prova dell'ottimizzatore per la network crr

Importazione dei dati

I dati sono gli stessi del primo esempio in Noma et al. sulla cessazione dell'abitudine al fumo tramite 3 diverse tipologie di assistenza.

```
devtools::load_all(".")

data("smoking", package = "tesi.ncrr")
ls()
smoking$tik <- with(smoking, log(rik) - log(nik - rik))
names(smoking)

# specifico il design della meta-analisi
des <- ncrr.design(smoking)
str(des)
des
```

Prima prova

codice "legacy" in cui provo un singolo studio (il 3 : A (baseline) vs. B)

```
theta.oss <- smoking[smoking$study.id == 3, "tik"]
Gamma <- crr.vcov.within(smoking[smoking$study.id == 3, "rik"],
                         smoking[smoking$study.id == 3, "nik"])
```

```
param.mv0 <- optim(c(0, 1, 0, 1, 0, 1), \(x) -llik1(x, theta.oss, Gamma),
                   lower = c(-Inf, -Inf, -Inf, 1e-10, -1, 1e-10),
                   upper = c(Inf, Inf, Inf, Inf, 1, Inf),
                   method = "L-BFGS-B",
                   control = list(fnscale = 1e-10, factr = 1, maxit = 1e6))
```

ottengo stime dei parametri sensate solo per i parametri di posizione, mentre le varianze e la correlazione si "incastrano su 0"

```
param.mv0
```

```
$par
[1] -0.4418085383  0.7937901905 -2.0650986664  0.0000000001  0.0000000000
[6]  0.0000000001

$value
[1] -2.422975

$counts
function gradient
      45       45

$convergence
[1] 0

$message
[1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
```

Riproduzione degli esempi giocattolo contenuti nel documento `verosim1.pdf`

Due studi con design {0; 1} e {0; 2}

```
# definisco il design `toy1` prendendo gli studi n. 4 e 6, che hanno design
# diversi, compatibili con l'esempio giocattolo
toy1 <- subset(des, c(4, 6))
str(toy1)
```

```
List of 6
 $ design    :List of 2
  ..$ : num [1:2] 0 1
  ..$ : num [1:2] 0 2
  ..- attr(*, "dim")= int 2
 $ gamma     : num [1:4] 0.05701 0.04942 0.01457 0.00368
 $ theta     : num [1:4] -3.5912 -3.1977 -2.1687 0.0336
```

```
 $ treatments: chr [1:4] "A - No contact" "B - Self help" "C - Individual counseling" "D - Gr
 $ x         :'data.frame': 50 obs. of  6 variables:
  ..$ study.id   : int [1:50] 1 1 1 2 2 2 3 3 4 4 ...
  ..$ treatment  : Factor w/ 4 levels "A - No contact",..: 1 3 4 2 3 4 1 2 1 2 ...
  ..$ is.baseline: num [1:50] 1 0 0 1 0 0 1 0 1 0 ...
  ..$ rik        : num [1:50] 9 23 10 11 12 29 79 77 18 21 ...
  ..$ nik        : num [1:50] 140 140 138 78 85 170 702 694 671 535 ...
  ..$ tik        : num [1:50] -2.68 -1.63 -2.55 -1.81 -1.81 ...
 $ baseline  : chr "A - No contact"
 - attr(*, "class")= chr "ncrr.design"
```

```r
init1 <- getInitial(toy1, transform = FALSE)
# invoco la funzione di ottimizzazione a partire dal design specificato
fn1 <- get.llik.from.design(toy1, echo = 0, transform = FALSE)
```

```r
# ottimizzazione vincolata
mv1 <- optim(init1, \(x) -fn1(x),
             lower = attr(init1, "lower"),
             upper = attr(init1, "upper"),
             method = "L-BFGS-B",
             control = list(fnscale = 1e-10, factr = 1, maxit = 1e6))
```

