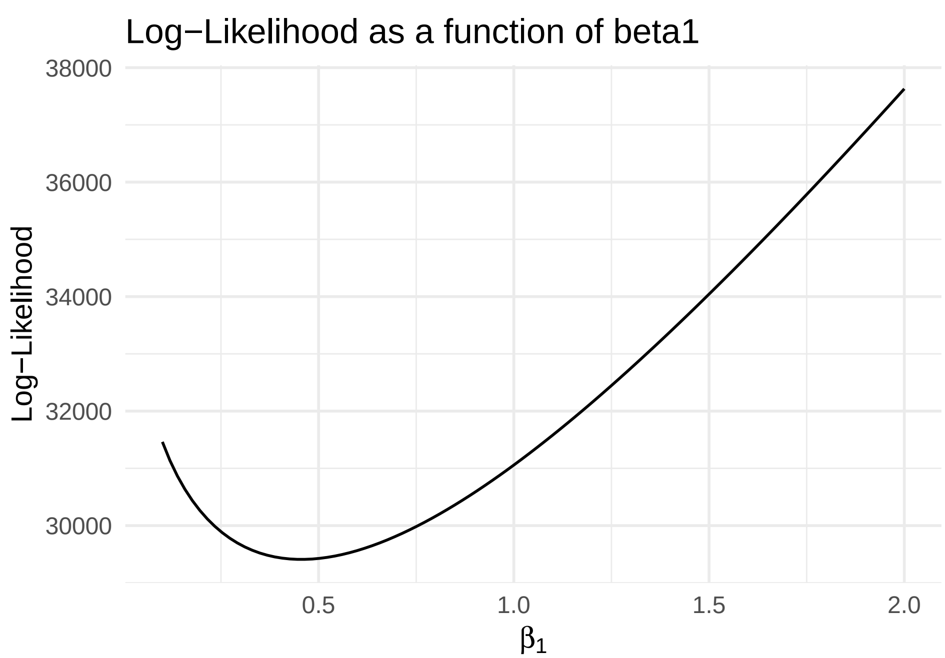
Variation of beta1, fixing the other parameters

```r
# Define the other theta parameters
beta1 <- 0.4
beta2 <- 0.2
alpha1 <- 1.5
alpha2 <- 1
```

Pour différentes valeurs proches de quantile, la convergence est plutôt changeante pour β_1 dont la vraie valeur est 0.4. Pour un quantile de 0.6, on obtient une valeur de β_1 proche de 0.4. Pour un quantile de 0.62, on obtient une valeur de β_1 proche de 0.5. Pour un quantile de 0.65, on obtient une valeur de β_1 proche de 0.6.

```r
q <- 0.61 # quantile
data_long <- excesses_indicators(data, q)
taus <- 0:10 # temporal lags
kij_tau_list <- calculate_kij_tau_list(data_long, taus) # conditional excesses
nj_tau_list <- calculate_marginal_excesses(data_long, taus, q)

# Generate a range of values for beta1
```

```r
beta1_values <- seq(0.1, 2, length.out = 100)

# Calculate the log-likelihood for each beta1 value
log_likelihood_values <- sapply(beta1_values, function(beta) {
  theta <- c(beta, beta2, alpha1, alpha2)
  log_likelihood(theta, data_long, distance_matrix, taus, kij_tau_list,
                 nj_tau_list)
})

# Data frame for ggplot
df <- data.frame(beta = beta1_values, log_likelihood = log_likelihood_values)

# Plot the log-likelihood as a function of beta1
ggplot(df, aes(x = beta, y = log_likelihood)) +
  geom_line() +
  labs(title = "Log-Likelihood as a function of beta1",
       x = expression(beta[1]),
       y = "Log-Likelihood") +
  theme_minimal()
```




```r
beta1_values[which.min(log_likelihood_values)]
```

```
## [1] 0.4646465
```

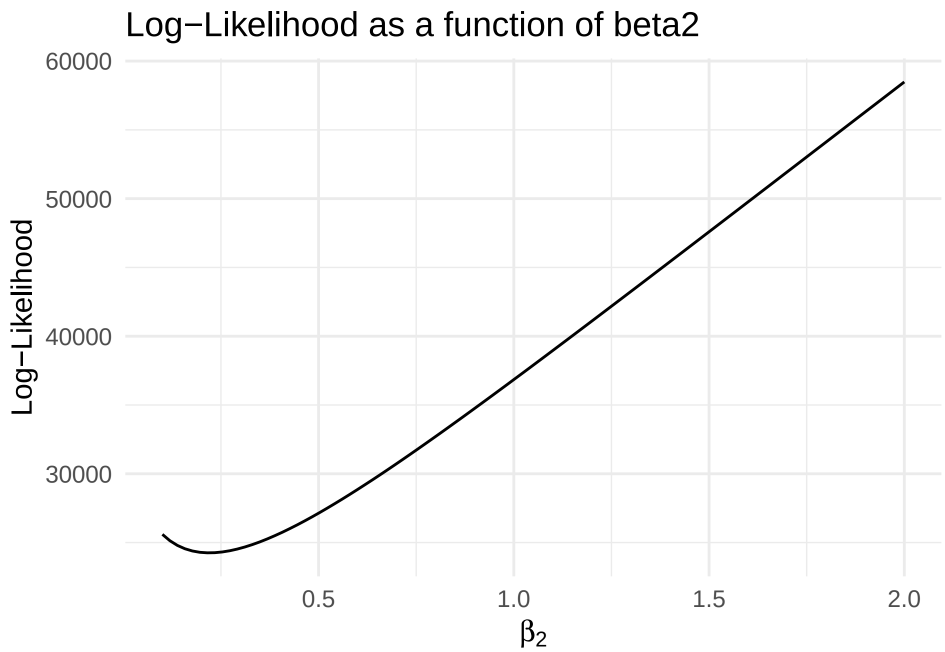
Variation of beta2, fixing the other parameters

