mclcar: an R Package for Maximum Monte Carlo Likelihood Estimation of Conditional Auto-regression Models

Zhe Sha
University of Oxford

Abstract

We briefly describe the Monte Carlo likelihood method in estimating CAR models and the implementation in the pkgmclcar. Then we demonstrate the usage of the package through examples of Gaussian, Binomial and Poisson data.

Keywords: Monte Carlo likelihood, CAR models, spatial statistics, response surface design, **mclcar**, R.

1. Introduction

Conditional auto-regression (CAR) models are frequently used with spatial data. However, the likelihood of such a model is expensive to compute even for a moderately sized data set of around 1000 sites. For models involving latent variables, the likelihood is not usually available in closed form.

1.1. CAR models

The CAR models are defined through full conditionals of observations y_i from each spatial unit i

$$y_i|y_{j\sim i} \sim \mathcal{N}(X_i\beta + \sum_j \rho\, w_{ij}(y_j - X_j\beta),\, \sigma^2) \tag{1}$$

where X_j is the j^{th} row of the design matrix X, β is a vector of the linear coefficients, ρ is the spatial coefficient showing the global strength of the spatial effect and $\{w_{ij}\}$ are the elements of the spatial weight matrix W reflecting the local spatial effect. The joint distribution of Y is the multivariate Gaussian distribution:

$$Y \;\sim\; \mathcal{N}(X\beta, \Sigma) \tag{2}$$

where $\Sigma = \sigma^2(I - \rho W)^{-1}$ is the variance-covariance matrix and $\rho \in (1/\lambda_1,\, 1/\lambda_N)$ where $\lambda_1 < \lambda_2 < \cdots < \lambda_N$ are the ordered eigenvalues of W.

Non-Gaussian observations can usually model the generalised linear models with a CAR latent

variable:

$$\begin{cases} \mathbf{y} \sim \pi(\mu) \\ g(\mu) = \eta \\ \eta = X\beta + Z, \;\; Z \sim \mathcal{N}(0, \Sigma) \end{cases} \tag{3}$$

Where π is the distribution of y and $g(\cdot)$ is some link function.

Maximum likelihood of the above models can be computationally expensive, especially for large N, due to the determinant of Σ in the likelihood of (2) and the integral for the latent variable Z for (3).

1.2. Overview

In this package, we implement the Monte Carlo approximation to the likelihood (extending the approach of Geyer and Thompson (1992)), and develop two strategies for maximising this. One strategy is to limit the step size by defining an experimental region using a Monte Carlo approximation to the variance of the estimates. The other is to use response surface methodology (RSM). The iterative procedures are fully automatic, with user-specified options to control the simulation and convergence criteria.

In the following we first briefly describe the Monte Carlo likelihood and the implemented optimization procedure; then we demonstrate some major features of the package **mclcar** through examples of Gaussian, Binomial and Poisson data.

2. The Monte Carlo likelihood estimation

The Monte Carlo likelihood is an importance sampling approximation to the log-likelihood ratio $\log L(\theta; y)/L(\psi; y)$, where $\theta, \psi \in \Theta$ and ψ is the parameter value used in the importance distribution. The likelihood is usually in the following forms

$$L(\theta; y) = f_\theta(y) = \frac{1}{c(\theta)} h_\theta(y) \tag{4}$$

$$L(\theta; y) = \int f_\theta(Y = y, Z)\, \mathrm{d}Z \tag{5}$$

where in (4) the likelihood is the product of a normalising constant $C(\theta)$ and the un-normalised density $h_\theta(y)$ and in (5), $f_\theta(Y, Z)$ is the joint density of the observed data Y and the unobserved or latent variable Z. The corresponding Monte Carlo likelihoods are

$$\hat{\ell}_\psi^{s_1}(\theta) = \log \frac{h_\theta(y)}{h_\psi(y)} - \log \frac{1}{s_1} \sum_i^{s_1} \frac{h_\theta(Y_i)}{h_\psi(Y_i)} \tag{6}$$

$$\hat{\ell}_\psi^{s_2}(\theta) = \log \frac{1}{s_2} \sum_i^{s_2} \frac{f_\theta(y, Z^{*(i)})}{f_\psi(y, Z^{*(i)})} \tag{7}$$

where Y_i are s_1 samples from $f_\psi(y)$ and $Z^{*(i)}$ are s_2 samples from $f_\psi(Z|Y = y)$.