σ^2_{01} e σ^2_{02} invece risultano negativi!!

```r
# controllo l'esito dell'ottimizzazione
source("diff_alpha1.R")
with(mv1, cat("Esito: ", convergence, " - ",
              if (is.null(message)) "OK" else message, "\n"))
```

```
Esito:  0  -  CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH
```

```r
# confronto i parametri calcolati con l'ottimizzatore...
param1 <- crr.split.par(mv1$par, 2)
param1
```

```
$sigma2
[1] 1e-10 1e-10

$rho
[1] 0
```

```
$sigma20
[1] 0.4692178

$mu0
[1] -2.850057

$beta
[1] -3.967246e-08  2.945763e-07

$alpha
[1] -3.19770097  0.03361726
```

```r
# ...con quelli in forma chiusa
alphahat <- do.call(alpha.cf1, append(param1[c("sigma2", "beta", "sigma20", "mu0")],
                                      list(design = toy1)))
alphahat # alpha calcolato in forma chiusa
```

```
     alpha1      alpha2
-3.19770097  0.03361726
```

```r
sigmahat <- do.call(sigma.cf1, append(param1[c("beta", "sigma20", "mu0")],
                                      list(design = toy1)))
sigmahat # sigma2 calcolato in forma chiusa
```

```
     sigma21      sigma22
-0.049417609 -0.003683326
```

Effettivamente, però, la forma della derivata (che ho calcolato con Maxima) sembrerebbe essere proprio così

$$\sigma_{01} = -\frac{(\gamma_{11} + \beta_{01}^2\gamma_{10})\sigma_0 + \gamma_{10}\gamma_{11}}{\sigma_0 + \gamma_{10}}$$
$$\sigma_{02} = -\frac{(\gamma_{22} + \beta_{02}^2\gamma_{20})\sigma_0 + \gamma_{20}\gamma_{22}}{\sigma_0 + \gamma_{20}}$$

C'è un'apparente contraddizione!

Teoricamente σ_{01} dovrebbe essere un parametro di varianza, quindi positivo; ma nell'economia del modello, ossia se si considera semplicemente la distribuzione di $\hat{\theta}$, è sufficiente che $\beta_{01}^2\sigma_0^2 + \sigma_{01}^2 > 0$ perchè la matrice di Var-cov della normale sia sensata.
Per cui, il modello "vuole andare" verso σ negativi perchè tanto la densità di $\hat{\theta}$ è comunque ben definita, mentre invece noi sappiamo che non lo è per via del modello statistico che sottende questa distribuzione.
Questo spiegherebbe anche il motivo per cui l'ottimizzatore s'incastra su 0.
Come si può agire in questo caso?

Provo a vedere se le stime cambiano fissando i parametri

```
# fisso parametro alpha
fixpar <- list(alpha = alphahat)
# disabilito stampa dei passaggi intermedi con `echo = 0`
fn1 <- get.llik.from.design(toy1, echo = 0, transform = FALSE)
mv1f <- optim(init1, \(x) -fn1(x, fixed = fixpar),
              lower = attr(init1, "lower"),
              upper = attr(init1, "upper"),
              method = "L-BFGS-B",
              control = list(fnscale = 1e-10, factr = 1, maxit = 1e6))

# confronto tra i parametri da stima non vincolata e vincolata rispettivamente
all.equal(crr.split.par(mv1$par, 2),
          subst.params(crr.split.par(mv1f$par, 2), fixpar))
```

```
[1] "Component \"sigma20\": Mean relative difference: 1.821518e-07"
[2] "Component \"mu0\": Mean relative difference: 7.882199e-08"
[3] "Component \"beta\": Mean relative difference: 0.00337464"
[4] "Component \"alpha\": names for current but not for target"
```

La differenza è comunque minima.

```
c(valore_ottimo = (mv1$value),
  valore_ottimo_vincolato = mv1f$value,
  differenza = mv1$value - mv1f$value)
```

```
          valore_ottimo valore_ottimo_vincolato             differenza
          -3.122380e-01          -3.122380e-01           5.635492e-13
```