Pour différentes valeurs proches de quantile, la convergence est plutôt stable pour β_2 pas très loin de la vraie valeur 0.2. Si on prend un quantile de 0.6, on obtient une valeur de β_2 proche de 0.14. Et si on prend un quantile de 0.65, on obtient une valeur de β_2 proche de 0.17. Si on prend un quantile de 0.68, on obtient une valeur de β_2 proche de 0.2. Si on prend un quantile de 0.7, on obtient une valeur de β_2 proche de 0.25.

```r
q <- 0.68 # quantile
data_long <- excesses_indicators(data, q)
print(head(data_long))
```

```
##   site time     value value_threshold excess
## 1   S1    1  1.572784        2.042166      0
## 2   S1    2  5.180775        2.042166      1
## 3   S1    3  2.897169        2.042166      1
## 4   S1    4  2.662405        2.042166      1
## 5   S1    5 10.207054        2.042166      1
## 6   S1    6  2.266449        2.042166      1
```

```r
taus <- 0:10 # temporal lags
kij_tau_list <- calculate_kij_tau_list(data_long, taus) # conditional excesses
nj_tau_list <- calculate_marginal_excesses(data_long, taus, q)

# Generate a range of values for beta2
beta2_values <- seq(0.1, 2, length.out = 100)

# Calculate the log-likelihood for each beta2 value
log_likelihood_values <- sapply(beta2_values, function(beta) {
  theta <- c(beta1, beta, alpha1, alpha2)
  log_likelihood(theta, data_long, distance_matrix, taus, kij_tau_list,
                 nj_tau_list)
})

# Data frame for ggplot
df <- data.frame(beta = beta2_values, log_likelihood = log_likelihood_values)

# Plot the log-likelihood as a function of beta2
ggplot(df, aes(x = beta, y = log_likelihood)) +
  geom_line() +
  labs(title = "Log-Likelihood as a function of beta2",
       x = expression(beta[2]),
       y = "Log-Likelihood") +
  theme_minimal()
```




```
beta2_values[which.min(log_likelihood_values)]
```

```
## [1] 0.2151515
```

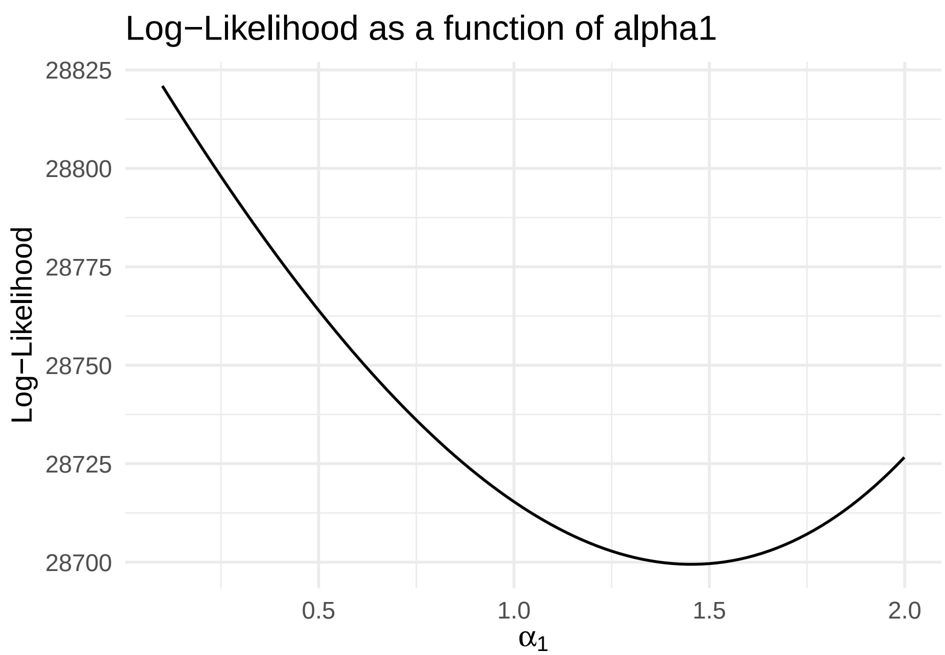
Variation of alpha1, fixing the other parameters

Pour une petite variation de la valeur du quantile, on obtient une valeur de α_1 très variable et qui ne converge pas toujours vers la vraie valeur de α_1.

Si on prend un quantile de 0.62, on obtient une valeur de α_1 proche de 1.5. Mais si on prend un quantile de 0.6, on obtient une valeur de α_1 proche de 1. Et si on prend un quantile de 0.65, on obtient une valeur de α_1 proche de 2.

```
q <- 0.62 # quantile
data_long <- excesses_indicators(data, q)
taus <- 0:10 # temporal lags
kij_tau_list <- calculate_kij_tau_list(data_long, taus) # conditional excesses
nj_tau_list <- calculate_marginal_excesses(data_long, taus, q)

# Generate a range of values for alpha1
alpha1_values <- seq(0.1, 1.999, length.out = 100)

# Calculate the log-likelihood for each alpha1 value
log_likelihood_values <- sapply(alpha1_values, function(alpha) {
  theta <- c(beta1, beta2, alpha, alpha2)
  log_likelihood(theta, data_long, distance_matrix, taus, kij_tau_list,
                 nj_tau_list)
})

# Data frame for ggplot
df <- data.frame(beta = alpha1_values, log_likelihood = log_likelihood_values)
```

```
# Plot the log-likelihood as a function of alpha1
ggplot(df, aes(x = beta, y = log_likelihood)) +
  geom_line() +
  labs(title = "Log-Likelihood as a function of alpha1",
       x = expression(alpha[1]),
       y = "Log-Likelihood") +
  theme_minimal()
```




```
alpha1_values[which.min(log_likelihood_values)]
```

```
## [1] 1.461909
```

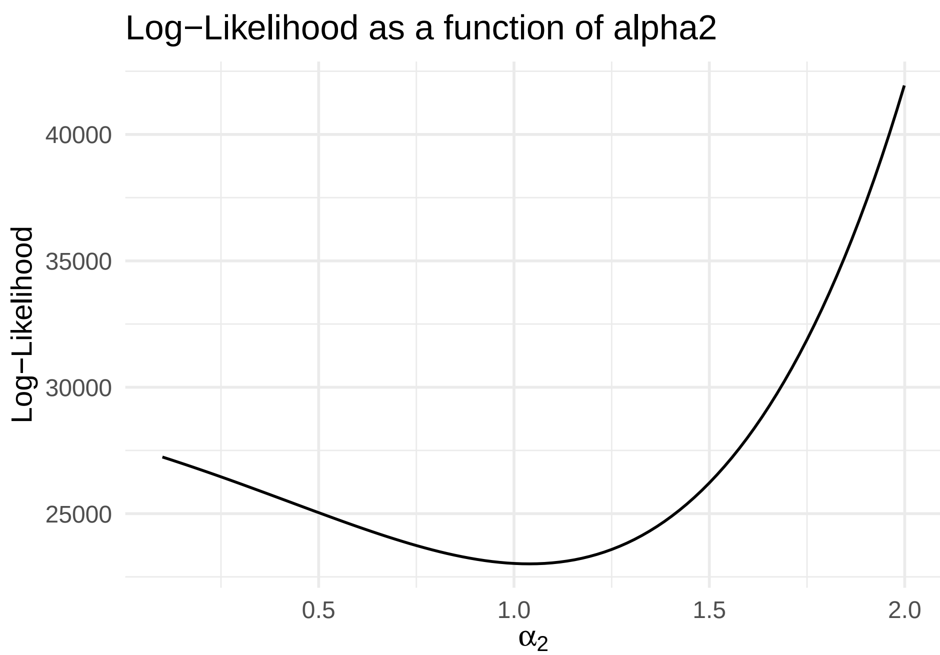
Variation of alpha2, fixing the other parameters

Pour une petite variation de la valeur du quantile, on obtient une valeur de α_2 oscillant autour de 1 pour le paramètre α_2.