Given a chosen ψ, the log-likelihood ratio differs from the log-likelihood by a constant and thus can be maximized to find the MLE. The Monte Carlo MLE (MC-MLE) is defined to be

$$\hat{\theta}_\psi^s = \arg\max_{\theta \in \Theta} \hat{\ell}_\psi^s(\theta) \tag{8}$$

Ideally, we would like to find the MC-MLE by directly maximising the Monte Carlo likelihood; however this might become infeasible as the Monte Carlo error increases as the distance between ψ and θ becomes large. When an reasonable initial value of ψ is available, iterations can be done to improve the accuracy by using the MC-MLE obtained from the current step as the ψ in the next step until the different between the two reaches some tolerance. When there is no good enough initial value, we put some constraints on the updating step in each iteration based on the sample variance estimates for the Monte Carlo errors.

The constrained iterative procedure is implemented by functions `OptimMCL` and `rsmMCL`: the former directly optimise the Monte Carlo likelihood and update in each iteration such that the error of the Monte Carlo likelihood at the new ψ does not exceed some tolerance; while the later use the response surface methodology (RSM, Box and Draper (2007)) in maximisation combined with the constraints in the *steepest ascent analysis* for updating the system. A quick introduction of implementing the RSM in R can be found in Lenth (2009).

3. Examples

Install and load the package.

```
> library(mclcar)
```

3.1. Simulate Data

The package provides several functions to simulate samples from a given direct CAR model in (2) or a GLM model with CAR latent variables as in (3). For example, we can generate CAR errors on a 10×10 torus with spatial coefficient $\rho = 0.2$ and precision $\tau^2 = 1/sigma^2 = 2/3$ by the following

```
> set.seed(33)
> n.torus <- 10
> rho <- 0.2
> sigma <- 1
> prec <- 1/sigma
> beta <- c(1, 1)
> XX <- cbind(rep(1, n.torus^2), sample(log(1:n.torus^2)/5))
> mydata1 <- CAR.simTorus(n1 = n.torus, n2 = n.torus, rho = rho, prec = prec)
```

The simulated data is a vector of length 100. When the spatial weight matrix W is supplied, the CAR errors can be generated by

```
> Wmat <- mydata1$W
> mydata2 <- CAR.simWmat(rho = rho, prec = prec, W = Wmat)
```

Then with the above we can generate observations from a linear model with CAR error

```
> y <- XX %*% beta + mydata1$X
> mydata1$data.vec <- data.frame(y=y, XX[,-1])
> mydata3 <- CAR.simLM(pars = c(0.1, 1, 2, 0.5), data = mydata1)
```

For the direct CAR models we can do exact evaluation of the likelihood for an object of the same struture as `mydata1`

```
> str(mydata1)

List of 3
 $ W       : num [1:100, 1:100] 0 1 0 0 0 0 0 0 0 1 ...
 $ X       : num [1:100] -0.8692 -1.3844 0.533 0.0556 -1.3113 ...
 $ data.vec:'data.frame': 100 obs. of  2 variables:
  ..$ y       : num [1:100] 0.892 0.353 2.307 1.956 0.57 ...
  ..$ XX....1.: num [1:100] 0.761 0.738 0.774 0.9 0.881 ...

> #### evaluate the log-likelihood
> ## without supplying lamdab -- the eigenvalues of W
> loglik.dCAR(pars = c(0.1, 1, 0.9, 2.1), data = mydata1)

[1] -163.7614

> ## with lamda
> lambda <- eigen(mydata1$W, symmetric = TRUE, only.values=TRUE)$values
> mydata1$lambda <- lambda
> loglik.dCAR(pars = c(0.1, 1, 0.9, 2.1), data = mydata1)

[1] -163.7614

> ## evaluate the profile log-likelihood of rho
> ploglik.dCAR(rho = 0.1, data = mydata1)

[1] -65.76634

> ## given rho = 0.1, find the least square estimates for beta and sigma
> get.beta.lm(rho = 0.1, data = mydata1)

                [,1]
(Intercept) 0.3631509
XX....1.    2.3623263

> sigmabeta(rho = 0.1, data = mydata1)

    sigma     beta1     beta2
1.3422579 0.3631509 2.3623263

> ## find the maximum pseudo-likelihood estimates
> (psi1 <- mple.dCAR(data = mydata1))

[1] 0.1855299 1.2390929 0.3595575 2.3672658
```

