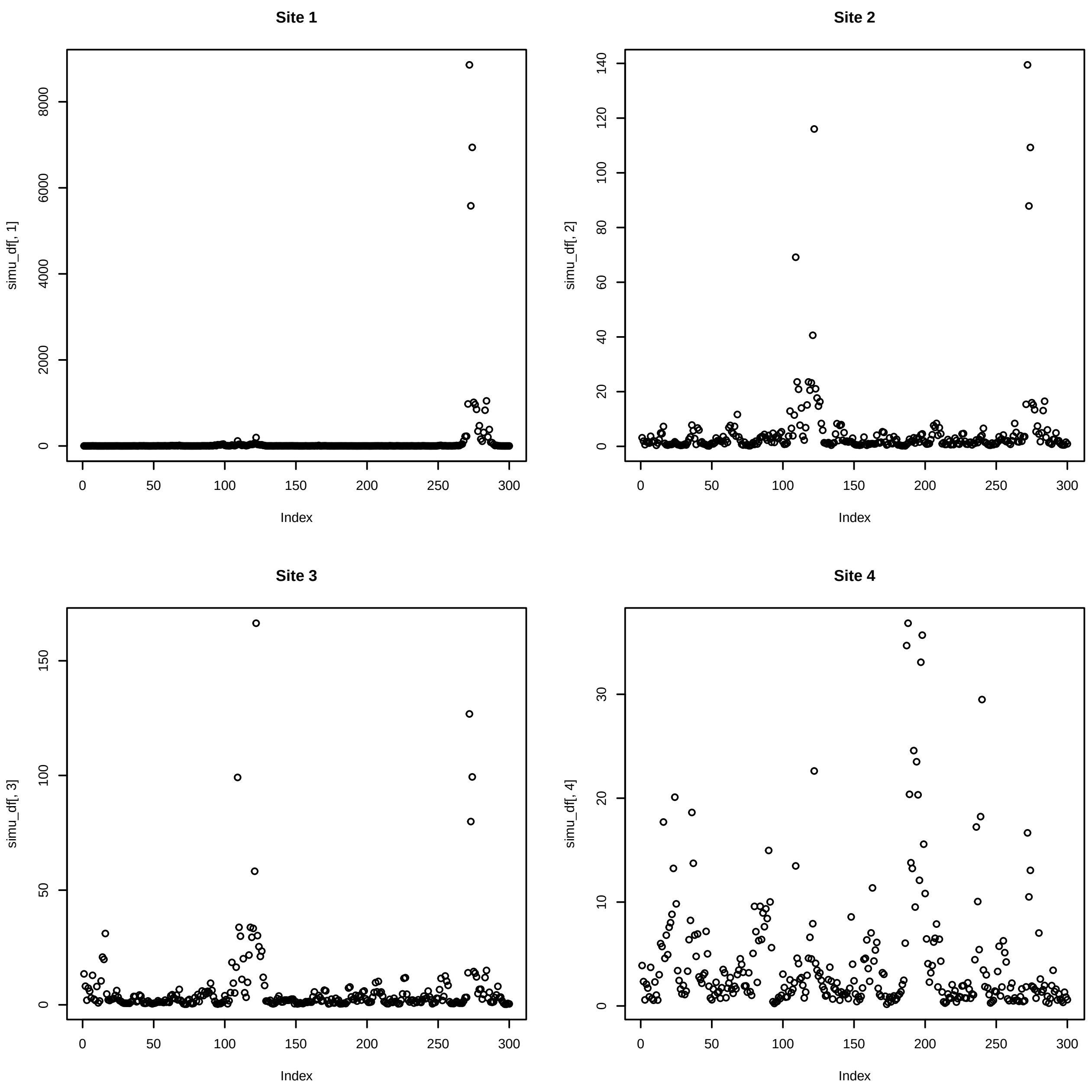
Debug optimisation with neg ll

2024-09-10

Simulation

```r
adv <- c(0, 0)
true_param <- c(0.4, 0.2, 1.5, 1) # ok verif sur simu
ngrid <- 5
spa <- 1:ngrid
nsites <- ngrid^2 # if the grid is squared
temp <- 1:300

# library(doParallel)
# library(foreach)

# num_iter <- 1

# detect core
# cl <- makeCluster(detectCores())
# registerDoParallel(cl)

# results <- foreach(i = 1:num_iter, .combine = rbind) %dopar% {
#   library(generain)
#   BR <- sim_BR(true_param[1], true_param[2], true_param[3],
#                  true_param[4], spa, spa, temp, 1, adv)

#   if (all(adv == c(0, 0))) {
#     foldername <- paste0("../data/simulations_BR/sim_", ngrid^2, "s_",
#                                  length(temp), "t/")
#   } else {
#     foldername <- paste0("../data/simulations_BR/sim_", ngrid^2, "s_",
#                                    length(temp), "t_adv/")
#   }

#   if (!dir.exists(foldername)) {
#     dir.create(foldername, recursive = TRUE)
#   }

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

# }

# stopCluster(cl)
# load the simulations
if (all(adv == c(0, 0))) {
    foldername <- paste0("../data/simulations_BR/sim_", ngrid^2, "s_",
                                length(temp), "t/")
} else {
    foldername <- paste0("../data/simulations_BR/sim_", ngrid^2, "s_",
                                  length(temp), "t_adv/")
}

file_path <- paste0(foldername, "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))
```



Verification que les marges sont des gumbel:

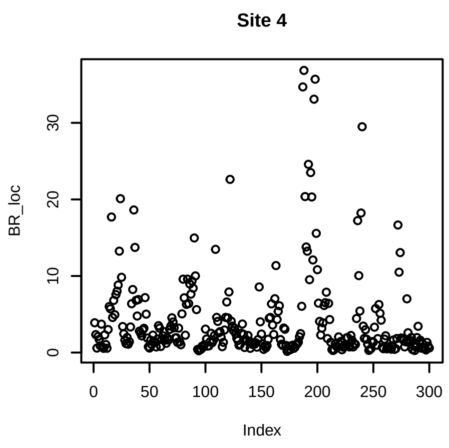
```r
# qq-plots margins
par(mfrow = c(2, 2), cex = 0.5)

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

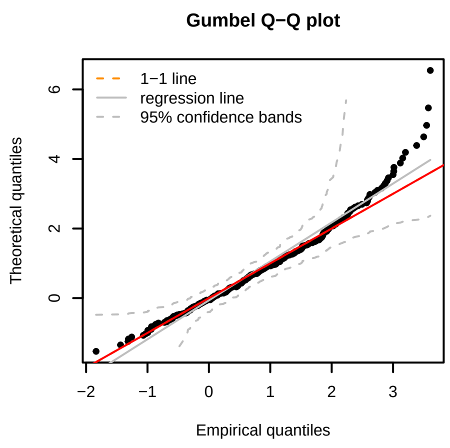
```
## $conv
## [1] 0
##
## $nllh
## [1] 456.0032
##
## $mle
## [1] 0.2863054 0.9784276
##
## $se
## [1] 0.05975817 0.04252630
```

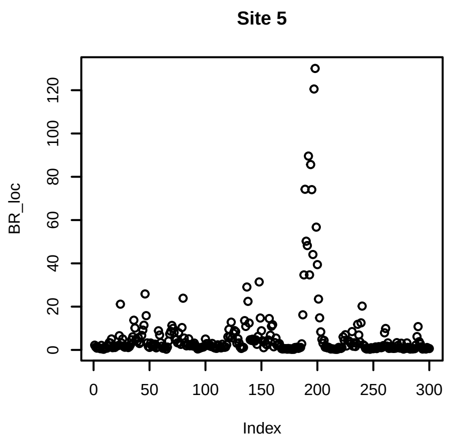
```
mu <- gumbel.fit$mle[1]
sigma <- gumbel.fit$mle[2]

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")
abline(0, 1, col = "red")


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

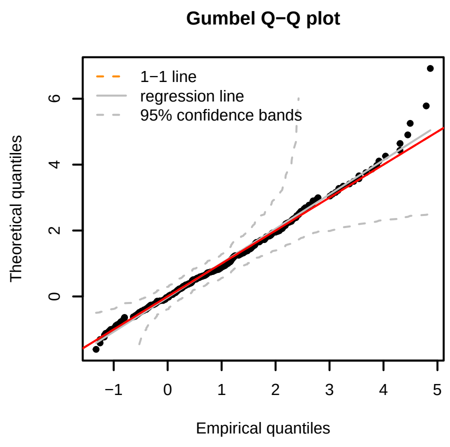
```
## $conv
## [1] 0
##
## $nllh
## [1] 480.268
##
## $mle
## [1] 0.3115968 1.0328937
##
## $se
## [1] 0.06291003 0.04645451
```

```
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")
abline(0, 1, col = "red")
```









Bulh model WLSE

Validation du modèle de Buhl avec WLSE, cela ne marche pas bien pour toutes les simus, pq?

```
# get the distances
dist_mat <- get_dist_mat(sites_coords,
                         latlon = FALSE) # distance matrix
df_dist <- reshape_distances(dist_mat) # reshape the distance matrix

hmax <- sqrt(17)
q <- 0.7
chispa <- spatial_chi_alldist(df_dist, simu_df, quantile = q,
                               hmax = hmax)
spa_estim <- get_estimate_variospa(chispa, weights = "exp", summary = F)
print(spa_estim)
```

```
## [1] 0.4552992 1.7684433
```

```
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),
```

Table 1: Set of lags for site 1 with tau = 2 and hnorm <= 2

s1	s2	h1	h2	tau	hnorm
1	2	0	1	2	1.000000
1	3	0	2	2	2.000000
1	6	1	0	2	1.000000
1	7	1	1	2	1.414214
1	11	2	0	2	2.000000
1	1	0	0	2	0.000000

```
                                       weights = "exp", summary = FALSE)
print(temp_estim)
```

```
## [1] 0.2300098 0.8520833
```

```
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")

kable(df_valid, format = "latex") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed",
  "responsive"), latex_options = "H")
```

	estim	rmse	mae
beta1	0.4552992	0.0552992	0.0552992
beta2	0.2300098	0.0300098	0.0300098
alpha1	1.7684433	0.2684433	0.2684433
alpha2	0.8520833	0.1479167	0.1479167

Set of lags

Calcul des exces marginaux et joints

Fonction qui calcule les excès joints ie les $k_{ij,\tau}$ pour chaque paire de sites (s_i, s_j) et chaque lag temporel τ.

```
get_marginal_excess <- function(data_rain, quantile) {
  Tmax <- nrow(data_rain)
  rain_unif <- rank(data_rain[, 1]) / (Tmax + 1) # for one sites
  marginal_excesses <- sum(rain_unif > quantile)
  return(marginal_excesses)
}

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 tau 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 <- as.data.frame(na.omit(rain_cp))
      colnames(rain_cp) <- c("s1", "s2")

      Tmax <- nrow(rain_cp) # number of total observations
      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

      Tobs <- length(rain_nolag) # number of observations for the lagged pair
                                 # i.e. T - tau

      # transform the data in uniform data
      rain_unif <- cbind(rank(rain_nolag) / (Tobs + 1),
                         rank(rain_lag) / (Tobs + 1))

      # get the conditional excesses on s2
      cp_cond <- rain_unif[rain_unif[, 2] > quantile, ]
      # number of joint excesses
      joint_excesses <- sum(cp_cond[, 1] > quantile)

      # store the number of excesses and T - tau
      excesses$Tobs[excesses$s1 == ind_s1
                    & excesses$s2 == ind_s2
                    & excesses$tau == t] <- Tobs

      excesses$kij[excesses$s1 == ind_s1
                  & excesses$s2 == ind_s2
                  & excesses$tau == t] <- joint_excesses
    }
  }
  return(excesses)
}
```

```
##   s1 s2 h1 h2 tau hnorm Tobs kij
## 1  1  2  0  1   0     1  300  15
## 2  1  2  0  1   1     1  299  14
## 3  1  2  0  1   2     1  298  13
## 4  1  2  0  1   3     1  297  13
## 5  1  2  0  1   4     1  296  13
## 6  1  2  0  1   5     1  295  11
```

