Manual for `SSimp.0`

Sina Rüeger, sina.rueeger@gmail.com

September 23, 2016

1 Parameters

`--data` sdfsf

`--lambda` sdfsdf

`--out` sdfsdf

A Edge-cases

B R-code

B.1 Application

```
# ============================================================================
# Approximative Bayesian Computing (ABC)
# Generic Markov Chain Monte Carlo sampler without likelihoods
#
# Andreas Scheidegger
# 10.11.2009
# ============================================================================

# See Marjoram et al. (2003), Markov Chain Monte Carlo without likelihoods, PNAS
    100(26), pp 15324-15328.
# Implementation of algorithm F

# -- Arguments --
# f.dist: function which simualtes data and returns the distance between the real
    and the simulated data.
#         The first argument must be the parameter vector.
# d.priori: function which returns the density of the prior distribution of a
    parameter vector.
# n.sample: number of samples
# eps: accepted tolerance between data and model output
# init: initial values
# sigma: vector of standard dev. of the gaussian proposal distribution
# verbose: number of samples between printed outputs
# ...: arguments for f.dist
#
# -- Value --
# matrix containing in each row a (autocorrelated) sample of all parameters and
    the corresponding delta value

ABC.MCMC <- function (f.dist, d.priori, n.sample, eps, init, sigma, verbose=100,
    ...) {

  # Number of parameter
  n.para <- length(init)

  # Matrix to store samples
  sample <- matrix(NA, ncol=n.para, nrow=n.sample)

  # Vector to store  distances
  delta <- rep(NA, n.sample)

  # Initional values
  sample[1,] <- as.matrix(init)
  delta[1] <- f.dist(sample[1,], ...)

  # repeat for each sample
  for (k in 2:n.sample)  {

    # print
    if((k %% verbose)==0) cat("k =", k, "\n")

    # F1: generate candidate point
```

```
    x.prob <- sample[k-1,]+rnorm(n.para,sd=sigma)    # gaussian proposal density (
        rounded to integer)

    # F2, F3: simulate data and calculate distance to the real data
    dist <- f.dist(x.prob, ...)

    # if dist <= eps go to F4
    if(dist <= eps) {

      # F4: calculate prob. h (due to the symmetrie of the proposal density, it is
          not necessary in F4)
      h <- min(1, d.priori(x.prob)/d.priori(sample[k-1,]) )

      # F5: accept x.prob mit prob. h
      if(runif(1)<h) {
        sample[k,] <- x.prob
        delta[k] <- dist
      # else stay on the same point
      } else {
        sample[k,] <- sample[k-1,]
        delta[k] <- delta[k-1]
      }

    # else stay on the same point
    } else {
      sample[k,] <- sample[k-1,]
      delta[k] <- delta[k-1]
    }

  }

  # combine samples and delta
  sample <- cbind(delta, sample)
  if(length(names(init))!=n.para) colnames(sample) <- c("delta", paste("para.",1:n
      .para, sep="") )
  if(length(names(init))==n.para) colnames(sample) <- c("delta", names(init) )

  #return samples and distances
  return(sample)
}
```