We can also generate Binomial or Poisson observations with CAR latent variables

```
> mydata4 <- CAR.simGLM(method="binom", n=c(10,10),
+                       pars = c(rho, sigma, beta),
+                       Xs=XX, n.trial = 5)
> mydata5 <- CAR.simGLM(method="poisson", n=c(10, 10),
+                       pars = c(rho, sigma, beta), Xs=XX)
> str(mydata5)

List of 10
 $ rho    : num 0.2
 $ sigma  : num 1
 $ beta   : num [1:2] 1 1
 $ y      : int [1:100] 7 41 18 31 6 8 4 10 6 4 ...
 $ covX   : num [1:100, 1:2] 1 1 1 1 1 1 1 1 1 1 ...
 $ W      : num [1:100, 1:100] 0 1 0 0 0 0 0 0 1 ...
 $ Z.true : num [1:100] 0.281 2.027 1.033 1.297 -0.129 ...
 $ eta    : num [1:100, 1] 2.04 3.77 2.81 3.2 1.75 ...
 $ Emean  : num [1:100, 1] 7.71 43.18 16.57 24.45 5.77 ...
 $ n.trial: num 1
```

The result is a list containing all the information of the GLM model used in the simulation.

3.2. Prepare the Monte Carlo samples

Before evaluating the Monte carlo likelihood, we need to generate Monte Carlo samples from the importance sampling distribution. For direct CAR models, we only need to specify the number of Monte Carlo samples by `n.samples` and the value of `psi` to be used in the importance sampler.