Verification

For a pair of sites (s_1, s_2) and a temporal lag $\tau = 2$, we have:

```
## [1] "Number of marginal excesses:  60"
```

```
## [1] "Number of joint excesses:  31"
```

Avec la fonction `get_marginal_excess` et la fonction `empirical_excesses` on retrouve bien les bons résultats:

```
# get the number of marginal excesses
n_marg1 <- sum(rain_cp$s2 > rain_cp_q)
n_marg2 <- get_marginal_excess(rain_cp, q)
print(n_marg1 == n_marg2) # ok
```

```
## [1] TRUE
```

```
q <- 0.8
excesses <- empirical_excesses(simu_df, quantile = q, df_lags = df_lags)
excesses_s1_s2 <- excesses[excesses$s1 == 1 & excesses$s2 == 2, ]
print(excesses_s1_s2)
```

```
##    s1 s2 h1 h2 tau hnorm Tobs kij
## 1   1  2  0  1   0     1  300  37
## 2   1  2  0  1   1     1  299  34
## 3   1  2  0  1   2     1  298  31
## 4   1  2  0  1   3     1  297  32
## 5   1  2  0  1   4     1  296  32
## 6   1  2  0  1   5     1  295  31
## 7   1  2  0  1   6     1  294  29
## 8   1  2  0  1   7     1  293  27
## 9   1  2  0  1   8     1  292  27
## 10  1  2  0  1   9     1  291  25
## 11  1  2  0  1  10     1  290  26
```

Pour un meme site, on doit avoir le meme nombre de dépassements marginale et conjoint sans décalage temporel. Prenons le site s_1 avec lui meme et un décalage temporel $\tau = 0$. On a bien le meme nombre de dépassements marginal et joints:

```
## [1] "Number of marginal excesses:  60"
```

```
## [1] "Number of joint excesses:  60"
```

Calcul du chi

Theorical chi

Pour le calcul du chi théorique, on utilise la formule suivante: $\chi(h, \tau) = 2 - 2\Phi(\sqrt{0.5\gamma(h, \tau)})$ où Φ est la fonction de répartition de la loi normale et $\gamma(h, \tau) = 2(\beta_1 ||h||^{\alpha_1} + \beta_2 \tau^{\alpha_2})$.

```r
theorical_chi_ind <- function(params, h, tau) {
  # get variogram parameter
  beta1 <- params[1]
  beta2 <- params[2]
  alpha1 <- params[3]
  alpha2 <- params[4]

  # Get vario and chi for each lagtemp
  varioval <- 2 * (beta1 * h^alpha1 + beta2 * tau^alpha2)
  phi <- pnorm(sqrt(0.5 * varioval))
  chival <- 2 * (1 - phi)

  return(chival)
}


theorical_chi <- function(params, df_lags) {
  chi_df <- df_lags # copy the dataframe
  chi_df$chi <- theorical_chi_ind(params, df_lags$hnorm, df_lags$tau)
  return(chi_df)
}
```

```r
tau_vect <- 0:10
df_lags <- get_lag_vectors(sites_coords, true_param,
                           hmax = sqrt(17), tau_vect = tau_vect)
chi_theorical <- theorical_chi(true_param, df_lags)
print(tail(chi_theorical))
```

```
##      s1 s2 h1 h2 tau hnorm       chi
## 3240 25 25  0  0   5     0 0.3173105
## 3241 25 25  0  0   6     0 0.2733217
## 3242 25 25  0  0   7     0 0.2367236
## 3243 25 25  0  0   8     0 0.2059032
## 3244 25 25  0  0   9     0 0.1797125
## 3245 25 25  0  0  10     0 0.1572992
```

```r
true_param <- c(0.4, 0.2, 1.5, 1)
# for one hnorm and one tau
hnorm <- 1
tau <- 3
semivar <- true_param[1]*hnorm^true_param[3] + true_param[2]*tau^true_param[4]
chi_h_t_verif <- 2 * (1 - pnorm(sqrt(semivar)))

chi_h_t <- chi_theorical$chi[chi_theorical$hnorm == hnorm &
                                   chi_theorical$tau == tau]

# print(chi_h_t) # all the same proba: good
print(unique(chi_h_t) == chi_h_t_verif) # ok
```

```
## [1] TRUE
```

Empirical chi

Pour le chi empirique je le calcule de deux facons:

- soit en comptant le nombre d'exces marginal et le nombre d'exces joints et je prends le ratio du nombre d'exces joints sur le nombre d'exces marginaux pour chaque paire de sites et lag temporel.
- soit en utilisant la fonction `get_chiq` qui estime le chi de avec la formule $2 - \frac{\log(c_u)}{\log(u)}$ où c_u est la proportion de couples de sites dont le maximum des rangs est inferieur à u.

Pour les deux facons de faire je n'obtiens pas les memes valeurs de chi empiriques et la seconde methode que j'utilise pour le modèle avec régression donne de meilleures correspondances avec les chi théoriques que l'autre méthode.

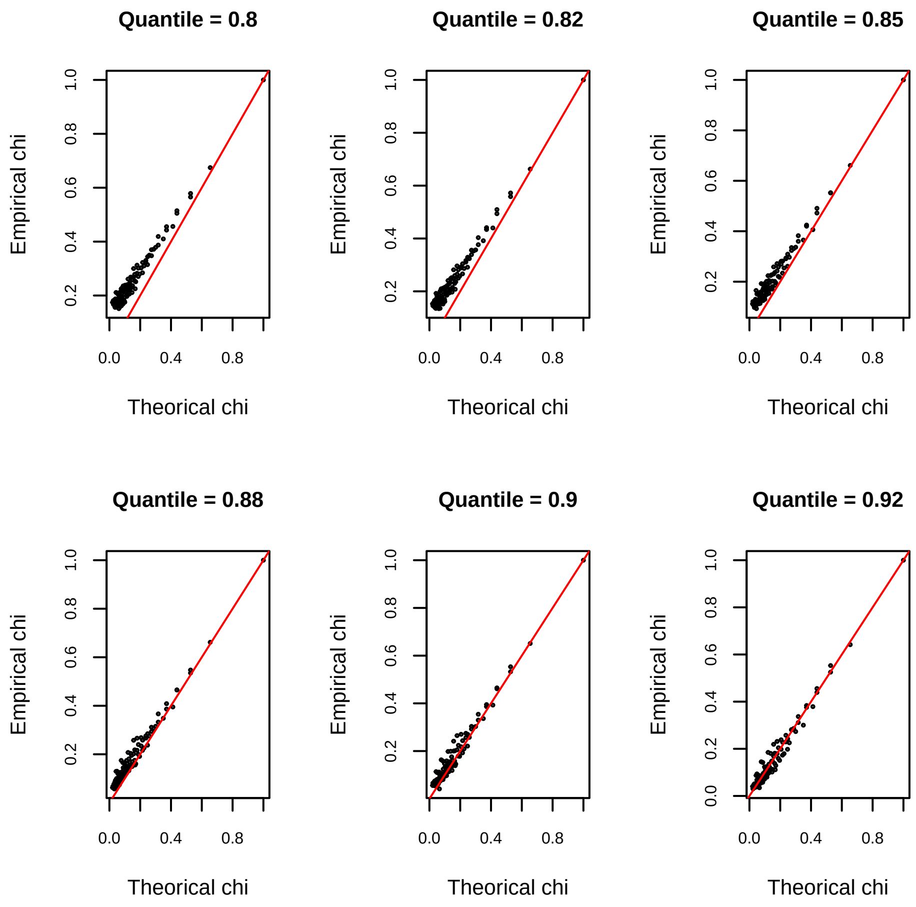
```r
# get chi empirical for a couple of sites in data and a quantile
get_chiq <- function(data, quantile) {
  n <- nrow(data)
  # Transform data into uniform ranks
  data_unif <- cbind(rank(data[, 1]) / (n + 1),
                     rank(data[, 2]) / (n + 1))
  # Calculate the maximum rank for each row
  rowmax <- apply(data_unif, 1, max)
  # Calculate the chi value
  u <- quantile
  cu <- mean(rowmax < u)
  chiu <- 2 - log(cu) / log(u)
  # Calculate the lower bound for chi
  chiulb <- 2 - log(pmax(2 * u - 1, 0)) / log(u)
  # Take the maximum of chi and chiulb
  chiu <- pmax(chiu, chiulb)
  return(chiu)
}

# Empirical spatio-temporal chi
chispatemp_dt <- function(data_rain, df_lags, quantile) {
  chi_st <- df_lags
  chi_st$chiemp <- NA
  chi_st$chiemp2 <- NA
  Tmax <- nrow(data_rain)
  for (t in unique(df_lags$tau)) {
    df_h_t <- df_lags[df_lags$tau == t, ] # get the dataframe for each lag
    for (i in seq_len(nrow(df_h_t))) { # loop over each pair of sites
      # get index pairs
      ind_s1 <- df_h_t$s1[i]
      ind_s2 <- as.numeric(as.character(df_h_t$s2[i]))
      rain_cp <- na.omit(data_rain[, c(ind_s1, ind_s2)])
      colnames(rain_cp) <- c("s1", "s2")
      rain_lag <- rain_cp$s1[(t + 1):Tmax]
      rain_nolag <- rain_cp$s2[1:(Tmax - t)]
      data <- cbind(rain_lag, rain_nolag) # get couple
      Tobs <- nrow(data) # T - tau
      data_unif <- cbind(rank(data[, 2]) / (Tobs + 1),
                         rank(data[, 1]) / (Tobs + 1))
      # get the number of marginal excesses
      n_marg <- sum(data_unif[, 1] > quantile)

      # get the number of joint excesses
      cp_cond <- data_unif[data_unif[, 1] > quantile,]
      joint_excesses <- sum(cp_cond[, 2] > quantile)
      chi_hat_h_tau <- joint_excesses / n_marg
      chi_st$chiemp[chi_st$s1 == ind_s1 & chi_st$s2 == ind_s2 &
                    chi_st$tau == t] <- chi_hat_h_tau
      chi_val <- get_chiq(data_unif, quantile)
      chi_st$chiemp2[chi_st$s1 == ind_s1 & chi_st$s2 == ind_s2 &
                      chi_st$tau == t] <- chi_val
    }
  }
  # if chi <= 0 then set it to 0.000001
  chi_st$chiemp <- ifelse(chi_st$chiemp <= 0, 0.000001, chi_st$chiemp)
  chi_st$chiemp2 <- ifelse(chi_st$chiemp2 <= 0, 0.000001, chi_st$chiemp2)
  return(chi_st)
}


chi_st <- chispatemp_dt(simu_df, df_lags, 0.9)
print(head(chi_st))
```

```
##   s1 s2 h1 h2 tau hnorm    chiemp   chiemp2
## 1  1  2  0  1   0     1 0.5000000 0.4574968
## 2  1  2  0  1   1     1 0.4827586 0.4518934
## 3  1  2  0  1   2     1 0.4137931 0.4086600
## 4  1  2  0  1   3     1 0.3793103 0.3649357
## 5  1  2  0  1   4     1 0.4482759 0.4348357
## 6  1  2  0  1   5     1 0.4137931 0.3910248
```