Due studi con design uguale {0; 1}

```
# seleziono solo studi n. 4 e 5
toy2 <- subset(des, c(4, 5))
str(toy2)
```

```
List of 6
 $ design    :List of 2
  ..$ : num [1:2] 0 1
  ..$ : num [1:2] 0 1
  ..- attr(*, "dim")= int 2
 $ gamma     : num [1:4] 0.057 0.0494 0.1331 0.0587
 $ theta     : num [1:4] -3.59 -3.2 -2.6 -1.9
 $ treatments: chr [1:4] "A - No contact" "B - Self help" "C - Individual counseling" "D - G
 $ x         :'data.frame': 50 obs. of  6 variables:
  ..$ study.id   : int [1:50] 1 1 1 2 2 2 3 3 4 4 ...
  ..$ treatment  : Factor w/ 4 levels "A - No contact",..: 1 3 4 2 3 4 1 2 1 2 ...
  ..$ is.baseline: num [1:50] 1 0 0 1 0 0 1 0 1 0 ...
  ..$ rik        : num [1:50] 9 23 10 11 12 29 79 77 18 21 ...
  ..$ nik        : num [1:50] 140 140 138 78 85 170 702 694 671 535 ...
  ..$ tik        : num [1:50] -2.68 -1.63 -2.55 -1.81 -1.81 ...
 $ baseline  : chr "A - No contact"
 - attr(*, "class")= chr "ncrr.design"
```

```
init2 <- getInitial(toy2, transform = FALSE)
fn2 <- get.llik.from.design(toy2, echo = 1, transform = FALSE)
```

```
mv2 <- optim(init2, \(x) -fn2(x),
             lower = attr(init2, "lower"),
             upper = attr(init2, "upper"),
             method = "L-BFGS-B",
             control = list(fnscale = 1e-10, factr = 1, maxit = 1e6))
```

```
with(mv2, cat("Esito: ", convergence, " - ",
              if (is.null(message)) "OK" else message, "\n"))
```

```
Esito:  52  -  ERROR: ABNORMAL_TERMINATION_IN_LNSRCH
```

```r
# i parametri ottimizzati in un formato più leggibile...
param2 <- crr.split.par(mv2$par, 1)
param2
```

```
$sigma2
[1] 1e-10

$rho
[1] 0

$sigma20
[1] 0.2163708

$mu0
[1] -3.117626

$beta
[1] 1.328258

$alpha
[1] 1.587044
```

```r
# ... da confrontare con le stime analitiche:

alphahat <- do.call(alpha.cf2, append(param2[c("sigma2", "beta", "sigma20", "mu0")],
                                      list(design = toy2)))
alphahat
```

```
[1] 1.677895
```

```r
sigmahat <- do.call(sigma.cf2, append(param2[c("alpha", "beta", "sigma20", "mu0")],
                                      list(design = toy2)))
sigmahat
```

```
[1] -0.1508319
```

Ottimizzazione vincolata

```r
fixpar <- list(alpha = alphahat)
fn2 <- get.llik.from.design(toy2, echo = 0, transform = FALSE)
mv2f <- optim(init2, \(x) -fn2(x, fixed = fixpar),
              lower = attr(init2, "lower"),
              upper = attr(init2, "upper"),
              method = "L-BFGS-B",
              control = list(fnscale = 1e-10, factr = 1, maxit = 1e6))

# confronto i parametri
all.equal(crr.split.par(mv2$par, 1),
          subst.params(crr.split.par(mv2f$par, 1), fixpar))
```

```
[1] "Component \"sigma20\": Mean relative difference: 0.03097004"
[2] "Component \"mu0\": Mean relative difference: 0.001161248"
[3] "Component \"beta\": Mean relative difference: 0.02059641"
[4] "Component \"alpha\": Mean relative difference: 0.05724542"
```

```r
# confronto il valore ottenuto della funzione obiettivo
c(valore_ottimo = (mv2$value),
  valore_ottimo_vincolato = mv2f$value,
  differenza = mv2$value - mv2f$value)
```

```
          valore_ottimo valore_ottimo_vincolato              differenza
           1.6854990887            1.6862237442           -0.0007246554
```

Esperimenti più complessi

TODO: Qui andranno implementate le derivate esplicite una volta che la struttura generale del modello statistico sarà definita.