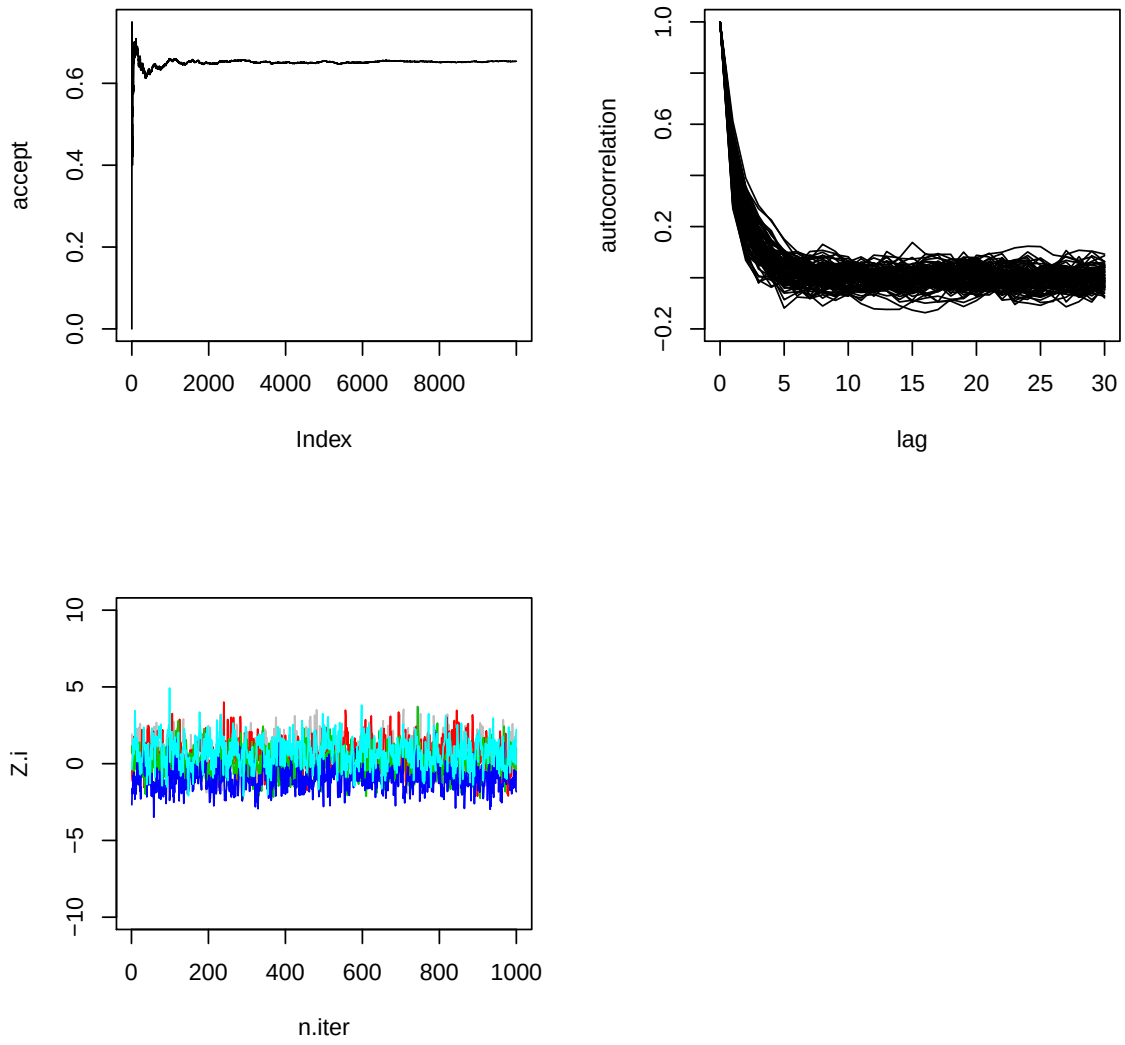
```
> #### Prepare the Monte Carlo samples for the direct CAR models
> mcdata1 <- mcl.prep.dCAR(psi = psi1, n.samples = 500, data = mydata1)
```

Monte Carlo samples for the GLM with latent CAR models need to be simuated by using MCMC algorithms and it can be done by `postZ`. Two major MCMC algorithms, the radom walk Metropolis Hastings (`"rwmh"`) and Metropolis Adjusted Langevin alogrithm (`"mala"`), are implemented in the function and the parameters in the algorithm can be controlled by `mcmc.control`. More details can be found in the package documentaion.

```
> Z.S0 <- CAR.simWmat(psi1[1], 1/psi1[2], mydata4$W) # initial value
> mc.cons <- list(method = "mala") # control the MCMC algorithm
> simZy <- postZ(data = mydata4, Z.start = Z.S0, psi = psi1,
+                family = "binom", mcmc.control = mc.cons,
+                plots = TRUE) # diagnostic plots for the MCMC
```

Similar to the direct CAR models, the function `mcl.prep.glm` provide as an wrapper for `postZ` for preparing the Monte Carlo samples.

```
> mc.BinData <- mcl.prep.glm(data = mydata4, family = "binom", psi = psi1,
+                            pilot.plot = FALSE, plot.diag = TRUE)
```



3.3. Evaluate the Monte Carlo likelihood

Now with the prepared Monte Carlo samples in the objects `mcdata1`, `mc.BinData` and `mc.PoiData`, we can evaluate the Monte Carlo likelhoods and get variance estimations.

```
> ## the Monte Carlo likelihoods at the true value
> pars.t <- c(rho, sigma, beta)
> mcl.dCAR(pars.t, data = mydata1, simdata = mcdata1, Evar = TRUE)

[1] -4.58463275  0.06991544
```

```
> mcl.glm(pars.t, family = "binom",  mcdata = mc.BinData, Evar = FALSE)

[1] 1.751072

> ## When Evar = TRUE the function returns the Monte Carlo likelihood,
> ## an variance estimate of the Monte Carlo likelihood
```

3.4. The iterative maximization procedure

The Monte Carlo likelihood can be directly maximised to find the Monte Carlo MLE in an iterative procedure that uses the Monte Carlo variance estimate as an constraints on step size of the update in each iteration. The function `OptimMCL` implement such procedure and users can define the starting value of the ψ in the importance sampler and control the parameters of the MCMC algorithm as before by setting `mc.control`.