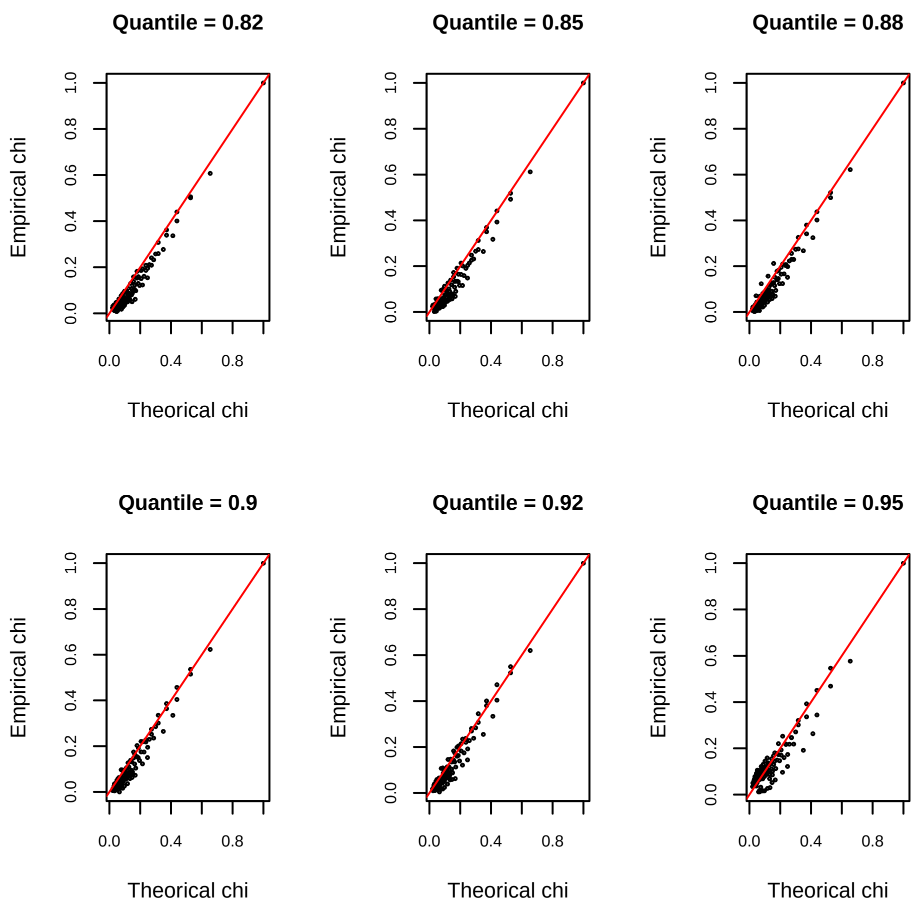
Empirique vs Theorique

Pour calculer le chi spatio-temporel empirique, on doit avoir un chi proche du chi theorique en moyennant les chi empiriques sur les paires de sites avec le meme lag temporel et la meme lag spatiale. On obtient des valeurs semblables empiriquement et théoriquement:

Premiere methode de calcul du chi empirique:



Deuxieme methode de calcul du chi empirique:



Cela ne donne pas de bons graphiques pour toutes les simu… pq???

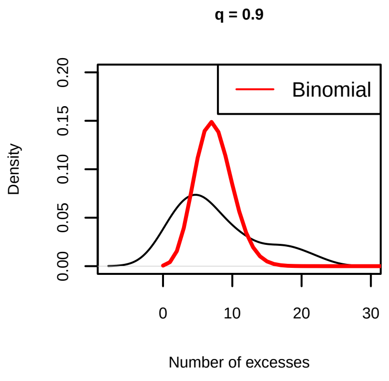
Distribution des dépassements

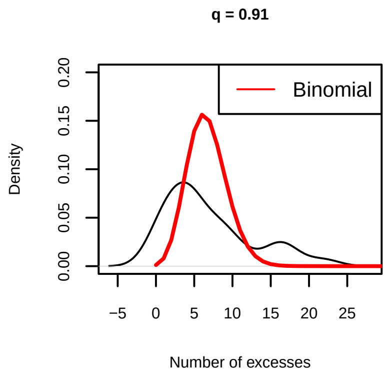
Pour chaque paire de sites dans le même lag spatial on a une variable $k_{h,\tau} = [k_{ij,\tau}, (i,j)|s_i - s_j = h]$ qui suit une distribution binomiale de paramètres $T - \tau$ et $p \times \chi_{\tau,h}$ où T est le nombre d'observations, τ le lag temporel, p la probabilité d'excès marginaux et $\chi_{\tau,h}$ est le chi théorique pour le lag temporel τ et le lag spatial h.

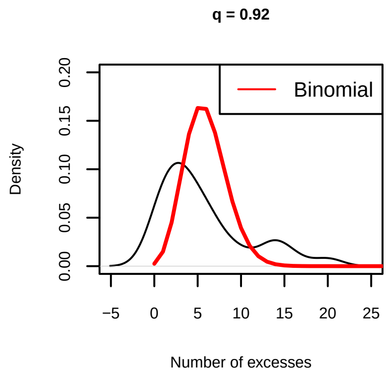
Pour différentes valeurs de quantiles, on peut vérifier la distribution des dépassements joints $k_{ij,\tau}$ par rapport à la distribution binomiale correspondante théoriquement. On se fixe un lag spatial $h = 2$, un lag temporel $\tau = 1$ et on fait varier le quantile:

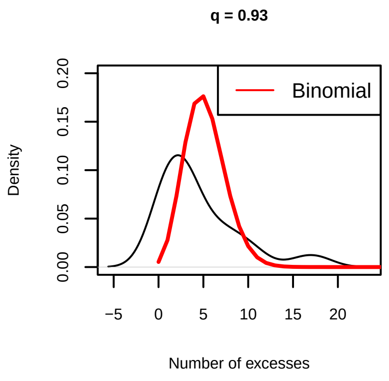
```
q_values <- seq(0.9, 0.96, by = 0.01)
df_lags <- get_lag_vectors(sites_coords, true_param,
                           hmax = sqrt(17), tau_vect = 0:10)
par(mfrow = c(ceiling(length(q_values)/3), 3))
for (q in q_values) {
  excesses <- empirical_excesses(simu_df, quantile = q, df_lags = df_lags)
  n_marg <- get_marginal_excess(simu_df, quantile = q)
  Tobs <- excesses$Tobs # T - tau
  Tmax <- nrow(simu_df)
  p_hat <- n_marg / Tmax # probability of marginal excesses
  kij <- excesses$kij # number of joint excesses
```

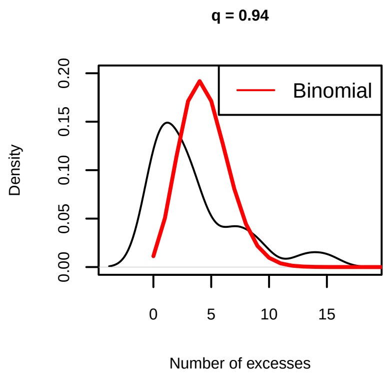
```r
  chi_theorical <- theorical_chi(true_param, df_lags)

  tau <- 1
  hnorm <- 2
  chi_tau_h <- chi_theorical$chi[chi_theorical$tau == tau &
                chi_theorical$hnorm == hnorm]
  k_tau_h <- excesses$kij[excesses$tau == tau &
           excesses$hnorm == hnorm]
  proba_tau_h <- unique(chi_tau_h * p_hat)
  n <- Tmax - tau # t - tau

  Tobs_tau_h <- unique(Tobs[excesses$tau == tau &
              excesses$hnorm == hnorm])
  x <- 0:Tobs_tau_h
  # Density
  plot(density(k_tau_h), main = paste("q =",
                    q), xlab = "Number of excesses",
                    ylim = c(0, 0.2), cex.main = 0.8,
                    cex.lab = 0.8, cex.axis = 0.8)
  # binomial density
  lines(x, dbinom(x, size = Tobs_tau_h, prob = proba_tau_h), col = "red",
    lwd = 2)
  legend("topright", legend = "Binomial", col = "red", lwd = 1)
}
```

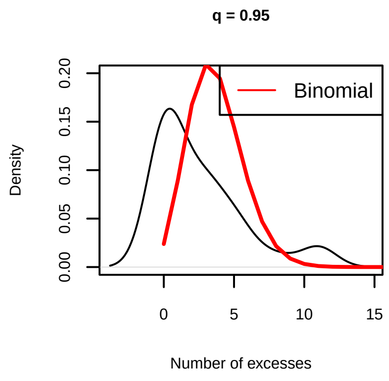


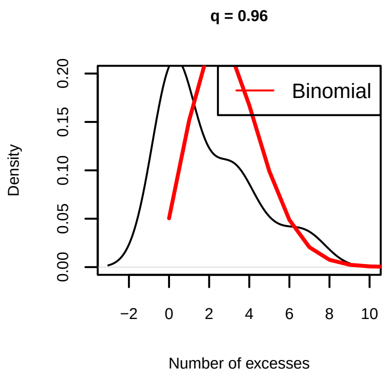




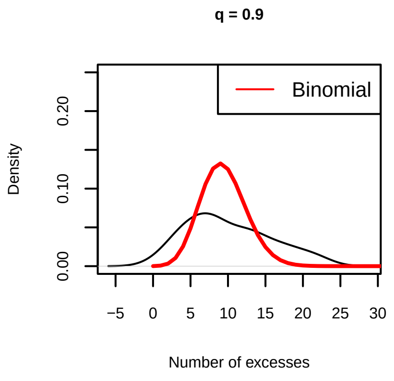


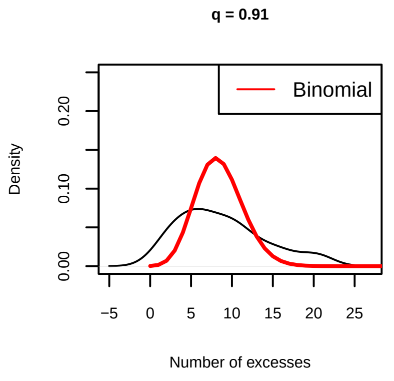


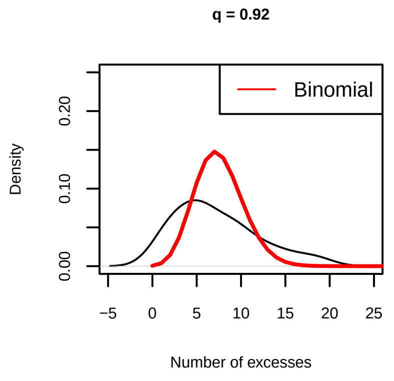


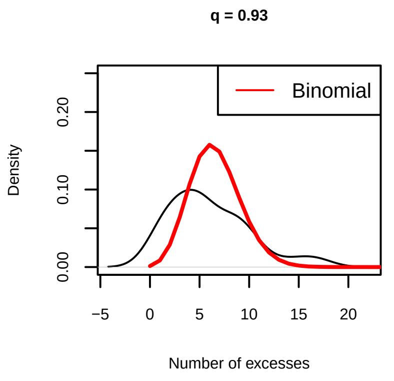


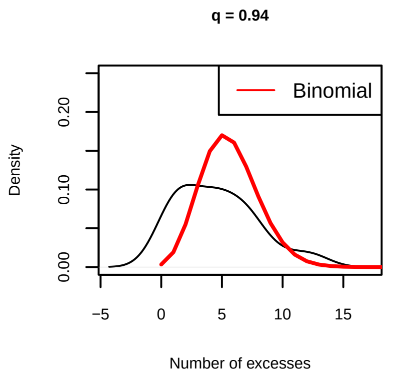
Pour un autre lag spatial $h = 1$ et un lag temporel $\tau = 3$:

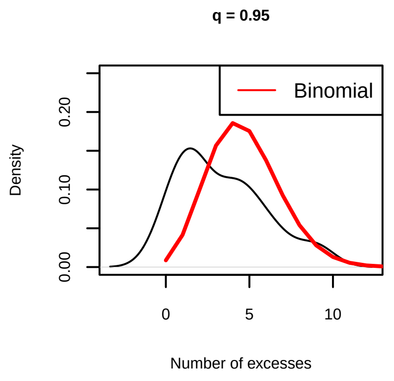


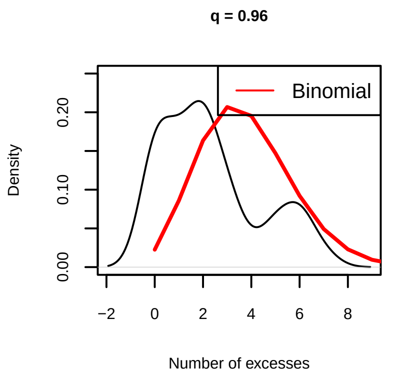




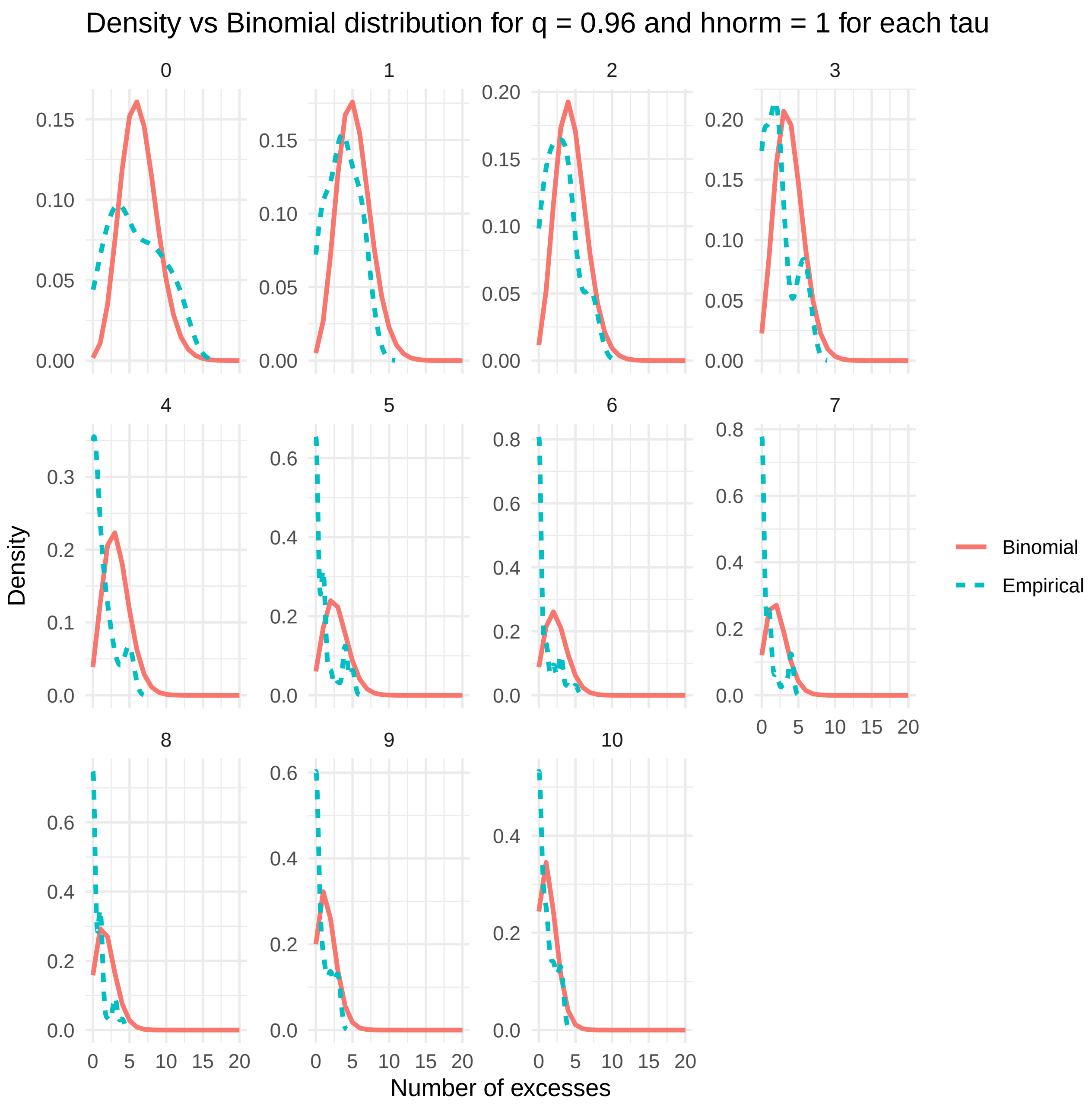




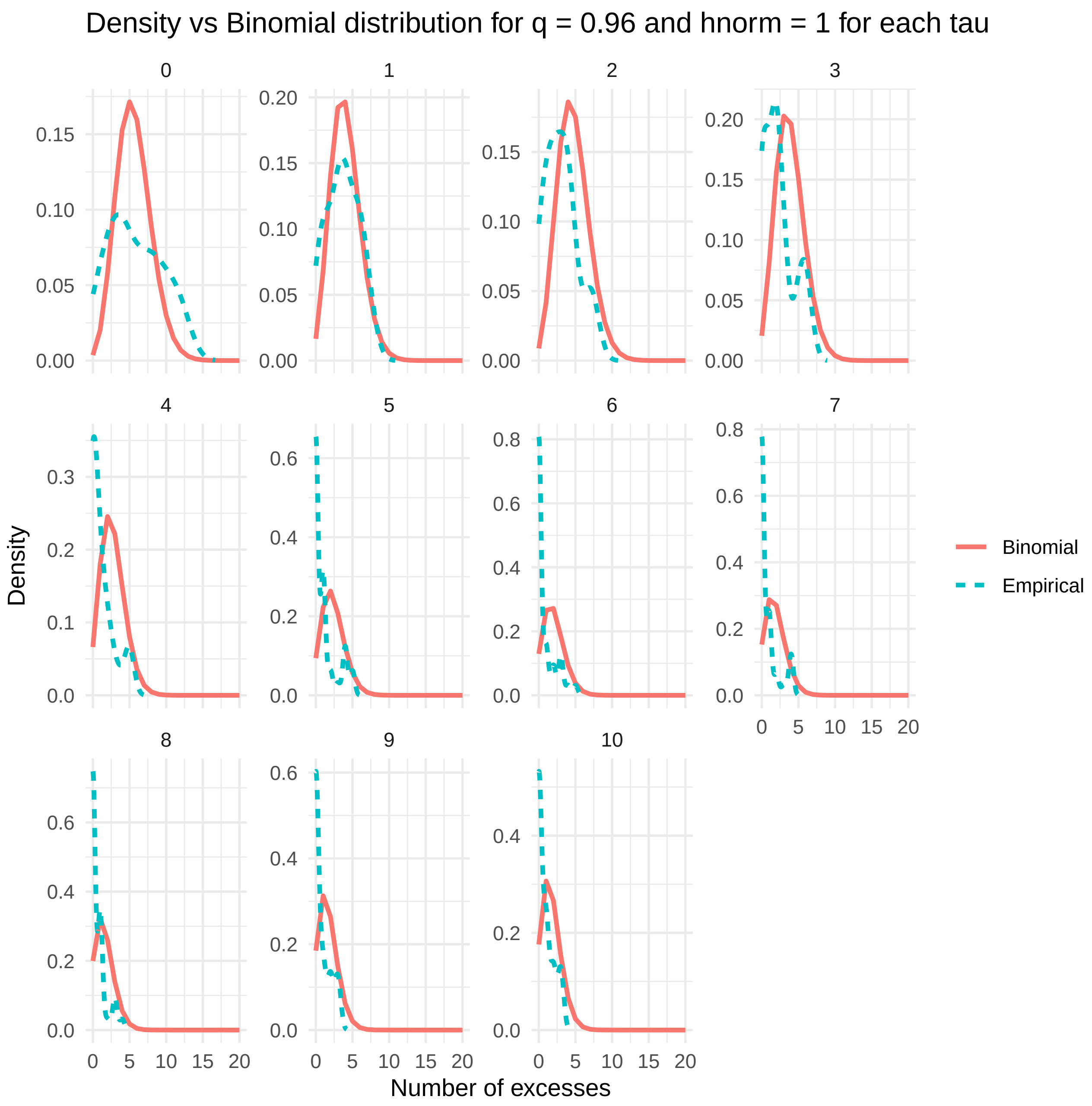




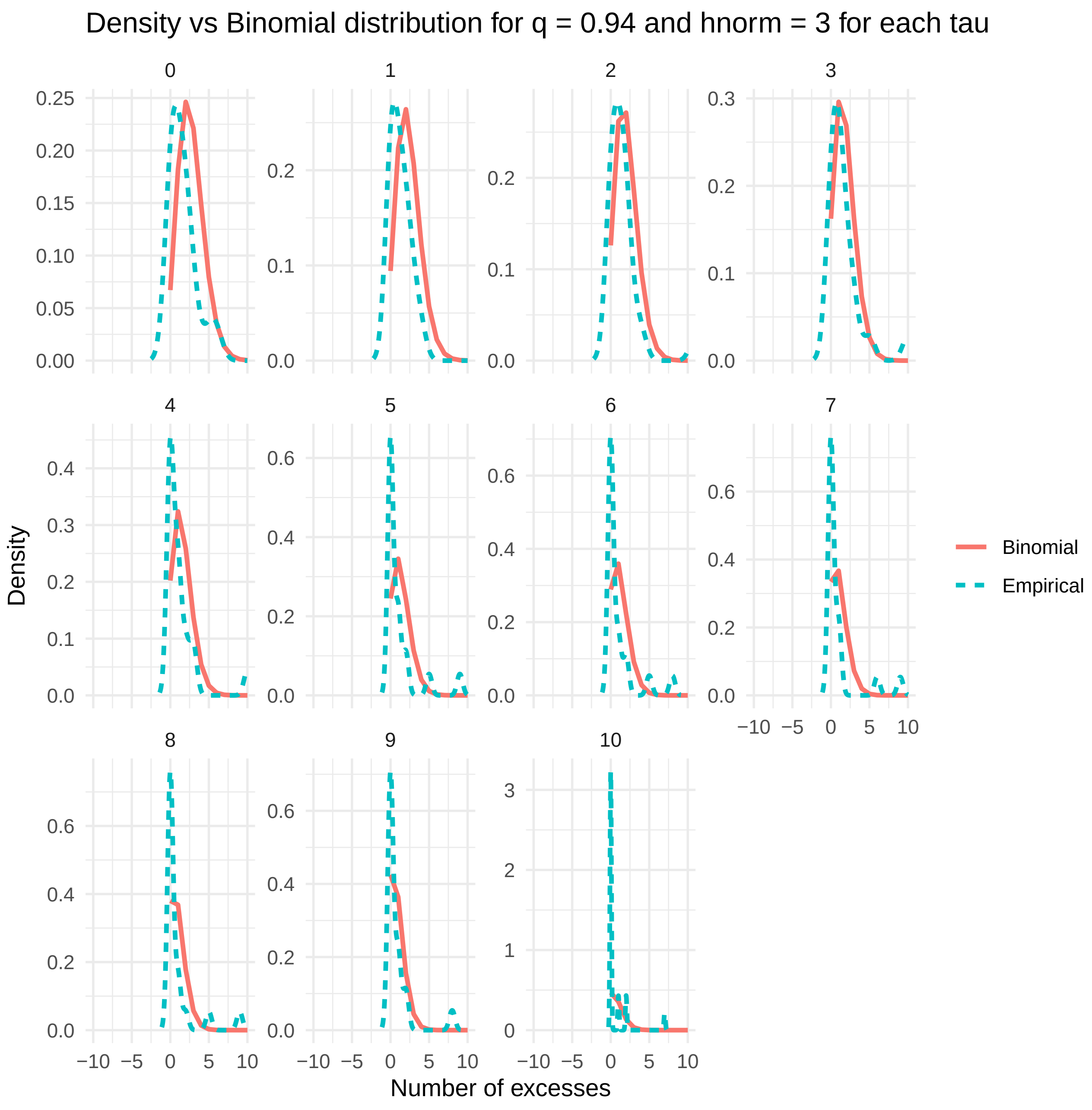
Maintenant on fixe le quantile et $h = 1$ et on fait varier le lag temporel τ, en prenant le chi théorique:



Idem en prenant le chi empirique moyen, c'est la meme chose qu'avec le theorique (à peu près):



Ici on fixe le quantile et $h = 3$ et on fait varier le lag temporel τ:



Optimisation

Log likelihood

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

  # print(params)
  if (length(params) == 6) {
    adv <- params[5:6]
  } else {
    adv <- c(0, 0)
  }

  # Bounds for the parameters
  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, -Inf, -Inf)
    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")
    return(1e9)
  }

  if (!all(adv == c(0, 0))) { # if we have the advection parameters
    # then the lag vectors are different
    df_lags <- get_lag_vectors(locations, params, hmax = hmax, tau_vect = tau)
  }

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

  n_marg <- get_marginal_excess(simu, quantile) # number of marginal excesses
  Tobs <- excesses$Tobs # T - tau
  p <- n_marg / nrow(simu) # probability of marginal excesses
  kij <- excesses$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 <- Tobs - kij # number of non-excesses
  # log-likelihood vector
  ll_vect <- kij * log(chi_vect) + non_excesses * log(1 - p * chi_vect)

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

Quelques résultats pour la log-likelihood:

```
q <- 0.94
nll <- neg_ll(true_param, simu_df, df_lags, sites_coords, quantile = q)
print(nll)
```

```
## [1] 19703.89
```

```
nll <- neg_ll(true_param + 0.05, simu_df, df_lags, sites_coords, quantile = q)
print(nll)
```

```
## [1] 19485.76
```

```
nll <- neg_ll(true_param - 0.1, simu_df, df_lags, sites_coords, quantile = q)
print(nll)
```

```
## [1] 22841.77
```

Si je réduis le nombre de décalage temporel:

For different sets of temporal lags 0 à tmax, we have the following RMSE for each parameter with optimisation:

Table 2: RMSE for each parameter and different sets of temporal lags 0:tmax

tmax	rmse_beta1	rmse_beta2	rmse_alpha1	rmse_alpha2
1	0.0170942	0.0083991	0.0572326	0.0000000
2	0.0349152	0.0100533	0.0027265	0.0158436
4	0.0561065	0.0101326	0.0603182	0.0344390
6	0.0749351	0.0182622	0.1137397	0.0151051
8	0.0879547	0.0092526	0.1496050	0.0230896
10	0.0856227	0.0308988	0.1546388	0.1612043

Optimisation on simulations

For one simulation

Pour differentes valeurs de quantiles, on peut estimer les paramètres du modèle spatio-temporel avec l'optimisation. On peut aussi calculer le RMSE pour chaque paramètre.

C'est beaucoup trop sensible au quantile.

On obtient les résultats suivants pour une simulation:

Table 3: Optim estimations for each quantile for one simulation

q	beta1	beta2	alpha1	alpha2
0.900	0.45243	0.29138	1.29649	0.64426
0.905	0.45539	0.24091	1.32656	0.76487
0.910	0.47558	0.29135	1.32440	0.68645
0.915	0.48562	0.23090	1.34536	0.83880
0.920	0.50939	0.30663	1.33900	0.71178
0.925	0.54612	0.25171	1.31788	0.81653
0.930	0.56100	0.33743	1.33296	0.70491
0.935	0.57452	0.27537	1.33093	0.81747
0.940	0.59898	0.38868	1.33852	0.69544
0.945	0.65143	0.35567	1.26113	0.75114
0.950	0.70409	0.42975	1.26025	0.73279
0.955	0.73536	0.38932	1.22551	0.77967
0.960	0.68609	0.56722	1.30528	0.74189