Si on prend un quantile de 0.62, on obtient une valeur de α_2 proche de 0.8. Si on prend un quantile de 0.65, on obtient une valeur de α_2 proche de 0.9. Si on prend un quantile de 0.68, on obtient une valeur de α_2 proche de 1. Si on prend un quantile de 0.7, on obtient une valeur de α_2 proche de 1. Si on prend un quantile de 0.75, on obtient une valeur de α_2 proche de 1.1.

```
q <- 0.7 # quantile
data_long <- excesses_indicators(data, q)
taus <- 0:10 # temporal lags
kij_tau_list <- calculate_kij_tau_list(data_long, taus) # conditional excesses
nj_tau_list <- calculate_marginal_excesses(data_long, taus, q)

# Generate a range of values for alpha2
alpha2_values <- seq(0.1, 1.999, length.out = 100)
```

```r
# Calculate the log-likelihood for each alpha2 value
log_likelihood_values <- sapply(alpha2_values, function(alpha) {
  theta <- c(beta1, beta2, alpha1, alpha)
  log_likelihood(theta, data_long, distance_matrix, taus, kij_tau_list,
                 nj_tau_list)
})

# Data frame for ggplot
df <- data.frame(alpha = alpha2_values, log_likelihood = log_likelihood_values)

# Plot the log-likelihood as a function of alpha2
ggplot(df, aes(x = alpha, y = log_likelihood)) +
  geom_line() +
  labs(title = "Log-Likelihood as a function of alpha2",
       x = expression(alpha[2]),
       y = "Log-Likelihood") +
  theme_minimal()
```




```r
alpha2_values[which.min(log_likelihood_values)]
```

```
## [1] 1.039909
```

Optimization

Pour l'optimisation, un problème est donc de choisir une valeur de quantile qui permet d'obtenir une convergence des paramètres alors que l'on a vu que ce n'est pas forcément la même valeur de quantile qui permet d'obtenir une valeur proche des vrais paramètres avec des paramètres fixés et aussi une légère variation de quantile peut changer significativement la valeur des paramètres estimés (surtout le paramètre α_1). Je décide de prendre la valeur de quantile pour laquelle on obtient une valeur de α_1 proche de 1.5.

```r
# Initialize theta parameters with true simulation parameters
theta_init <- c(0.4, 0.2, 1.5, 1)

q <- 0.62 # quantile
data_long <- excesses_indicators(data, q)
taus <- 0:10 # temporal lags
kij_tau_list <- calculate_kij_tau_list(data_long, taus) # conditional excesses
nj_tau_list <- calculate_marginal_excesses(data_long, taus, q)

# Optimize composite log-likelihood
optim_result <- optim(par = theta_init,
                      fn  = log_likelihood,
                      data = data_long,
                      distance_matrix = distance_matrix,
                      taus = taus,
                      kij_tau_list = kij_tau_list,
                      nj = nj_tau_list,
                      method = "CG", # "L-BFGS-B"
                      control = list(maxit = 5000))

convergence <- optim_result$convergence
theta_opt <- optim_result$par

print(convergence) # 0 means convergence
```

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

```r
print(theta_opt)
```

```
## [1] 0.9743115 0.1105188 0.2885603 0.7057527
```

Cela converge mais ne donne pas les vraies valeurs des paramètres. J'ai essayé avec d'autres méthodes d'optimisation mais cela donne le même résultat.

Optimization, fixing some parameters

J'utilise une autre fonction d'optimisation qui me permet de fixer facilement des paramètres. Pour cela j'ai du modifier un peu la fonction de log-vraisemblance pour qu'elle prenne en compte les paramètres fixes.

```r
log_likelihood_par <- function(beta1, beta2, alpha1, alpha2, data,
                               distance_matrix, taus, kij_tau_list,
                               nj_tau_list) {
  theta <- c(beta1, beta2, alpha1, alpha2)
  ll <- 0
  sites <- unique(data$site)

  lower.bound <- c(1e-6, 1e-6, 1e-6, 1e-6)
  upper.bound <- c(Inf, Inf, 1.999, 1.999)

  # Check if the parameters are in the bounds
  if (any(theta < lower.bound) || any(theta > upper.bound)) {
```

```
    print("out of bounds")
    return(1e8)
  }

  for (tau in taus) {
    kij_matrix <- kij_tau_list[[as.character(tau)]]
    nj_df <- nj_tau_list[[as.character(tau)]]
    for (i in 1:length(sites)) {
      for (j in 1:length(sites)) {
        if (i != j) {
          site_i <- sites[i]
          site_j <- sites[j]
          nj <- nj_df$excess[nj_df$site == site_j]
          kij <- kij_matrix[site_i, site_j]

          chi <- chi_ij(site_i, site_j, tau, theta, distance_matrix)

          ll <- ll + kij * log(chi) + (nj - kij) * log(1 - chi)
        }
      }
    }
  }

  return(-ll)  # Negative log likelihood
}
```

Optimization, fixing β_2, α_1 and α_2

Je fixe les paramètres β_2, α_1 et α_2

```
library(bbmle)

# q = 0.62 (reminder)
res <- mle2(log_likelihood_par, start = list(beta1 = theta_init[1],
                                 beta2 = theta_init[2],
                                 alpha1 = theta_init[3],
                                 alpha2 = theta_init[4]),
                 data = list(data = data_long,
                      distance_matrix = distance_matrix,
                      taus = taus,
                      kij_tau_list = kij_tau_list,
                      nj_tau_list = nj_tau_list,
                      method = "L-BFGS-B",
                      lower = c(1e-6, 1e-6, 1e-6, 1e-6),
                      upper = c(Inf, Inf, 1.999, 1.999)),
                  control = list(maxit = 10000),
                  fixed = list(beta2 = 0.2, alpha1 = 1.5, alpha2 = 1))

print(res@details$conv) # 0 means convergence
```

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

```r
print(res@coef)
```

```
##     beta1
## 0.4852404
```