The function has an aditional option `control` that can be defined by users to control the number of iterations and etc. in the iterative procedure.

```
> iter.mcmle <- OptimMCL(data = mydata1, psi0 = psi1, family = "gauss",
+                        control = list(mc.var = FALSE, verbose = FALSE))
> summary(iter.mcmle, family = "gauss", mc.covar=FALSE)

$MC.mle
              sigma     beta1     beta2
0.1720960 1.2552967 0.3601331 2.3664747

$N.iter
[1] 3

$total.time
elapsed
 58.129

$convergence
[1] TRUE

$hessian
NULL

$mc.covar
NULL

$mc.samples
[1]  500 1000

> ## similar syntax for Binomial data but take longer time to run
```

```
> ## iter.mcmle.b <- OptimMCL(data = mydata4, psi0 = psi1, family = "binom",
> ##                          control = list(mc.var = TRUE, verbose = TRUE))
```

By default `verbose` = TRUE in the list of `control` and the function print out a few lines of summary information of the current iteration.

The summary of `OptimMCL` prints the Monte Carlo MLE found in the final iteration, the corresponding Hessian matrix and a few other informations about the iteration.

3.5. The RSM optimisation procedure

We can also use the response surfance methodology to find the Monte Carlo MLE and approximate the likelihood surface around the MLE by `rsmMCL`. The function has usage as `OptimMCL` and users can specify the size of the design region through `K` and number of design points `n01, n02` in the `control` list for the response surface design.

For the data from a direct CAR models, we can choose to compare the exact likelihood surface and the approximated surface in each iteration by setting `eval = TRUE` and the range of the ρ, σ^2 and the number of grid points to be evaluated in each coordination. For example,

```
> exacts = list(eval = TRUE, rho = c(-0.25, 0.25),
+ sigma = c(0.5, 2), length = 100)
```

```
> rsm.mcmle1 <- rsmMCL(data = mydata1, psi0 = c(-0.1, sigmabeta(-0.1, mydata1)),
+                      family = "gauss",
+                      control = list(n.iter= 10, trace.all = TRUE))
```

The same thing can be done for the GLM model with latent CAR variables, except that there is no exact likelihood values to be compared.

```
> rsm.mcmle2 <- rsmMCL(data = mydata5, psi0 = c(0, 1, 2, 2), family = "poisson",
+                     control = list(n.iter = 20, trace.all = TRUE))
```