Table 4: RMSE for each parameter and different quantiles for one simulation

q	rmse_beta1	rmse_beta2	rmse_alpha1	rmse_alpha2
0.900	0.05243	0.09138	0.20351	0.35574
0.905	0.05539	0.04091	0.17344	0.23513
0.910	0.07558	0.09135	0.17560	0.31355
0.915	0.08562	0.03090	0.15464	0.16120
0.920	0.10939	0.10663	0.16100	0.28822
0.925	0.14612	0.05171	0.18212	0.18347
0.930	0.16100	0.13743	0.16704	0.29509
0.935	0.17452	0.07537	0.16907	0.18253
0.940	0.19898	0.18868	0.16148	0.30456
0.945	0.25143	0.15567	0.23887	0.24886
0.950	0.30409	0.22975	0.23975	0.26721
0.955	0.33536	0.18932	0.27449	0.22033
0.960	0.28609	0.36722	0.19472	0.25811

Optimisation en fixant des parametres

Fixer trois paramètres

Table 5: Optim estimations for each quantile, true beta1 = 0.4

Quantile	beta1_estim	Convergence
0.900	0.3371889	0
0.905	0.3514756	0
0.910	0.3774021	0
0.915	0.3968367	0
0.920	0.4281868	0
0.925	0.4452487	0
0.930	0.4874109	0
0.935	0.4932203	0
0.940	0.5512394	0

Table 6: Optim estimations for each quantile, true beta2 = 0.2

Quantile	beta2_estim	Convergence
0.900	0.1469415	0
0.905	0.1564408	0
0.910	0.1691743	0
0.915	0.1834961	0
0.920	0.1984644	0
0.925	0.2065283	0
0.930	0.2297358	0
0.935	0.2348088	0
0.940	0.2698248	0

Table 7: Optim estimations for each quantile, true alpha1 = 1.5

Quantile	alpha1_estim	Convergence
0.900	1.338691	0
0.905	1.378525	0
0.910	1.436291	0
0.915	1.482719	0
0.920	1.543351	0
0.925	1.577543	0
0.930	1.650604	0
0.935	1.665039	0
0.940	1.754201	0
0.945	1.751942	0
0.950	1.870495	0

Table 8: Optim estimations for each quantile, true alpha2 = 1

Quantile	alpha2_estim	Convergence
0.900	0.8287727	0
0.905	0.8651687	0
0.910	0.8983192	0
0.915	0.9457304	0
0.920	0.9759903	0
0.925	0.9997185	0
0.930	1.0442274	0
0.935	1.0612232	0
0.940	1.1200534	0
0.945	1.1368101	0
0.950	1.2258209	0

Fixer deux paramètres

Table 9: Optim estimations for each quantile fixing beta2 and alpha2, true beta1 = 0.4, alpha1 = 1.5

Quantile	beta1_estim	alpha1_estim	Convergence
0.80	0.1800755	1.397896	0
0.81	0.1970962	1.368019	0
0.82	0.2134234	1.352602	0
0.83	0.2354265	1.340357	0
0.84	0.2533175	1.361911	0
0.85	0.2739199	1.361728	0
0.86	0.2987673	1.333629	0
0.87	0.3134615	1.350305	0
0.88	0.3402808	1.322212	0
0.89	0.3408630	1.391386	0
0.90	0.3696627	1.408907	0
0.91	0.4199788	1.392800	0
0.92	0.4950725	1.352000	0
0.93	0.5912940	1.298140	0
0.94	0.7065171	1.237039	0
0.95	0.9303811	1.097223	0

Table 10: Optim estimations for each quantile fixing beta1 and alpha1, true beta2 = 0.2, alpha2 = 1

Quantile	beta2_estim	alpha2_estim	Convergence
0.900	0.2675763	0.6814943	0
0.905	0.2346126	0.7844355	0
0.910	0.3068363	0.6822130	0
0.915	0.2642360	0.8049865	0
0.920	0.3703710	0.6646203	0
0.925	0.3429890	0.7269879	0
0.930	0.4643225	0.6188686	0
0.935	0.4080128	0.7005012	0
0.940	0.5715546	0.5895952	0
0.945	0.5507963	0.6256810	0
0.950	0.7002265	0.5901718	0

Fixer un paramètre

Table 11: Optim estimations for each fixing alpha2, true beta1 = 0.4, beta2 = 0.2, alpha1 = 1.5

Quantile	beta1_estim	beta2_estim	alpha1_estim	Convergence
0.900	0.5140909	0.1428237	1.218511	0
0.905	0.4900538	0.1521837	1.281666	0
0.910	0.5311266	0.1561779	1.256209	0
0.915	0.5089571	0.1697536	1.315886	0
0.920	0.5638164	0.1734403	1.275787	0
0.925	0.5751711	0.1768677	1.285736	0
0.930	0.6220802	0.1884074	1.268358	0
0.935	0.6068650	0.1938219	1.296426	0
0.940	0.6708453	0.2131787	1.267448	0
0.945	0.7077869	0.2187294	1.210525	0
0.950	0.7762281	0.2553941	1.200448	0