Esperimento 1

specifico il design e provo a condurre un esperimento selezionando un gruppo di studi con design "semplice" (2 trattamenti a confronto, di cui uno è il baseline)

```r
# prendo tutti gli studi con design {0; 1}
des1 <- subset(des, which(sapply(des$design, \(d) identical(c(0, 1), d))))
str(des1)
```

```
List of 6
 $ design    :List of 3
  ..$ : num [1:2] 0 1
  ..$ : num [1:2] 0 1
  ..$ : num [1:2] 0 1
  ..- attr(*, "dim")= int 3
 $ gamma     : num [1:6] 0.0139 0.0143 0.057 0.0494 0.1331 ...
 $ theta     : num [1:6] -2.07 -2.08 -3.59 -3.2 -2.6 ...
 $ treatments: chr [1:4] "A - No contact" "B - Self help" "C - Individual counseling" "D - G
 $ x         :'data.frame': 50 obs. of  6 variables:
  ..$ study.id   : int [1:50] 1 1 1 2 2 2 3 3 4 4 ...
  ..$ treatment  : Factor w/ 4 levels "A - No contact",..: 1 3 4 2 3 4 1 2 1 2 ...
  ..$ is.baseline: num [1:50] 1 0 0 1 0 0 1 0 1 0 ...
  ..$ rik        : num [1:50] 9 23 10 11 12 29 79 77 18 21 ...
  ..$ nik        : num [1:50] 140 140 138 78 85 170 702 694 671 535 ...
  ..$ tik        : num [1:50] -2.68 -1.63 -2.55 -1.81 -1.81 ...
 $ baseline  : chr "A - No contact"
 - attr(*, "class")= chr "ncrr.design"
```

```
par.init <- getInitial(des1, transform = FALSE)

# calcolo esplicitamente i parametri della normale per hat(theta)
# prova per i mu0
do.call(crr.get.mu,
        append(list(des1),
               crr.split.par(param.mv0$par, np = 1)[c("alpha", "beta", "mu0")]))
```

```
[1] -2.065099 -2.081064 -2.065099 -2.081064 -2.065099 -2.081064
```

```
# prova per i sigma
do.call(crr.get.sigma,
        append(list(des1),
               crr.split.par(param.mv0$par, np = 1)[c("beta", "sigma20", "rho", "sigma2")]))
```

```
             [,1]         [,2]         [,3]         [,4]         [,5]
[1,] 1.000000e-10 7.937902e-11 0.000000e+00 0.000000e+00 0.000000e+00
[2,] 7.937902e-11 1.630103e-10 0.000000e+00 0.000000e+00 0.000000e+00
[3,] 0.000000e+00 0.000000e+00 1.000000e-10 7.937902e-11 0.000000e+00
[4,] 0.000000e+00 0.000000e+00 7.937902e-11 1.630103e-10 0.000000e+00
[5,] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 1.000000e-10
[6,] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 7.937902e-11
```

```
              [,6]
[1,] 0.000000e+00
[2,] 0.000000e+00
[3,] 0.000000e+00
[4,] 0.000000e+00
[5,] 7.937902e-11
[6,] 1.630103e-10
```

```
do.call(crr.get.sigma,
        append(list(des1),
               crr.split.par(par.init, np = 1)[c("beta", "sigma20", "rho", "sigma2")]))
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    1    0    0    0    0
[2,]    1    2    0    0    0    0
[3,]    0    0    1    1    0    0
[4,]    0    0    1    2    0    0
[5,]    0    0    0    0    1    1
[6,]    0    0    0    0    1    2
```

```
opt.fn <- get.llik.from.design(des1, echo = 3, transform = FALSE)
```

```
param.mv1 <- optim(par.init, \(x) -opt.fn(x),
                   lower = attr(par.init, "lower"),
                   upper = attr(par.init, "upper"),
                   method = "L-BFGS-B",
                   control = list(fnscale = 1e-10, factr = 1, maxit = 1e6))
```

```
# Stime MV in formato leggibile
crr.split.par(param.mv1$par, 1, FALSE)
```

```
$sigma2
[1] 1e-10

$rho
[1] 0

$sigma20
[1] 0.4419063
```

```
$mu0
[1] -2.641997

$beta
[1] 0.7517716

$alpha
[1] -0.4534419
```

Ottimizzazione sulle trasformate dei parametri (log per la varianza, Fisher per la correlazione)

```r
opt.fn <- get.llik.from.design(des1, echo = 3, transform = TRUE)
param.mv12 <- optim(getInitial(des1, transform = TRUE), \(x) -opt.fn(x),
                    method = "Nelder-Mead",
                    control = list(fnscale = 1e-10, factr = 1, maxit = 1e6))
```

```r
param.mv12
```

```
$par
      alpha        beta         mu0     sigma20         rho      sigma2
 -0.4534395   0.7517776  -2.6420062  -0.8167505  -1.2157174 -16.6835894

$value
[1] 2.842031

$counts
function gradient
     671       NA

$convergence
[1] 0

$message
NULL
```