Ici on obtient bien une valeur relativement proche de la vraie valeur de β_1. Et si on modifie le quantile on obtient une valeur encore plus proche de la vraie valeur de β_1 étant 0.4. C'est cohérent avec le plot de vraisemblance précédent. Avec différentes méthodes d'optimisation, on obtient les mêmes résultats.

```r
q <- 0.6 # modified quantile
data_long <- excesses_indicators(data, q)
taus <- 0:10 # temporal lags
kij_tau_list <- calculate_kij_tau_list(data_long, taus) # conditional excesses
nj_tau_list <- calculate_marginal_excesses(data_long, taus, q)

res <- mle2(log_likelihood_par, start = list(beta1 = theta_init[1],
                                  beta2 = theta_init[2],
                                  alpha1 = theta_init[3],
                                  alpha2 = theta_init[4]),
                 data = list(data = data_long,
                      distance_matrix = distance_matrix,
                      taus = taus,
                      kij_tau_list = kij_tau_list,
                      nj_tau_list = nj_tau_list,
                      method = "CG"),
                  control = list(maxit = 100),
                  fixed = list(beta2 = 0.2, alpha1 = 1.5, alpha2 = 1))

print(res@details$conv) # 0 means convergence
```

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

```r
print(res@coef)
```

```
##     beta1
## 0.4220089
```

Optimization, fixing β_1, α_1 and α_2

Idem, fonctionne bien mais pour un bon choix de quantile pour récupérer une valeur proche de la vraie valeur de β_2 étant 0.2.

```r
q <- 0.68 # modified quantile
data_long <- excesses_indicators(data, q)
taus <- 0:10 # temporal lags
kij_tau_list <- calculate_kij_tau_list(data_long, taus) # conditional excesses
nj_tau_list <- calculate_marginal_excesses(data_long, taus, q)

res <- mle2(log_likelihood_par, start = list(beta1 = theta_init[1],
                                  beta2 = theta_init[2],
```

```
                                  alpha1 = theta_init[3],
                                  alpha2 = theta_init[4]),
                  data = list( data = data_long,
                       distance_matrix = distance_matrix,
                       taus = taus,
                       kij_tau_list = kij_tau_list,
                       nj_tau_list = nj_tau_list,
                       method = "CG"),
                   control = list(maxit = 10000),
                   fixed = list(beta1 = 0.4, alpha1 = 1.5, alpha2 = 1))

print(res@details$conv) # 0 means convergence
```

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

```
print(res@coef)
```

```
##    beta2
## 0.220311
```

Optimization, fixing β_1, β_2 and α_2

Idem pour obtenir une valeur proche de la vraie valeur de α_1 étant 1.5 en prenant un bon choix de quantile, sinon ça ne marche pas.

```
q <- 0.62 # modified quantile
data_long <- excesses_indicators(data, q)
taus <- 0:10 # temporal lags
kij_tau_list <- calculate_kij_tau_list(data_long, taus) # conditional excesses
nj_tau_list <- calculate_marginal_excesses(data_long, taus, q)

res <- mle2(log_likelihood_par, start = list(beta1 = theta_init[1],
                                  beta2 = theta_init[2],
                                  alpha1 = theta_init[3],
                                  alpha2 = theta_init[4]),
                  data = list(data = data_long,
                       distance_matrix = distance_matrix,
                       taus = taus,
                       kij_tau_list = kij_tau_list,
                       nj_tau_list = nj_tau_list,
                       method = "CG"),
                   control = list(maxit = 10000),
                   fixed = list(beta1 = 0.4, beta2 = 0.2, alpha2 = 1))
```

```
## [1] "out of bounds"
## [1] "out of bounds"
```

```
print(res@details$conv) # 0 means convergence
```

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

```
print(res@coef)
```

```
##   alpha1
## 1.454047
```

Optimization, fixing β_1, β_2 and α_1

Idem pour obtenir une valeur proche de la vraie valeur de α_2 étant 1 en prenant un bon choix de quantile.

```
q <- 0.68 # modified quantile
data_long <- excesses_indicators(data, q)
taus <- 0:10 # temporal lags
kij_tau_list <- calculate_kij_tau_list(data_long, taus) # conditional excesses
nj_tau_list <- calculate_marginal_excesses(data_long, taus, q)

res <- mle2(log_likelihood_par, start = list(beta1 = theta_init[1],
                                 beta2 = theta_init[2],
                                 alpha1 = theta_init[3],
                                 alpha2 = theta_init[4]),
                 data = list(data = data_long,
                      distance_matrix = distance_matrix,
                      taus = taus,
                      kij_tau_list = kij_tau_list,
                      nj_tau_list = nj_tau_list,
                      method = "CG"),
                  control = list(maxit = 10000),
                  fixed = list(beta1 = 0.4, beta2 = 0.2, alpha1 = 1.5))
```

```
## [1] "out of bounds"
## [1] "out of bounds"
## [1] "out of bounds"
## [1] "out of bounds"
```

```
print(res@details$conv) # 0 means convergence
```

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

```
print(res@coef)
```

```
##    alpha2
## 0.9900594
```

Optimization, fixing β_1, α_1 the spatial parameters

Ici cela ne fonctionne plus du tout surement car les quantiles pour obtenir une bonne convergence individuelle des paramètres ne sont pas les mêmes.

```
q <- 0.6 # quantile
data_long <- excesses_indicators(data, q)
taus <- 0:10 # temporal lags
kij_tau_list <- calculate_kij_tau_list(data_long, taus) # conditional excesses
nj_tau_list <- calculate_marginal_excesses(data_long, taus, q)
res <- mle2(log_likelihood_par, start = list(beta1 = theta_init[1],
                                beta2 = theta_init[2],
                                alpha1 = theta_init[3],
                                alpha2 = theta_init[4]),
                data = list(data = data_long,
                     distance_matrix = distance_matrix,
                     taus = taus,
                     kij_tau_list = kij_tau_list,
                     nj_tau_list = nj_tau_list,
                     method = "CG"),
                 control = list(maxit = 10000),
                 fixed = list(beta1 = 0.4, alpha1 = 1.5))
```

```
## [1] "out of bounds"
## [1] "out of bounds"
## [1] "out of bounds"
## [1] "out of bounds"
## [1] "out of bounds"
## [1] "out of bounds"
## [1] "out of bounds"
## [1] "out of bounds"
## [1] "out of bounds"
## [1] "out of bounds"
## [1] "out of bounds"
## [1] "out of bounds"
## [1] "out of bounds"
## [1] "out of bounds"
## [1] "out of bounds"
## [1] "out of bounds"
## [1] "out of bounds"
## [1] "out of bounds"
## [1] "out of bounds"
## [1] "out of bounds"
## [1] "out of bounds"
```

```
print(res@details$conv) # 0 means convergence
```