Table 12: Optim estimations for each quantile, fixing alpha1, true beta1 = 0.4, beta2 = 0.2, alpha2 = 1

Quantile	beta1_estim	beta2_estim	alpha2_estim	Convergence
0.900	0.3566699	0.3283365	0.6217075	0
0.905	0.3723821	0.2696375	0.7406553	0
0.910	0.3880566	0.3234071	0.6665173	0
0.915	0.4067302	0.2559580	0.8153244	0
0.920	0.4236098	0.3370750	0.6931952	0
0.925	0.4444789	0.2847561	0.7863250	0
0.930	0.4646492	0.3711187	0.6857019	0
0.935	0.4754750	0.3070472	0.7911947	0
0.940	0.5002951	0.4230007	0.6786296	0
0.945	0.4996623	0.4077268	0.7166022	0
0.950	0.5417321	0.4864968	0.7008737	0

Table 13: Optim estimations for each quantile, fixing beta2, true beta1 = 0.4, alpha1 = 1.5, alpha2 = 1

Quantile	beta1_estim	alpha1_estim	alpha2_estim	Convergence
0.900	0.4929072	1.245014	0.8257884	0
0.905	0.4713408	1.305676	0.8612238	0
0.910	0.5185984	1.271635	0.8670701	0
0.915	0.4969204	1.331331	0.9153768	0
0.920	0.5615108	1.279388	0.9158286	0
0.925	0.5694098	1.292503	0.9320643	0
0.930	0.6319000	1.259818	0.9516662	0
0.935	0.6122747	1.292128	0.9736681	0
0.940	0.7014780	1.241164	1.0055867	0
0.945	0.7405372	1.184057	1.0209639	0
0.950	0.8401946	1.154043	1.0900050	0

Table 14: Optim estimations for each quantile, fixing beta1, true beta2 = 0.2, alpha1 = 1.5, alpha2 = 1

Quantile	beta2_estim	alpha1_estim	alpha2_estim	Convergence
0.900	0.3126412	1.390181	0.6372515	0
0.905	0.2622712	1.425956	0.7505125	0
0.910	0.3247985	1.457306	0.6658266	0
0.915	0.2665175	1.495562	0.8017795	0
0.920	0.3584286	1.526752	0.6741896	0
0.925	0.3170814	1.561163	0.7510911	0
0.930	0.4175382	1.598160	0.6492516	0
0.935	0.3558026	1.616913	0.7424326	0
0.940	0.4915876	1.657754	0.6322536	0
0.945	0.4788735	1.645690	0.6662792	0
0.950	0.5874937	1.710139	0.6413021	0

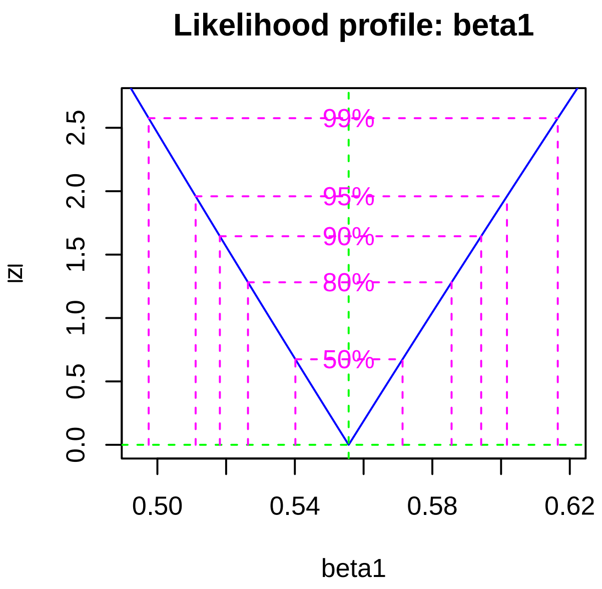
Sans fixer de paramètre

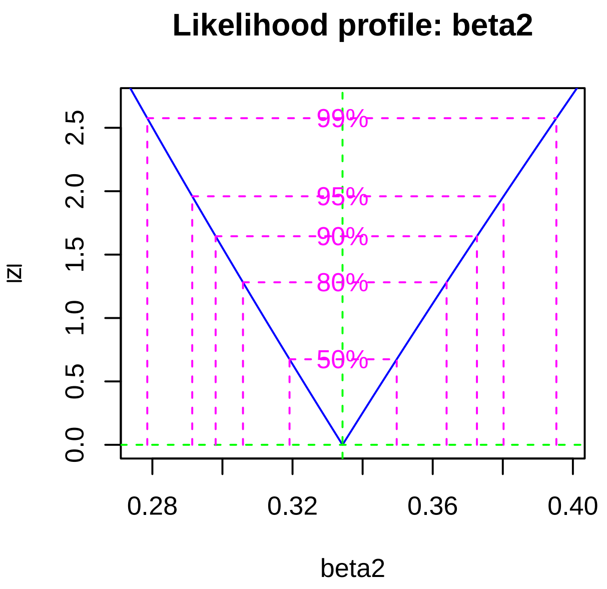
Table 15: Optim estimations for each quantile, true beta1 = 0.4, beta2 = 0.2, alpha1 = 1.5, alpha2 = 1

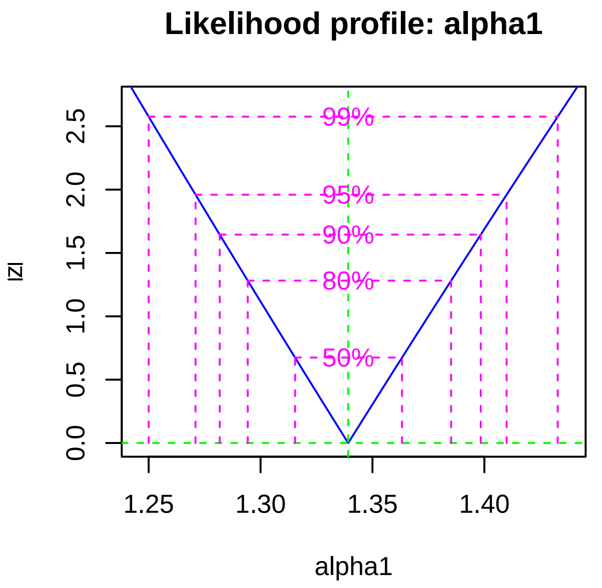
Quantile	beta1_estim	beta2_estim	alpha1_estim	alpha2_estim	Convergence
0.900	0.4475386	0.2873506	1.303464	0.6633962	0
0.905	0.4505773	0.2379404	1.333419	0.7827251	0
0.910	0.4712258	0.2885933	1.330329	0.7023265	0
0.915	0.4806927	0.2283957	1.351903	0.8548750	0
0.920	0.5042393	0.3033848	1.345579	0.7275929	0
0.925	0.5409256	0.2494535	1.324010	0.8313238	0
0.930	0.5556534	0.3342622	1.339162	0.7193928	0
0.935	0.5695209	0.2719346	1.336713	0.8331521	0
0.940	0.5932429	0.3858969	1.344652	0.7078204	0
0.945	0.6454614	0.3532462	1.266960	0.7632285	0
0.950	0.6978880	0.4269431	1.265847	0.7436901	0

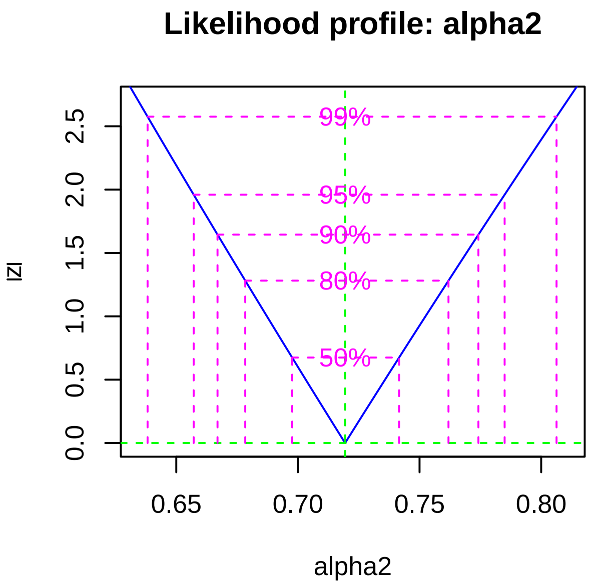
Profiled log likelihood

```r
library(bbmle)
q <- 0.93
excesses <- empirical_excesses(simu_df, quantile = q, df_lags = df_lags)

mle_rain <- mle2(neg_ll_par, start = list(beta1 = true_param[1],
                                 beta2 = true_param[2],
                                 alpha1 = true_param[3],
                                 alpha2 = true_param[4]),
                 data = list(simu = simu_df,
                        quantile = q,
                        df_lags = df_lags,
                        locations = sites_coords,
                        excesses = excesses,
                        method = "CG"),
                  control = list(maxit = 10000))

rain_pro <- bbmle::profile(mle_rain) # profiled likelihood
par(mfrow = c(2, 2))
plot(rain_pro)
```









```
bbmle::confint(rain_pro, level = 0.95) # confidence intervals
```

```
##              2.5 %    97.5 %
## beta1  0.5111321 0.6017176
## beta2  0.2913777 0.3802181
## alpha1 1.2709518 1.4099607
## alpha2 0.6571412 0.7849785
```

Avec advection

```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

foldername <- paste0("../data/simulations_BR/sim_adv_", ngrid^2, "s_",
                                length(temp), "t/")

# load the simulations
file_path <- paste0(foldername, "br_", ngrid^2, "s_", length(temp),
                "t_", 1, ".csv")
simu_df_adv <- read.csv(file_path)
nsites <- ncol(simu_df_adv)
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
df_lags <- get_lag_vectors(sites_coords, true_param,
                          hmax = sqrt(17), tau_vect = 0:10)

print(df_lags[df_lags$tau == 2 & df_lags$s1 == 1, ])
```

```
##      s1 s2   h1    h2 tau     hnorm
## 3     1  2 -0.1  0.96   2 0.9651943
## 14    1  3 -0.1  1.96   2 1.9625494
## 25    1  4 -0.1  2.96   2 2.9616887
## 36    1  5 -0.1  3.96   2 3.9612624
## 47    1  6  0.9 -0.04   2 0.9008885
## 58    1  7  0.9  0.96   2 1.3159027
## 69    1  8  0.9  1.96   2 2.1567568
## 80    1  9  0.9  2.96   2 3.0938003
## 91    1 10  0.9  3.96   2 4.0609851
## 102   1 11  1.9 -0.04   2 1.9004210
## 113   1 12  1.9  0.96   2 2.1287555
## 124   1 13  1.9  1.96   2 2.7297619
## 135   1 14  1.9  2.96   2 3.5173285
## 146   1 16  2.9 -0.04   2 2.9002758
## 157   1 17  2.9  0.96   2 3.0547668
## 168   1 18  2.9  1.96   2 3.5002286
## 179   1 21  3.9 -0.04   2 3.9002051
## 190   1 22  3.9  0.96   2 4.0164163
## 2973  1  1 -0.1 -0.04   2 0.1077033
```

```r
q <- 0.9
excesses <- empirical_excesses(simu_df_adv, quantile = q, df_lags = df_lags)

n_marg <- get_marginal_excess(simu_df_adv, quantile = q)
Tobs <- excesses$Tobs # T - tau
Tmax <- nrow(simu_df_adv)
```

```
p_hat <- n_marg / Tmax # probability of marginal excesses
kij <- excesses$kij # number of joint excesses
```

```
## [1] 60120.27
```

