OxWaSP Module 1: Adaptive MCMC

Virginia Aglietti Tamar Loach

October 19, 2016

1 Introduction to Adaptive MCMC - the AM algorithm

MCMC algorithms allow sampling from complicated, high-dimensional distributions. Choice of the proposal distribution (from which samples are taken in an attempt to approximate sampling from the target distribution π) determines the ability of the algorithm to explore the parameter space fully and hence draw a good sample. Adaptive MCMC algorithms tackle this challenge by using samples already generated to learn about the target distribution; they push this knowledge back to the choice of proposal distribution iteratively.

This project explores adaptive MCMC algorithms existing in the literature that use covariance estimators to improve convergence to a target distribution supported on a subset of $\mathbb{R}^d$. In this schema we learn about the target distribution π through estimation of its correlation structure from the MCMC samples. We use this correlation structure to improve our estimate of the target.

We first implement an adaptive MCMC algorithm AM (Haario et al., 2001) which is a modification of the random walk Metropolis-Hastings algorithm. In AM the proposal distribution is updated at time t to be a normal distribution centered on the current point X_{t-1} with covariance $C_t(X_0, ..., X_{t-1})$ that depends on the the whole history of the chain. The use of historic states means the resulting chain is non-markovian, and reversibility conditions are not satisfied. Haario et al show that, with a small update to the usual Metropolis-Hastings acceptance probability, the right ergodic properties and correct simulation of the target distribution none the less remain. The probability with which to accept candidate points in the chain becomes:

$$\alpha(X_{t-1}, Y) = \min\left(1, \frac{\pi(Y)}{\pi(X_{t-1})}\right)$$

With C_t given by:

$$C_t = s_d \text{cov}(X_0, ..., X_{t-1}) + s_d \epsilon I_d$$

Here cov() is the usual empirical covariance matrix, and the parameter $s_d = \frac{2.4^2}{d}$ (Gelman et al., 1996). ϵ is chosen to be very small compared to the subset of $\mathbb{R}^d$ upon which the target function is supported. The AM algorithm is computationally feasible due to recursive updating of the covariance matrix on acquisition of each new sample through the relation:

$$C_{t+1} = \frac{t-1}{t} C_t + \frac{s_d}{t} (t\bar{X}_{t-1}\bar{X}_{t-1}^T - (t+1)\bar{X}_t\bar{X}_t^T + X_t X_t^T + \epsilon I_d)$$

with the mean calculated recursively by:

$$\bar{X}_{t+1} = \frac{t\bar{X}_t + X_{t+1}}{t+1}$$

Because of the instability of the covariance matrix, to implement the adaptivity we first run the algorithm with no change to the covariance of the proposal distribution. The adaptation starts at a user defined point in time, and until this time the covariance of the proposal is chosen to represent our best knowledge of the target distribution.

2 An example - testing the AM algorithm

We now numerically test the AM algorithm. We have used two different target distributions: a correlated Gaussian distribution $N(0, \Sigma)$ and a "banana"-shaped distribution ((Roberts and Rosenthal, 2009)) given by:

$$f_B(x_1, ..., x_d) \propto \exp\left[-x_1^2/200 - \frac{1}{2}\left(x_2 + Bx_1^2 - 100B\right)^2 - \frac{1}{2}\left(x_3^2 + x_4^2 + ... + x_d^2\right)\right]$$

$B > 0$ is the "bananicty" constant (set to 0.1 throughout) and d is the dimension. We have chosen the correlated Guassian distribution as targetting this demonstrates how the use of empirical covariance improves convergence - we learn the target's covariance as we move through steps of the MCMC. The banana-shaped distribution is an additional example with an irregular shape. We use this to test the ability of the markov chain to fully explore the state space with and without adaption. We first run our implementation of the AM algorithm targetting $N(0, \Sigma)$ with

```
d = 8
n_iter = 200     #  Total number of iterations
x_1 = rep(5,d)       #  Vector of inital values
t_adapt = 50      #  When to start adapting
adapt = "None"       #  Choose the type of adaptation.

X_MH = mcmc(target = pi_norm_corr,
         n_iter = n_iter,
         x_1 = x_1,
         adapt=adapt)
```