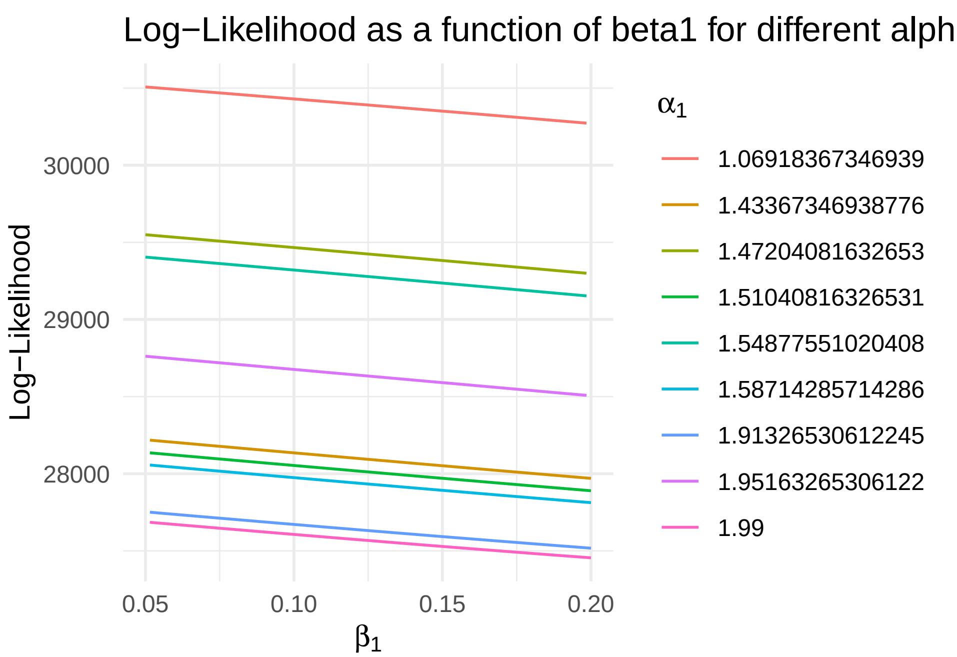
```
## [1] 0
```

```
print(res@coef)
```

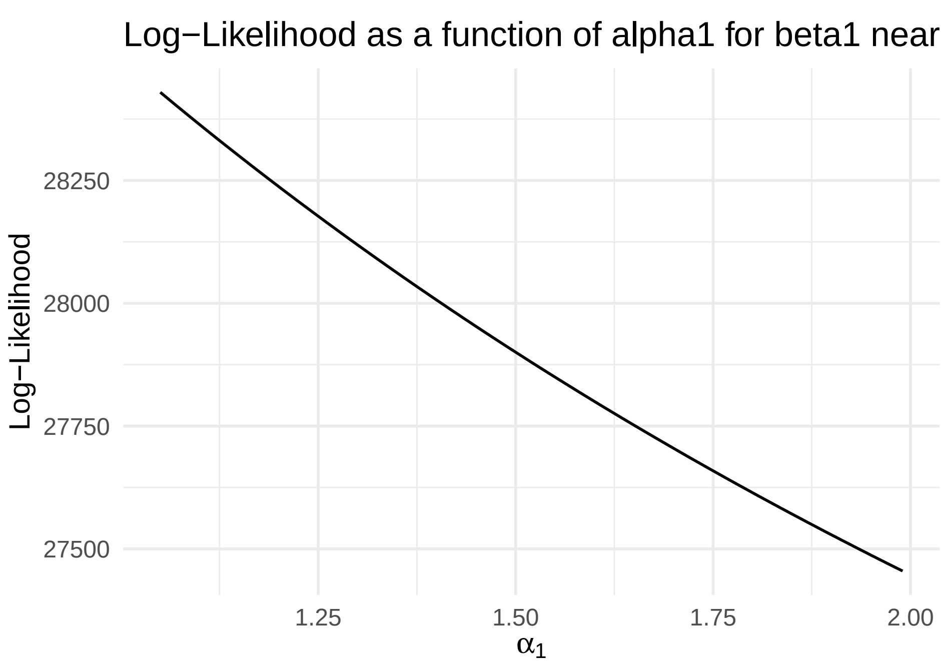
```
##     beta2    alpha2
## 0.5274704 0.2486680
```

Ici je trace la vraisemblance en fonction de β_1 pour différentes valeurs de α_1.

```r
q <- 0.65 # quantile
data_long <- excesses_indicators(data, q)
taus <- 0:10 # temporal lags
kij_tau_list <- calculate_kij_tau_list(data_long, taus) # conditional excesses
nj_tau_list <- calculate_marginal_excesses(data_long, taus, q)

# variation of beta1 and alpha1
beta1_values <- seq(0.05, 0.2, length.out = 100)
alpha1_values <- seq(1.05, 1.99, length.out = 50)

# Calculate the log-likelihood for each beta1 and alpha1 value
log_likelihood_values <- matrix(0, nrow = length(beta1_values),
                                ncol = length(alpha1_values))

for (i in 1:length(beta1_values)) {
  for (j in 1:length(alpha1_values)) {
    theta <- c(beta1_values[i], beta2, alpha1_values[j], alpha2)
    log_likelihood_values[i, j] <- log_likelihood(theta, data_long,
                                              distance_matrix, taus,
                                              kij_tau_list, nj_tau_list)
  }
}

# Data frame for ggplot
df <- data.frame(beta1 = rep(beta1_values, each = length(alpha1_values)),
                 alpha1 = rep(alpha1_values, length(beta1_values)),
                 log_likelihood = as.vector(log_likelihood_values))

# Filter the data frame for the desired alpha1 values
df_alpha1 <- df[abs(df$alpha1 - c(0.5, 1, 1.5, 2)) < 0.1, ]

# Plot the log-likelihood as a function of beta1 for different alpha1
ggplot(df_alpha1, aes(x = beta1, y = log_likelihood, color = factor(alpha1))) +
  geom_line() +
  labs(title = "Log-Likelihood as a function of beta1 for different alpha1",
    x = expression(beta[1]),
    y = "Log-Likelihood",
    color = expression(alpha[1])) +
  theme_minimal()
```




```
df_beta1 <- df[df$beta1 == beta1_values[which.min(abs(beta1_values - 0.4))], ]

# Plot the log-likelihood as a function of alpha1
ggplot(df_beta1, aes(x = alpha1, y = log_likelihood)) +
  geom_line() +
  labs(title = "Log-Likelihood as a function of alpha1 for beta1 near 0.4",
       x = expression(alpha[1]),
       y = "Log-Likelihood") +
  theme_minimal()
```



On voit qu'on a aucun minimum ici même lorsque le beta1 est proche de 0.4.