Fixer cinq paramètres

Table 16: Optim estimations for each quantile, true beta1 = 0.4

Quantile	beta1_estim	Convergence
0.920	0.2943196	0
0.925	0.3076608	0
0.930	0.3534190	0
0.935	0.3653126	0
0.940	0.4314644	0
0.945	0.4408757	0
0.950	0.5081529	0

Table 17: Optim estimations for each quantile, true beta2 = 0.2

Quantile	beta2_estim	Convergence
0.90	0.1055638	0
0.91	0.1400286	0
0.92	0.1739979	0
0.93	0.2141604	0
0.94	0.2977664	0
0.95	0.3838401	0

Table 18: Optim estimations for each quantile, true alpha1 = 1.5

Quantile	alpha1_estim	Convergence
0.90	0.9515365	0
0.91	1.0846740	0
0.92	1.1973177	0
0.93	1.3456412	0
0.94	1.4879908	0
0.95	1.6146977	0

Table 19: Optim estimations for each quantile, true alpha2 = 1

Quantile	alpha2_estim	Convergence
0.90	0.6798008	0
0.91	0.8201645	0
0.92	0.9276537	0
0.93	1.0261054	0
0.94	1.1915489	0
0.95	1.3199009	0

Table 20: Optim estimations for each quantile, true adv1 = 0.05

Quantile	adv1_estim	Convergence
0.90	0.1127511	0
0.91	0.0932661	0
0.92	0.0682833	0
0.93	0.0127521	0
0.94	-0.1254595	0
0.95	-0.2364928	0

Table 21: Optim estimations for each quantile, true adv2 = 0.02

Quantile	adv2_estim	Convergence
0.80	0.0020148	0
0.81	0.0007163	0
0.82	-0.0006313	0
0.83	-0.0031163	0
0.84	-0.0051081	0
0.85	-0.0066433	0
0.86	-0.0122297	0
0.87	-0.0169949	0
0.88	-0.0183501	0
0.89	-0.0218259	0
0.90	-0.0278208	0

Estimation de l'advection en fixant les autres paramètres

Table 22: Optim estimations for each quantile, fixing beta1, beta2, alpha1, alpha2

Quantile	adv1_estim	adv2_estim	Convergence
0.800	0.1600499	-0.0034783	0
0.805	0.1601237	-0.0041579	0
0.810	0.1584613	-0.0047028	0
0.815	0.1569799	-0.0043619	0
0.820	0.1562777	-0.0057510	0
0.825	0.1544489	-0.0058830	0
0.830	0.1529224	-0.0080884	0
0.835	0.1513097	-0.0093522	0
0.840	0.1477401	-0.0097701	0
0.845	0.1457119	-0.0090800	0
0.850	0.1451720	-0.0109913	0
0.855	0.1437399	-0.0125055	0
0.860	0.1423508	-0.0165918	0
0.865	0.1394730	-0.0184203	0
0.870	0.1364713	-0.0210291	0
0.875	0.1344264	-0.0211113	0
0.880	0.1304424	-0.0218132	0
0.885	0.1286889	-0.0230441	0
0.890	0.1213271	-0.0247231	0
0.895	0.1195753	-0.0293369	0
0.900	0.1140544	-0.0303015	0
0.905	0.1103176	-0.0321743	0
0.910	0.0943166	-0.0389673	0
0.915	0.0911129	-0.0509150	0
0.920	0.0698369	-0.0635428	0
0.925	0.0585347	-0.0656518	0
0.930	0.0176291	-0.0768808	0
0.935	0.0032736	-0.0992949	0
0.940	-0.0814749	-0.1467542	0
0.945	-0.0940106	-0.1674766	0
0.950	-0.1847543	-0.1781670	0

Estimation des paramètres en fixant l'advection:

Table 23: Optim estimations for each quantile, fixing adv1 = 0.05, adv2 = 0.02

Quantile	beta1_estim	beta2_estim	alpha1_estim	alpha2_estim	Convergence
0.920	0.3030700	0.3385247	1.379166	0.7612109	0
0.925	0.3144200	0.2941794	1.390161	0.8497224	0
0.930	0.3263195	0.4014027	1.406322	0.7406043	0
0.935	0.3433469	0.3497261	1.378683	0.8424624	0
0.940	0.3522116	0.4834349	1.367830	0.7880942	0
0.945	0.3504983	0.4306028	1.391494	0.8722372	0
0.950	0.3352821	0.5834611	1.441279	0.8102129	0

Optimisation avec advection

Table 24: Optim estimations for each quantile, true beta1 = 0.4, beta2 = 0.2, alpha1 = 1.5, alpha2 = 1, adv1 = 0.05, adv2 = 0.02

Quantile	beta1_estim	beta2_estim	alpha1_estim	alpha2_estim	adv1_estim	adv2_estim	Convergence
0.80	0.0844110	0.1210059	1.721620	0.2864330	-0.1107711	-0.2010732	0
0.81	0.0895904	0.1322119	1.704236	0.2736210	-0.1182816	-0.2028394	0
0.82	0.0978306	0.1354284	1.684470	0.3264491	-0.1230472	-0.1995182	0
0.83	0.1139855	0.1478513	1.633022	0.2895092	-0.1296304	-0.1985769	0
0.84	0.1252489	0.1641239	1.627516	0.2569351	-0.1285044	-0.1913289	0
0.85	0.2928048	0.0005835	1.091331	1.1362838	-0.1608366	-0.2508375	0
0.86	0.1442630	0.1810029	1.628346	0.2736871	-0.1244049	-0.1895795	0
0.87	0.2771690	0.0004635	1.372954	1.2324550	-0.1359297	-0.2526561	0
0.88	0.2793094	0.0009495	1.422125	1.2295096	-0.1428927	-0.2338875	0
0.89	0.1764000	0.2380451	1.660702	0.3911577	-0.1145083	-0.1620524	0
0.90	0.5160449	0.4163741	1.325482	0.0001020	0.1843230	-0.1250737	0
0.91	0.2125817	0.3130571	1.597773	0.4534244	-0.1280784	-0.1558901	0
0.92	0.2157425	0.3478765	1.633185	0.4756903	-0.1372140	-0.1715014	0
0.93	0.2265817	0.4265801	1.668601	0.4564472	-0.1488297	-0.1504605	0
0.94	0.2493132	0.4791208	1.636165	0.5234382	-0.1792591	-0.1880583	0
0.95	0.2450992	0.5506359	1.686868	0.6106656	-0.1941791	-0.1782521	0

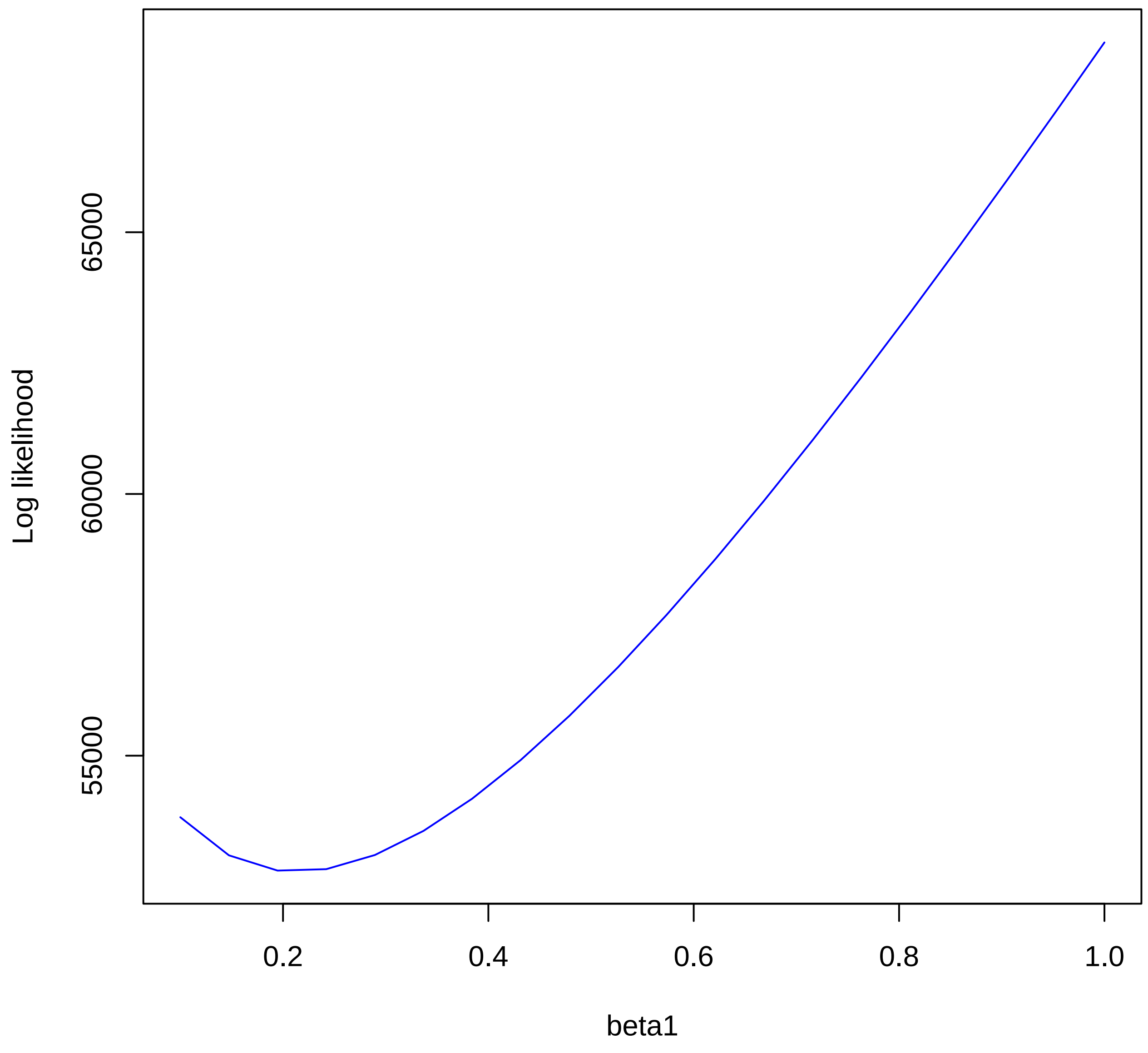
Profiled log likelihood

```
# q <- 0.95
# excesses <- empirical_excesses(simu_df_adv, quantile = q, df_lags = df_lags)

# mle_rain <- mle2(neg_ll_par_adv, start = list(beta1 = true_param[1],
#                                   beta2 = true_param[2],
#                                   alpha1 = true_param[3],
```