Ottimizzazione con random start

```r
environment(opt.fn)$echo <- 0

par2 <- getInitial(des1, seed = c(alpha = 3, beta = 4, sigma20 = 6, sigma2 = 9),
                   rep = 50, transform = TRUE)
for (i in 1:nrow(par2)) {
  mvcurr <- optim(par2[i, ], \(x) -opt.fn(x),
                  method = "Nelder-Mead",
                  control = list(fnscale = 1e-10, factr = 1, maxit = 1e6))
  print(mvcurr$counts)
  par2[i, ] <- mvcurr$par
}
```

```
function gradient
     539       NA
function gradient
     739       NA
function gradient
     615       NA
function gradient
     623       NA
function gradient
    1085       NA
function gradient
     813       NA
function gradient
    1017       NA
function gradient
    1165       NA
function gradient
    1249       NA
function gradient
     791       NA
function gradient
    1183       NA
function gradient
     769       NA
function gradient
    1115       NA
function gradient
    1103       NA
function gradient
     593       NA
```

```
function gradient
     711       NA
function gradient
     705       NA
function gradient
     277       NA
function gradient
     781       NA
function gradient
     489       NA
function gradient
     917       NA
function gradient
     783       NA
function gradient
     653       NA
function gradient
    1237       NA
function gradient
     905       NA
function gradient
    1013       NA
function gradient
    1065       NA
function gradient
     825       NA
function gradient
     935       NA
function gradient
    1289       NA
function gradient
    1127       NA
function gradient
     439       NA
function gradient
     963       NA
function gradient
    1085       NA
function gradient
     513       NA
function gradient
     583       NA
function gradient
```

```
    1241       NA
function gradient
    1721       NA
function gradient
     595       NA
function gradient
    1159       NA
function gradient
     893       NA
function gradient
     607       NA
function gradient
    1011       NA
function gradient
     819       NA
function gradient
     629       NA
function gradient
     597       NA
function gradient
     693       NA
function gradient
     771       NA
function gradient
    1133       NA
function gradient
     915       NA
```

```r
# stampa di statistiche descrittive per tutti i valori ottimizzati
crr.split.par(par2, transform = TRUE, np = 1) |>
  do.call(cbind, args = _) |>
  apply(2, \(x) c(min = min(x), max = max(x), mediana = median(x),
                  media = mean(x), dev.std = sd(x),
                  scarto.medio.assoluto.mediana = mean(abs(x - median(x))))) |>
  t() |>
  as.data.frame()
```

```
                min          max       mediana         media      dev.std
sigma2   3.840744e-13  0.003116877  1.254287e-07  6.537789e-05 0.0004406548
rho     -1.000000e+00  1.000000000  9.998695e-01  4.961262e-01 0.8426699538
sigma20  4.389981e-01 31.552406880  4.418764e-01  1.064024e+00 4.3997025425
mu0     -2.644568e+00  3.176305875 -2.641973e+00 -2.525586e+00 0.8228259743
```

```
beta     7.374179e-01  0.752683083  7.517213e-01  7.514299e-01 0.0020749907
alpha   -4.847303e-01 -0.451029232 -4.535468e-01 -4.542136e-01 0.0045827766
        scarto.medio.assoluto.mediana
sigma2                   6.534028e-05
rho                      5.038600e-01
sigma20                  6.225569e-01
mu0                      1.166333e-01
beta                     5.017574e-04
alpha                    1.177145e-03
```

```
#apply(par2, 1, \(x) crr.split.par(x, transform = TRUE, np = 1)$sigma20)
#apply(par2, 1, \(x) crr.split.par(x, transform = TRUE, np = 1)$rho)
```