Optimization, fixing β_2, α_2 the temporal parameters

Ici cela ne fonctionne plus du tout surement car les quantiles pour obtenir une bonne convergence individuelle des paramètres ne sont pas les mêmes.

```
q <- 0.6 # quantile
data_long <- excesses_indicators(data, q)
taus <- 0:10 # temporal lags
kij_tau_list <- calculate_kij_tau_list(data_long, taus) # conditional excesses
nj_tau_list <- calculate_marginal_excesses(data_long, taus, q)
res <- mle2(log_likelihood_par, start = list(beta1 = theta_init[1],
                                  beta2 = theta_init[2],
                                  alpha1 = theta_init[3],
                                  alpha2 = theta_init[4]),
                 data = list(data = data_long,
                      distance_matrix = distance_matrix,
                      taus = taus,
                      kij_tau_list = kij_tau_list,
                      nj_tau_list = nj_tau_list,
                      method = "CG"),
                  control = list(maxit = 10000),
                  fixed = list(beta2 = 0.2, alpha2 = 1))
```

```
## [1] "out of bounds"
## [1] "out of bounds"
## [1] "out of bounds"
## [1] "out of bounds"
## [1] "out of bounds"
## [1] "out of bounds"
```

```
print(res@details$conv) # 0 means convergence
```

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

```
print(res@coef)
```

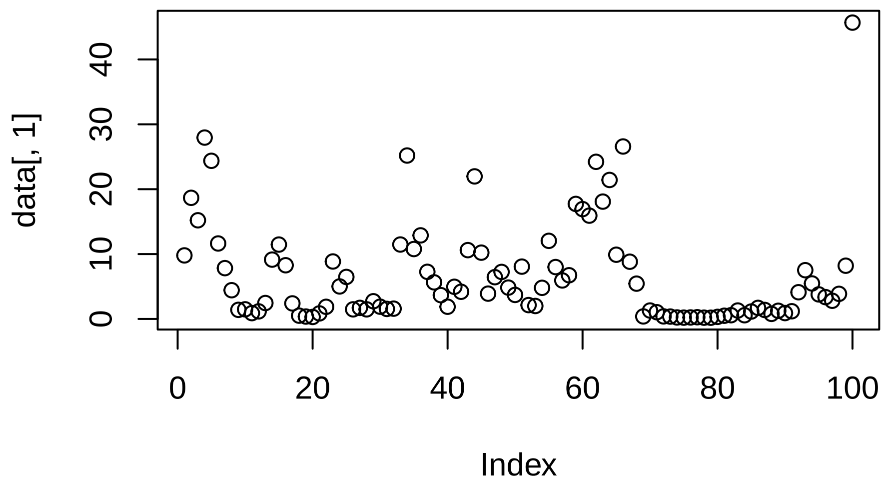
```
##     beta1    alpha1
## 0.4941840 0.4440029
```

New simulation

Simulation avec 25 sites et 300 pas de temps.

```
# spatial and temporal structures
ngrid <- 5
spa <- 1:ngrid
nsites <- ngrid^2 # if the grid is squared
temp <- 1:300

# Simulation
num_iterations <- 5
```

```r
list_BR <- list()
for (i in 1:num_iterations) {
  file_path <- paste0("../data/simulations_BR/sim_", ngrid^2, "s_",
                      length(temp), "t/br_", ngrid^2, "s_", length(temp), "t_",
                      i, ".csv")
  df <- read.csv(file_path)
  list_BR[[i]] <- df
}

simu_df <- list_BR[[1]] # first simulation
nsites <- ncol(simu_df) # number of sites
data <- simu_df
plot(data[, 1])
```




```r
# Calculate distance matrix
sites_coords <- expand.grid(x = spa, y = spa)
distance_matrix <- as.matrix(dist(sites_coords))

# Calculate excesses indicators
q <- 0.7 # quantile
data_long <- excesses_indicators(data, q)
print(head(data_long))
```

```
##   site time     value value_threshold excess
## 1   S1    1  9.793713        8.018164      1
## 2   S1    2 18.667294        8.018164      1
## 3   S1    3 15.207787        8.018164      1
## 4   S1    4 27.955214        8.018164      1
## 5   S1    5 24.374926        8.018164      1
## 6   S1    6 11.630862        8.018164      1
```

```r
taus <- 0:10 # temporal lags
kij_tau_list <- calculate_kij_tau_list(data_long, taus) # conditional excesses
print(kij_tau_list$`0`)
```

```
##     S1 S10 S11 S12 S13 S14 S15 S16 S17 S18 S19 S2 S20 S21 S22 S23 S24 S25 S3 S4
```