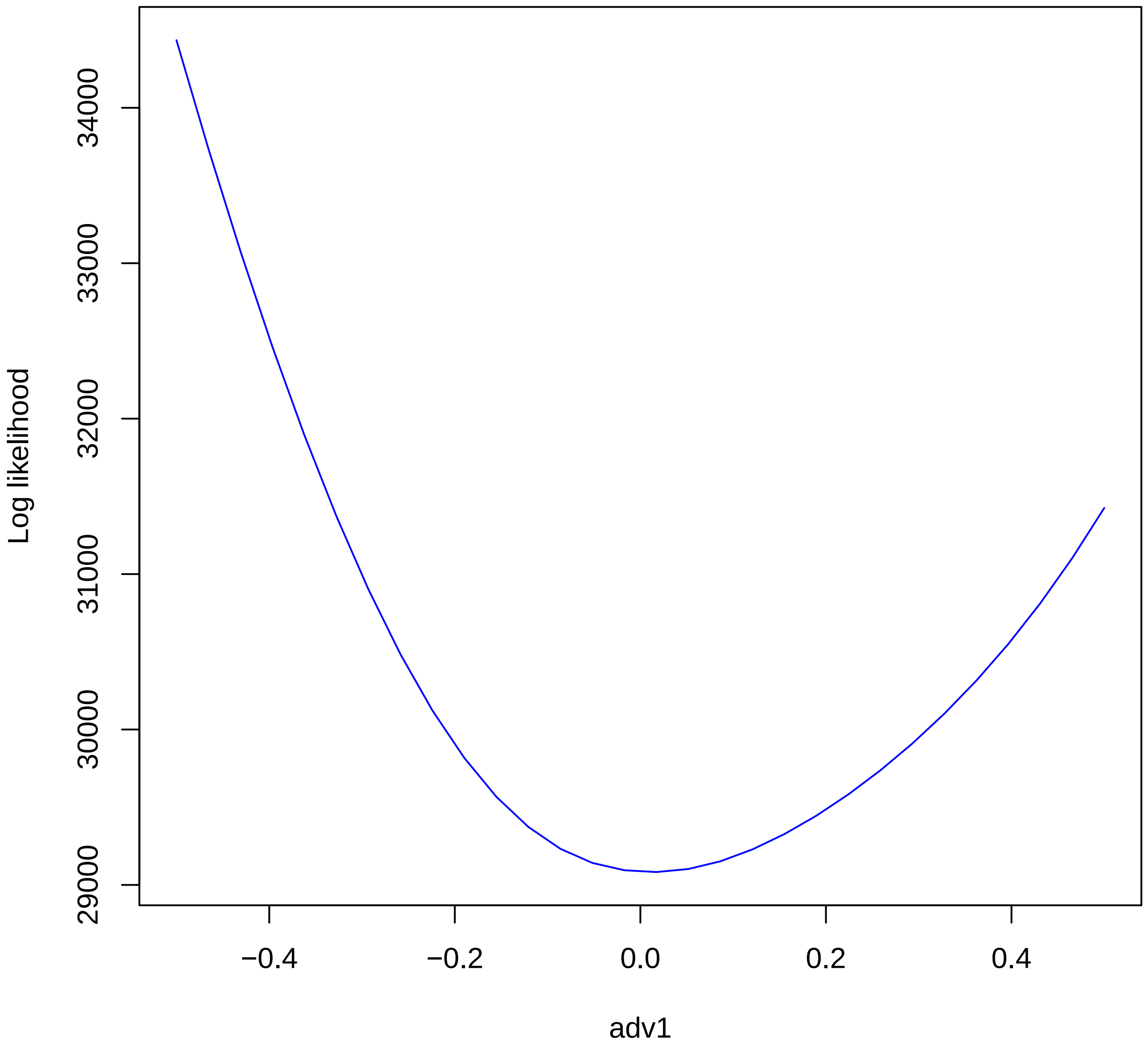
```r
#                                  alpha2 = true_param[4],
#                                  adv1 = true_param[5],
#                                  adv2 = true_param[6]),
#                  data = list(simu = simu_df_adv,
#                         quantile = q,
#                         df_lags = df_lags,
#                         locations = sites_coords,
#                         excesses = excesses,
#                         method = "CG"),
#                   control = list(maxit = 10000))

# rain_pro <- bbmle::profile(mle_rain) # profiled likelihood
# par(mfrow = c(2, 3))
# plot(rain_pro)

# bbmle::confint(rain_pro, level = 0.95) # confidence intervals
```

Plot log likelihood

```r
# Define the fixed parameters
beta2 <- 0.2
alpha1 <- 1.5
alpha2 <- 1.0
adv1 <- 0.05
adv2 <- 0.02

beta1_values <- seq(0.1, 1, length.out = 20)
q <- 0.9
nll <- numeric(length(beta1_values))

for (i in 1:length(nll)) {
  excesses <- empirical_excesses(simu_df_adv, quantile = q, df_lags = df_lags)
  nll[i] <- neg_ll_par_adv(beta1_values[i], beta2, alpha1, alpha2, adv1, adv2,
                      simu_df_adv, df_lags, sites_coords, quantile = q,
                      excesses = excesses)
}

# plot
par(mfrow = c(1, 1))
plot(beta1_values, nll, type = "l", col = "blue",
     xlab = "beta1", ylab = "Log likelihood")
```



```r
# Define the fixed parameters
beta1 <- 0.4
beta2 <- 0.2
alpha1 <- 1.5
alpha2 <- 1.0
adv1 <- 0.05
adv2 <- 0.02

adv1_values <- seq(-0.5, 0.5, length.out = 30)
q <- 0.93
nll <- numeric(length(adv1_values))

for (i in 1:length(nll)) {
  excesses <- empirical_excesses(simu_df_adv, quantile = q, df_lags = df_lags)
  nll[i] <- neg_ll_par_adv(beta1, beta2, alpha1, alpha2, adv1_values[i], adv2,
                    simu_df_adv, df_lags, sites_coords, quantile = q,
                    excesses = excesses)
}

# plot
par(mfrow = c(1, 1))
plot(adv1_values, nll, type = "l", col = "blue",
     xlab = "adv1", ylab = "Log likelihood")
```

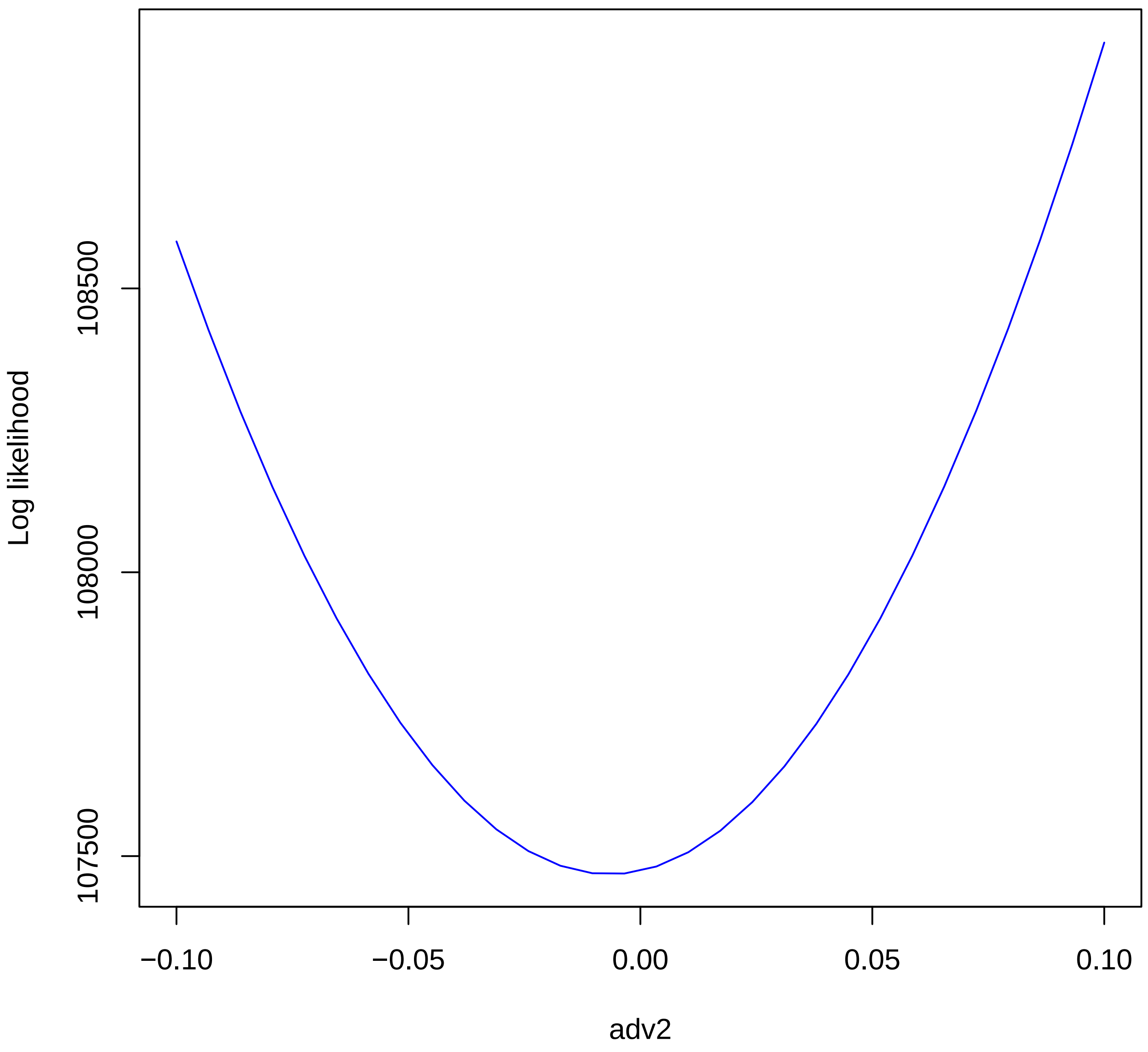


```
min_adv <- adv1_values[which.min(nll)]
print(min_adv)
```

```
## [1] 0.01724138
```

```
# Define the fixed parameters
beta1 <- 0.4
```

```r
beta2 <- 0.2
alpha1 <- 1.5
alpha2 <- 1.0
adv1 <- 0.05
adv2 <- 0.02

adv2_values <- seq(-0.1, 0.1, length.out = 30)
q <- 0.85
nll <- numeric(length(adv2_values))

for (i in 1:length(nll)) {
  excesses <- empirical_excesses(simu_df_adv, quantile = q, df_lags = df_lags)
  nll[i] <- neg_ll_par_adv(beta1, beta2, alpha1, alpha2, adv1, adv2_values[i],
                      simu_df_adv, df_lags, sites_coords, quantile = q,
                      excesses = excesses)
}

# plot
par(mfrow = c(1, 1))
plot(adv2_values, nll, type = "l", col = "blue",
     xlab = "adv2", ylab = "Log likelihood")
```



```
min_adv <- adv2_values[which.min(nll)]
print(min_adv)
```

```
## [1] -0.003448276
```