Esperimento 2

Butto dentro tutti gli studi che hanno baseline 0, anche con design a tre trattamenti.

```
des2 <- subset(des, c(1, 3:4, 6:8))
par.init <- getInitial(des2, transform = TRUE)

# calcoli dei parametri della normale per hat(theta)
# prova per i mu0
do.call(crr.get.mu,
        append(list(des2),
               crr.split.par(par.init, np = 3, transform = TRUE)[c("alpha", "beta", "mu0")]))
```

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

```
# prova per i sigma
do.call(crr.get.sigma,
        append(list(des2),
               crr.split.par(par.init, np = 3, transform = TRUE)[c("beta", "sigma20", "rho",
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
[1,]    1    1    1    0    0    0    0    0    0     0     0     0     0
[2,]    1    2    1    0    0    0    0    0    0     0     0     0     0
[3,]    1    1    2    0    0    0    0    0    0     0     0     0     0
[4,]    0    0    0    1    1    0    0    0    0     0     0     0     0
```

```
 [5,]    0    0    0    1    2    0    0    0    0    0    0    0    0
 [6,]    0    0    0    0    0    1    1    0    0    0    0    0    0
 [7,]    0    0    0    0    0    1    2    0    0    0    0    0    0
 [8,]    0    0    0    0    0    0    0    1    1    0    0    0    0
 [9,]    0    0    0    0    0    0    0    1    2    0    0    0    0
[10,]    0    0    0    0    0    0    0    0    0    1    1    0    0
[11,]    0    0    0    0    0    0    0    0    0    1    2    0    0
[12,]    0    0    0    0    0    0    0    0    0    0    0    1    1
[13,]    0    0    0    0    0    0    0    0    0    0    0    1    2
```

Conduco l'ottimizzazione sui parametri modificati (il log per i parametri di varianza, la trasformata di Fisher per la correlazione) tramite Nelder-Mead.

```
opt.fn <- get.llik.from.design(des2, echo = 0, transform = TRUE)
```

```
param.mv2 <- optim(par.init, \(x) -opt.fn(x),
                   #lower = attr(par.init, "lower"),
                   #upper = attr(par.init, "upper"),
                   method = "Nelder-Mead",
                   control = list(fnscale = 1e-10, factr = 1, maxit = 1e6))
```

```
# stime MV in formato leggibile
crr.split.par(param.mv2$par, 3, TRUE)
```

```
$sigma2
[1] 0.4400165 1.1085385 2.3814461

$rho
[1] 0.5477996

$sigma20
[1] 0.7013531

$mu0
[1] -2.777796

$beta
[1] 0.7497412 0.8910234 1.3174203

$alpha
[1] -0.009402901  0.525898177  0.636190976
```

Integrazione delle derivate analitiche

Immagino che in questo caso la derivata debba tener conto della trasformazione dei parametri, giusto?

Provo anche con il Simulated Annealing, come alternativa lenta ma che dovrebbe fornire un risultato più robusto, non avendo a disposizione le derivate esplicite.

```
sann.mv2 <- optim(par.init, \(x) -opt.fn(x),
                  #lower = attr(par.init, "lower"),
                  #upper = attr(par.init, "upper"),
                  method = "SANN",
                  control = list(fnscale = 1e-10, factr = 1, maxit = 1e6,
                                 temp = 10, tmax = 10, trace = 1))
save(sann.mv2, file = "sann_optim.rda")
```

Diagnostiche di confronto tra Nelder-mead e Simulated Annealing

```
load("../sann_optim.rda")

# confronto dei parametri
all.equal(crr.split.par(param.mv2$par, 3, TRUE),
          crr.split.par(sann.mv2$par, 3, TRUE))
```

```
[1] "Component \"sigma2\": Mean relative difference: 0.9150497"
[2] "Component \"rho\": Mean relative difference: 0.7409922"
[3] "Component \"sigma20\": Mean relative difference: 5.066221"
[4] "Component \"mu0\": Mean relative difference: 0.09546018"
[5] "Component \"beta\": Mean relative difference: 0.1849387"
[6] "Component \"alpha\": Mean relative difference: 1.316539"
```

```
# confronto dei valori della f.o.
c(valore_ottimo_NELMEAD = (param.mv2$value),
  valore_ottimo_SANN = sann.mv2$value,
  differenza = param.mv2$value - sann.mv2$value)
```

```
valore_ottimo_NELMEAD    valore_ottimo_SANN            differenza
            -2.918652              7.289381            -10.208033
```