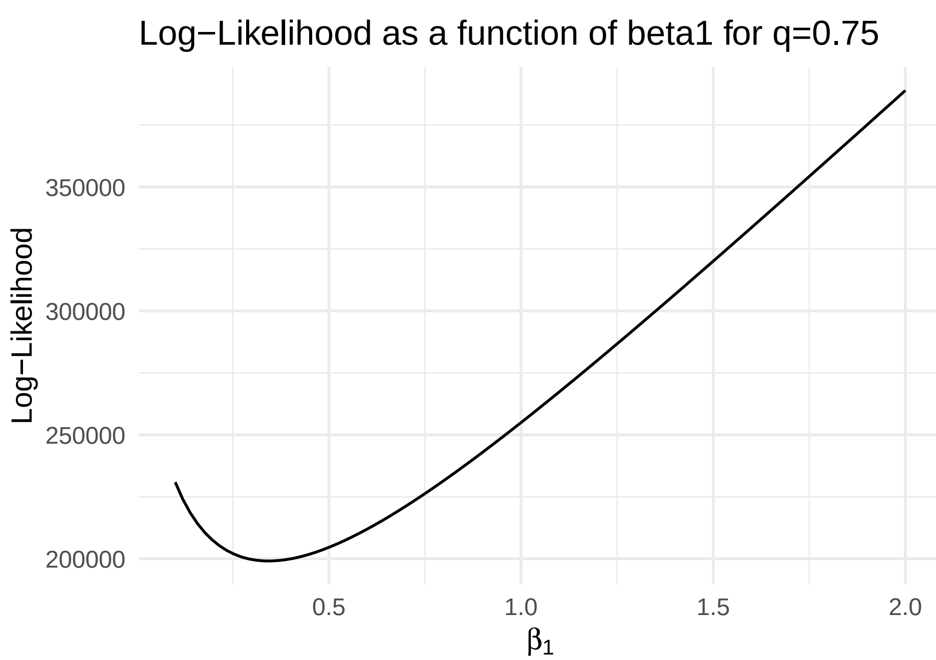
```
## S1   30   8  24  18  16  15  14  19  19  22  16 18  15  13  18  20  16  15 19 12
## S10   8  30   7  18  19  21  22   4  13  14  15 19  16   4   8  11   9  10 16 25
## S11  24   7  30  19  15  16  15  24  24  22  22 15  21  17  24  26  22  21 15 11
## S12  18  18  19  30  22  27  25  16  24  25  23 22  23  11  18  23  16  17 18 22
## S13  16  19  15  22  30  21  19   9  16  23  15 28  15   4  10  15   9   9 24 24
## S14  15  21  16  27  21  30  28  13  22  22  23 19  23  10  17  20  16  17 15 22
## S15  14  22  15  25  19  28  30  12  21  20  23 17  24  11  16  19  17  18 13 21
## S16  19   4  24  16   9  13  12  30  21  16  19  9  18  23  26  23  25  24  9  8
## S17  19  13  24  24  16  22  21  21  30  23  27 16  26  16  24  28  22  23 16 16
## S18  22  14  22  25  23  22  20  16  23  30  21 23  20  10  17  22  16  16 23 18
## S19  16  15  22  23  15  23  23  19  27  21  30 15  29  16  23  26  23  24 14 15
## S2   18  19  15  22  28  19  17   9  16  23  15 30  15   4  10  15   9   9 26 24
## S20  15  16  21  23  15  23  24  18  26  20  29 15  30  16  22  25  23  24 13 15
## S21  13   4  17  11   4  10  11  23  16  10  16  4  16  30  21  17  22  22  3  3
## S22  18   8  24  18  10  17  16  26  24  17  23 10  22  21  30  25  27  27 10 10
## S23  20  11  26  23  15  20  19  23  28  22  26 15  25  17  25  30  23  24 15 15
## S24  16   9  22  16   9  16  17  25  22  16  23  9  23  22  27  23  30  29  9  8
## S25  15  10  21  17   9  17  18  24  23  16  24  9  24  22  27  24  29  30  9  9
## S3   19  16  15  18  24  15  13   9  16  23  14 26  13   3  10  15   9   9 30 20
## S4   12  25  11  22  24  22  21   8  16  18  15 24  15   3  10  15   8   9 20 30
## S5    9  28   8  19  20  22  22   5  14  15  16 20  16   3   9  12   9  10 17 26
## S6   22  10  27  22  16  19  18  22  27  23  24 16  23  16  24  28  22  23 16 14
## S7   12  25  11  22  24  22  21   8  16  18  15 24  15   3  10  15   8   9 20 30
## S8   14  23  13  22  26  22  20   8  16  20  15 26  15   3  10  15   8   9 23 27
## S9   10  27  10  21  22  23  22   7  16  17  16 21  16   5  11  14   9  10 17 27
##     S5 S6 S7 S8 S9
## S1   9 22 12 14 10
## S10 28 10 25 23 27
## S11  8 27 11 13 10
## S12 19 22 22 22 21
## S13 20 16 24 26 22
## S14 22 19 22 22 23
## S15 22 18 21 20 22
## S16  5 22  8  8  7
## S17 14 27 16 16 16
## S18 15 23 18 20 17
## S19 16 24 15 15 16
## S2  20 16 24 26 21
## S20 16 23 15 15 16
## S21  3 16  3  3  5
## S22  9 24 10 10 11
## S23 12 28 15 15 14
## S24  9 22  8  8  9
## S25 10 23  9  9 10
## S3  17 16 20 23 17
## S4  26 14 30 27 27
## S5  30 11 26 24 26
## S6  11 30 14 16 13
## S7  26 14 30 27 27
## S8  24 16 27 30 24
## S9  26 13 27 24 30
```

```r
nj_tau_list <- calculate_marginal_excesses(data_long, taus, q)
print(nj_tau_list$`0`)
```

```
##    site excess
## 1    S1     84
## 2    S2     83
## 3    S3     75
## 4    S4     76
## 5    S5     73
## 6    S6     81
## 7    S7     83
## 8    S8     77
## 9    S9     78
## 10  S10     77
## 11  S11     90
## 12  S12     86
## 13  S13     80
## 14  S14     77
## 15  S15     78
## 16  S16     91
## 17  S17     87
## 18  S18     74
## 19  S19     81
## 20  S20     82
## 21  S21     96
## 22  S22     92
## 23  S23     86
## 24  S24     90
## 25  S25     89
```

Variation of beta1, fixing the other parameters

Pour un quantile de 0.8, on obtient une valeur de β_1 proche de 0.48. Pour un quantile de 0.75, on obtient une valeur de β_1 proche de 0.35. Pour un quantile de 0.7, on obtient une valeur de β_1 proche de 0.25.

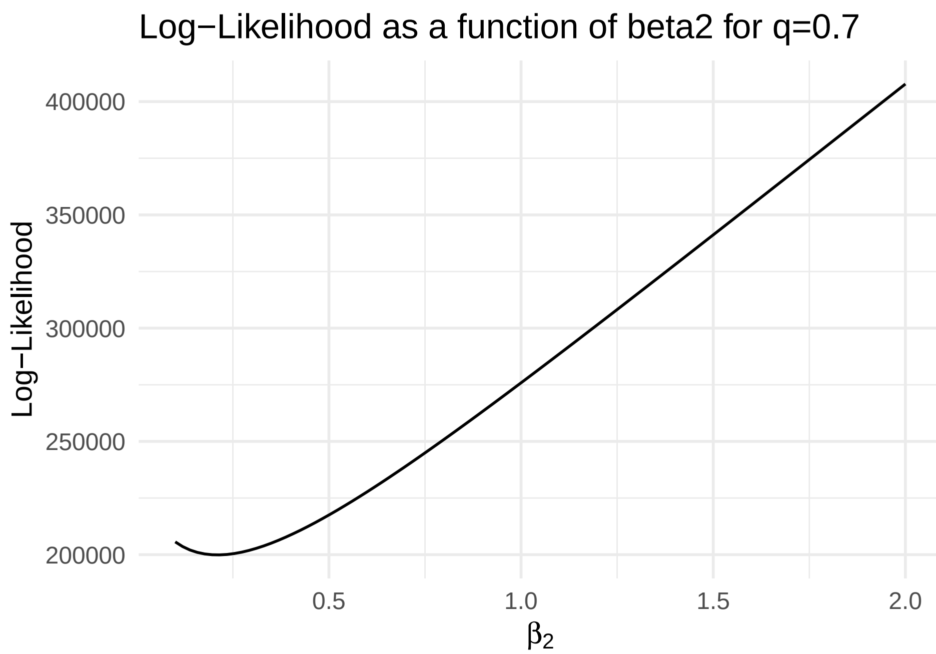


```
## [1] 0.3494949
```

Variation of beta2, fixing the other parameters

Si on prend un quantile de 0.7, on obtient une valeur de β_2 proche de 0.13. Si on prend un quantile de 0.75, on obtient une valeur de β_2 proche de 0.21.

```
##   site time     value value_threshold excess
## 1   S1    1  9.793713        8.928166      1
## 2   S1    2 18.667294        8.928166      1
## 3   S1    3 15.207787        8.928166      1
## 4   S1    4 27.955214        8.928166      1
## 5   S1    5 24.374926        8.928166      1
## 6   S1    6 11.630862        8.928166      1
```