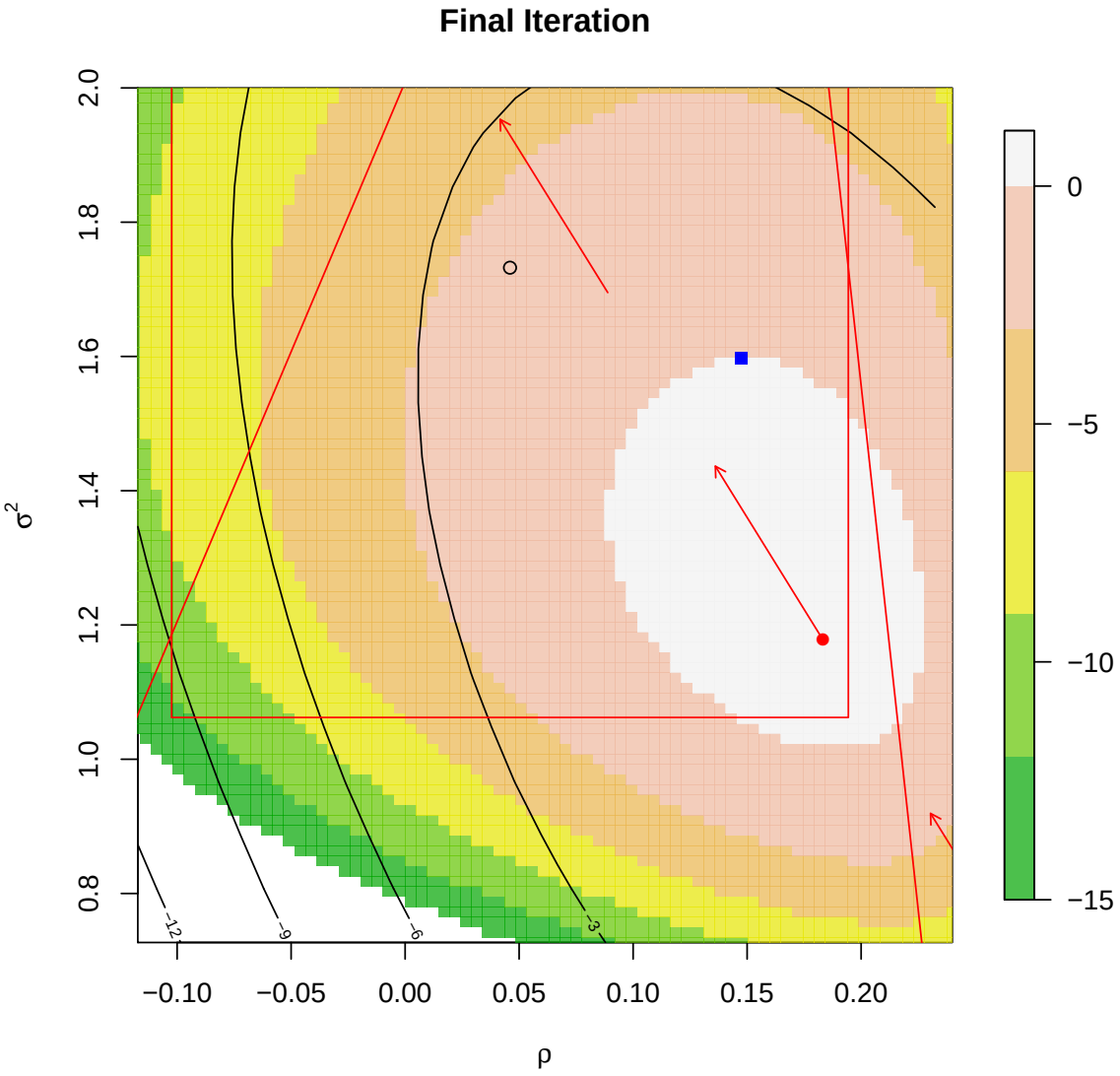
The result can be printed by `summary`.

```
> summary(rsm.mcmle1, family = "gauss", mc.covar=FALSE)

$MC.mle
[1] 0.1832807 1.1779017 0.6129708 2.0189213

$N.iter
[1] 2

$total.time
elapsed
 12.402
```

```
$convergence
[1] TRUE

$hessian
NULL

$mc.covar
NULL

$mc.samples
[1]  500 1000
```