Facendolo invece partire dall'ottimo individuato da Nelder-Mead:

```
sann.mv22 <- optim(param.mv2$par, \(x) -opt.fn(x),
                   #lower = attr(par.init, "lower"),
                   #upper = attr(par.init, "upper"),
                   method = "SANN",
                   control = list(fnscale = 1e-10, factr = 1, maxit = 1e6,
                                  temp = 10, tmax = 10, trace = 1))
save(sann.mv22, file = "sann_optim2.rda")
```

Confronti

```
load("../sann_optim2.rda")

# confronto dei parametri
all.equal(crr.split.par(param.mv2$par, 3, TRUE),
          crr.split.par(sann.mv22$par, 3, TRUE))
```

```
[1] "Component \"sigma2\": Mean relative difference: 0.9082057"
[2] "Component \"rho\": Mean relative difference: 0.03577454"
[3] "Component \"sigma20\": Mean relative difference: 2.776739"
[4] "Component \"mu0\": Mean relative difference: 0.1395841"
[5] "Component \"beta\": Mean relative difference: 0.5088941"
[6] "Component \"alpha\": Mean relative difference: 2.574915"
```

```
# confronto valori della f.o.
c(valore_ottimo_NELMEAD = (param.mv2$value),
  valore_ottimo_SANN = sann.mv22$value,
  differenza = param.mv2$value - sann.mv22$value)
```

```
valore_ottimo_NELMEAD    valore_ottimo_SANN            differenza
            -2.918652             -4.613472              1.694820
```

Nelder-Mead con punti di partenza casuali

```
par22 <- getInitial(des2, seed = c(alpha = 3, beta = 4, sigma20 = 6, sigma2 = 9),
                    rep = 50, transform = TRUE)
for (i in 1:nrow(par22)) {
  mvcurr <- optim(par22[i, ], \(x) -opt.fn(x),
                  method = "Nelder-Mead",
                  control = list(fnscale = 1e-10, factr = 1, maxit = 1e6))
  message("Random start: ", i, "; n. iter: ", mvcurr$counts[1])
  par22[i, ] <- mvcurr$par
}
```

```r
# per ogni random start calcolo alcune statistiche delle stime MV ottenute
# (sui parametri ritrasformati in scala originale)
crr.split.par(par22, 3, transform = TRUE) |>
  do.call(cbind, args = _) |>
  apply(2, \(x) c(min = min(x), max = max(x), mediana = median(x),
                  media = mean(x), dev.std = sd(x),
                  scarto.medio.assoluto.mediana = mean(abs(x - median(x))))) |>
  t() |>
  as.data.frame()
```

```
                 min         max      mediana       media      dev.std
sigma21  6.659595e-17 3797.291016  0.645939060 142.1231536  546.7487163
sigma22  5.080447e-09 1554.370862  1.699437127  95.8401667  284.0074309
sigma23  3.546545e-11  884.053860  3.613726078  83.9896964  195.6415519
rho     -9.745931e-01    1.000000  0.008062299   0.1279957    0.6163536
sigma20  1.104054e-03 6587.294116  0.345611166 246.6191003 1216.5504241
mu0     -3.052983e+00    8.726469 -2.540371455  -2.0748116    1.7119363
beta1   -1.986816e+01    7.401745  0.808574143   0.5537402    3.8007214
beta2   -8.294903e+00   11.292009  1.184598588   1.0290069    3.2968466
beta3   -1.193666e+01    6.232125  0.106664635  -0.3808679    3.6366669
alpha1  -1.158843e+01   10.058294 -0.433023574  -0.2918642    4.5409409
alpha2  -1.227188e+01   14.626977  2.106944941   1.2489877    5.1537191
alpha3  -1.752803e+01   13.915821 -0.695408393  -1.3329731    5.8264827
        scarto.medio.assoluto.mediana
sigma21                     142.0363081
sigma22                      95.1993249
sigma23                      83.5589145
rho                           0.4722116
sigma20                     246.4198938
mu0                           0.5777098
beta1                         1.9091662
beta2                         2.3173950
beta3                         2.6912633
alpha1                        2.9826612
alpha2                        3.7679963
alpha3                        4.5021994
```

non un grande risultato...