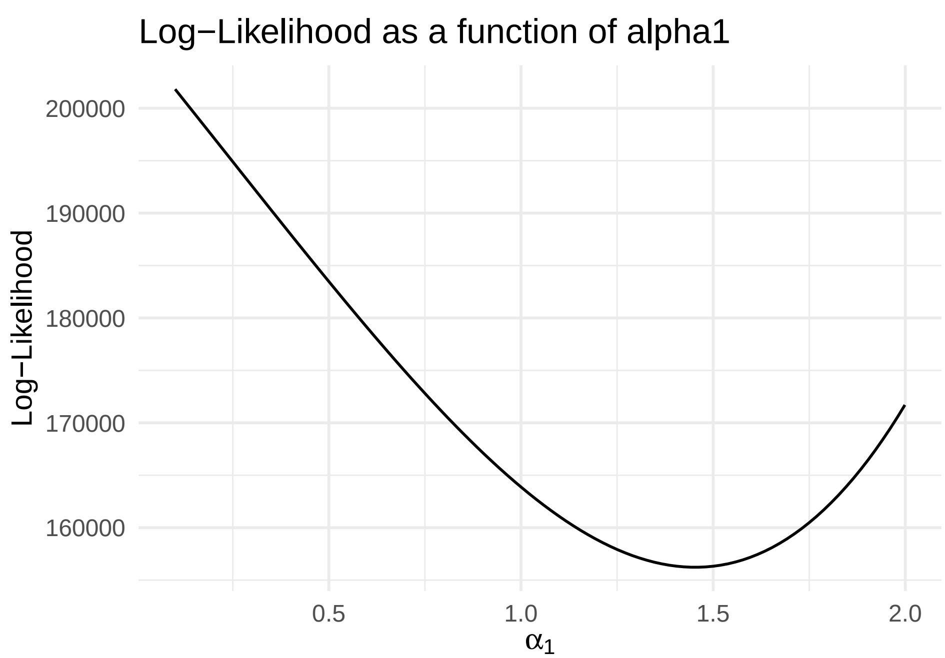

```
## [1] 0.2151515
```

Variation of alpha1, fixing the other parameters

Si on prend un quantile de 0.7, on obtient une valeur de α_1 proche de 1. Si on prend un quantile de 0.75, on obtient une valeur de α_1 proche de 1.2. Si on prend un quantile de 0.8, on obtient une valeur de α_1 proche de 1.5.

```
q <- 0.8 # quantile
data_long <- excesses_indicators(data, q)
taus <- 0:10 # temporal lags
kij_tau_list <- calculate_kij_tau_list(data_long, taus) # conditional excesses
nj_tau_list <- calculate_marginal_excesses(data_long, taus, q)

# Generate a range of values for alpha1
```

```r
alpha1_values <- seq(0.1, 1.999, length.out = 100)

# Calculate the log-likelihood for each alpha1 value
log_likelihood_values <- sapply(alpha1_values, function(alpha) {
  theta <- c(beta1, beta2, alpha, alpha2)
  log_likelihood(theta, data_long, distance_matrix, taus, kij_tau_list,
                 nj_tau_list)
})

# Data frame for ggplot
df <- data.frame(beta = alpha1_values, log_likelihood = log_likelihood_values)

# Plot the log-likelihood as a function of alpha1
ggplot(df, aes(x = beta, y = log_likelihood)) +
  geom_line() +
  labs(title = "Log-Likelihood as a function of alpha1",
       x = expression(alpha[1]),
       y = "Log-Likelihood") +
  theme_minimal()
```

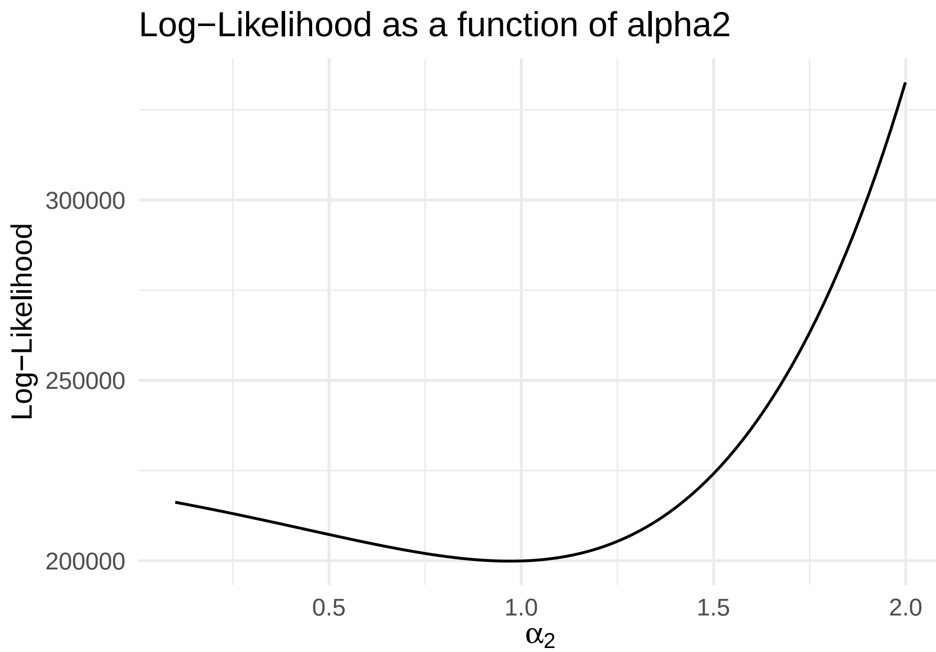



```r
alpha1_values[which.min(log_likelihood_values)]
```

```
## [1] 1.461909
```

Variation of alpha2, fixing the other parameters

Si on prend un quantile de 0.7, on obtient une valeur de α_2 proche de 0.7. Si on prend un quantile de 0.75, on obtient une valeur de α_2 proche de 1. Si on prend un quantile de 0.8, on obtient une valeur de α_2 proche de 1.2.




```
## [1] 0.9631818
```

Optimization

Trop long et ne donne pas du tout les vraies valeurs des paramètres.

```
q <- 0.8 # quantile
data_long <- excesses_indicators(data, q)
taus <- 0:10 # temporal lags
kij_tau_list <- calculate_kij_tau_list(data_long, taus) # conditional excesses
nj_tau_list <- calculate_marginal_excesses(data_long, taus, q)

theta_init <- c(0.4, 0.2, 1.5, 1) # true parameters
# Optimize composite log-likelihood
optim_result <- optim(par = theta_init,
                      fn  = log_likelihood,
                      data = data_long,
                      distance_matrix = distance_matrix,
                      taus = taus,
                      kij_tau_list = kij_tau_list,
                      nj = nj_tau_list,
                      method = "CG", # "L-BFGS-B"
                      control = list(maxit = 5000))

convergence <- optim_result$convergence
theta_opt <- optim_result$par

print(convergence) # 0 means convergence
print(theta_opt)
```