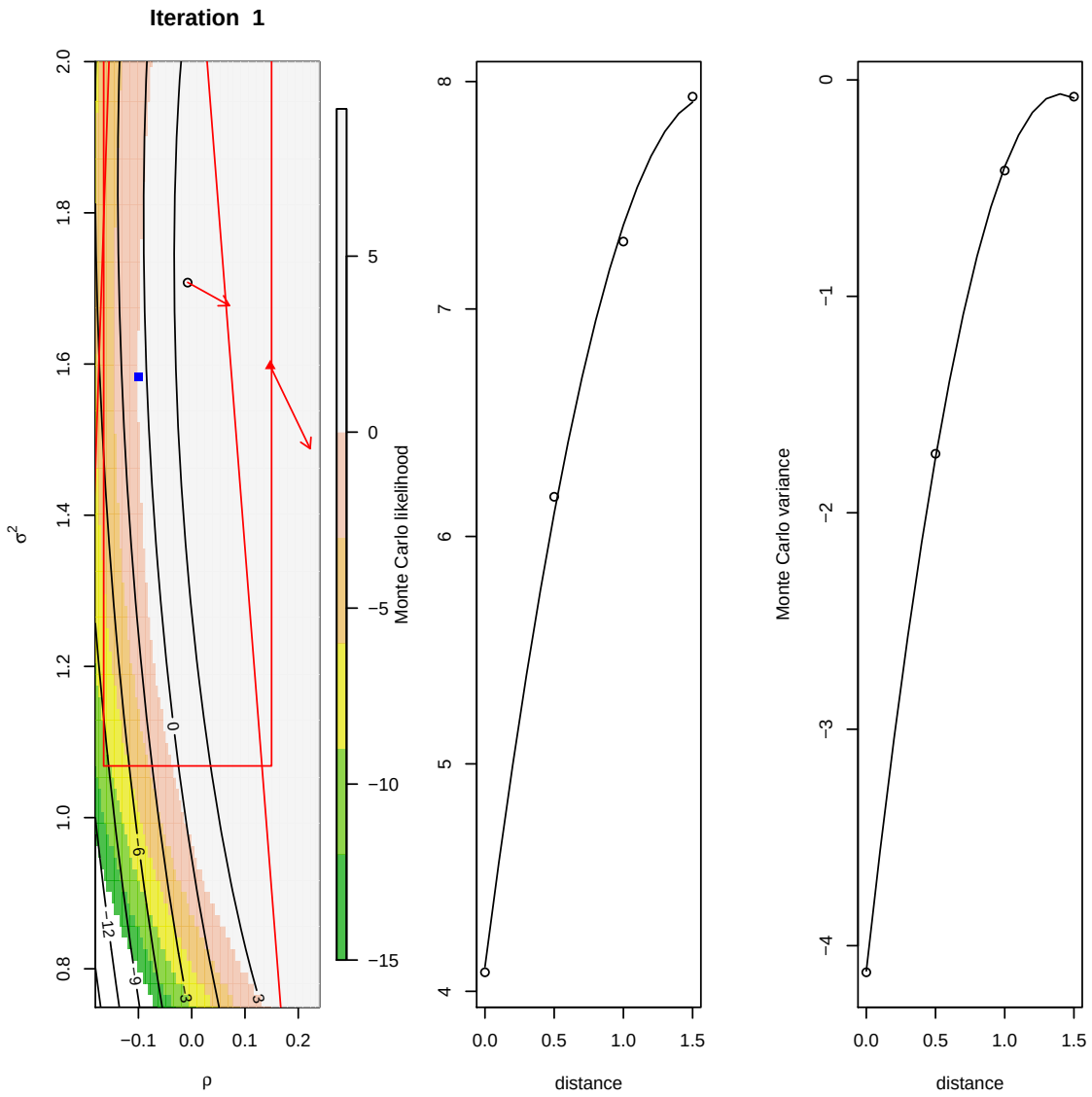
The final fitted response surface can be plotted by setting `trace.all = FALSE`.

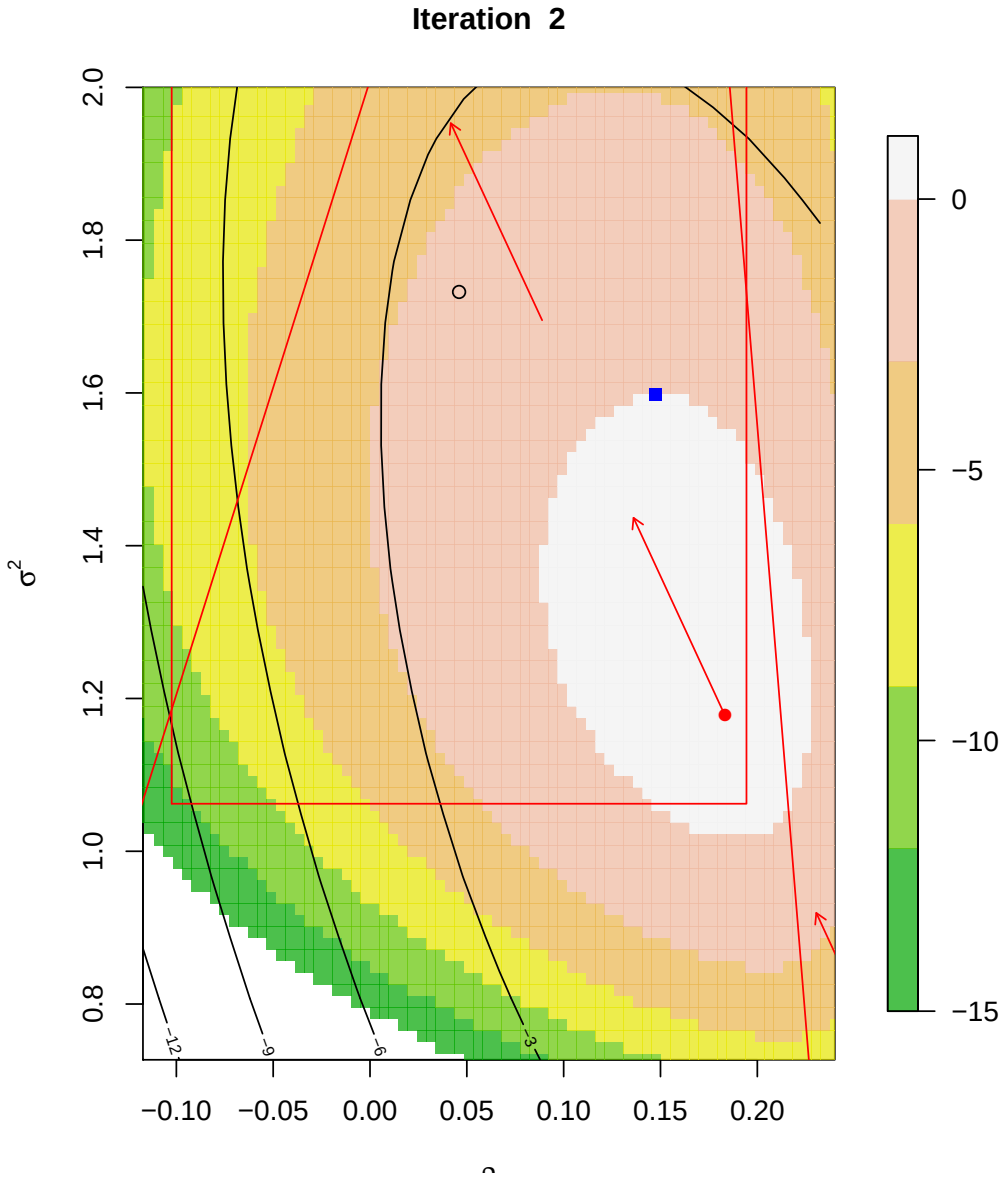
```
> plot(rsm.mcmle1, family = "gauss", trace.all = FALSE)
```



Otherwise, the entire evolution of the response surface is shown.

```
> plot(rsm.mcmle1, family = "gauss")

Path of steepest ascent from ridge analysis:
Path of steepest ascent from ridge analysis:
```


Iteration 1
σ²
ρ
Monte Carlo likelihood
distance
Monte Carlo variance
distance



References

Box GEP, Draper NR (2007). *Response Surfaces, Mixtures, and Ridge Analyses*. Wiley, New York.

Geyer CJ, Thompson EA (1992). "Constrained Monte Carlo maximum likelihood for dependent data." *Journal of the Royal Statistical Society. Series B (Methodological)*, **54**(3), 657–699.

Lenth RV (2009). "Response-Surface Methods in R, Using `rsm`." *Journal of Statistical Software*, **32**(7), 1–17. http://www.jstatsoft.org/v32/i07/.

Affiliation:

Zhe Sha
Department of Statistics
University of Oxford
24-29 St Giles', Oxford
OX1 3LB, UK
E-mail: zhesha1006@gmail.com