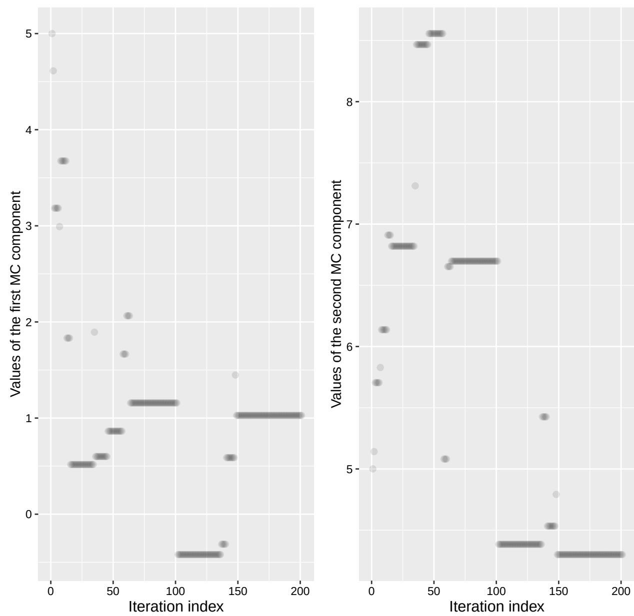


Figure 1: This is the first graph

```
## Running MCMC targeting  pi_norm_corr with  200 iterations.
## Adaptation algorithm is:  None .
## Covariance matrix estimator is:  Sample covariance .
## MCMC finished in 0.1594176 seconds.
## The last acceptance rate was: 0.085
```

```
adapt = "AM"       #  Choose the type of adaptation.
cov_estimator="Sample covariance"    #  Choose the type of covariance matrix estimator.

X_AM = mcmc(target = pi_norm_corr,
         n_iter = n_iter,
         x_1 = x_1,
         adapt=adapt,
         t_adapt = t_adapt,
         cov_estimator=cov_estimator
         )

## Running MCMC targeting  pi_norm_corr with  200 iterations.
## Adaptation algorithm is:  AM .
## Covariance matrix estimator is:  Sample covariance .
## MCMC finished in 2.17844 seconds.
```

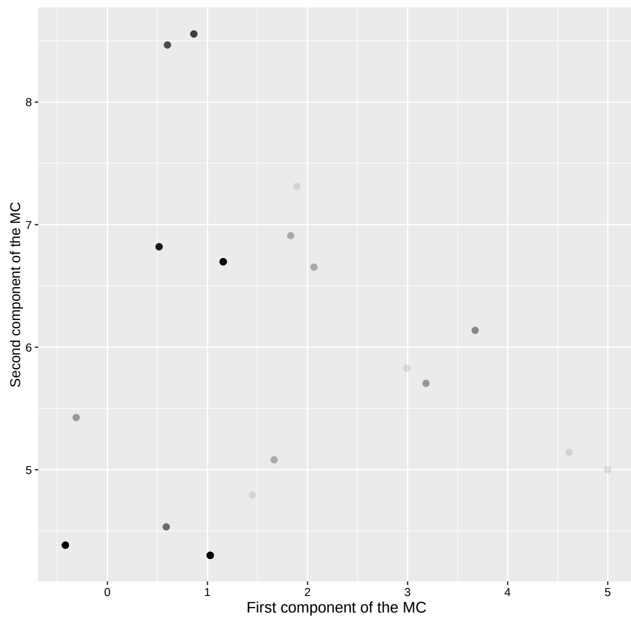


Figure 2: This is the first graph

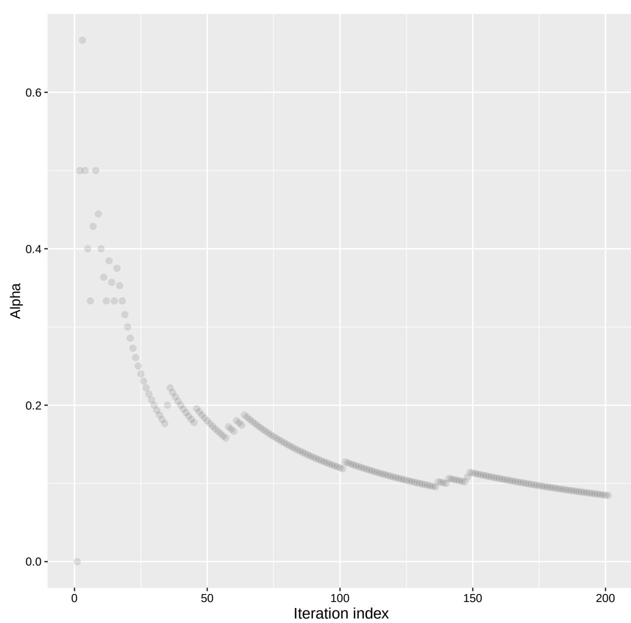


Figure 3: This is the first graph

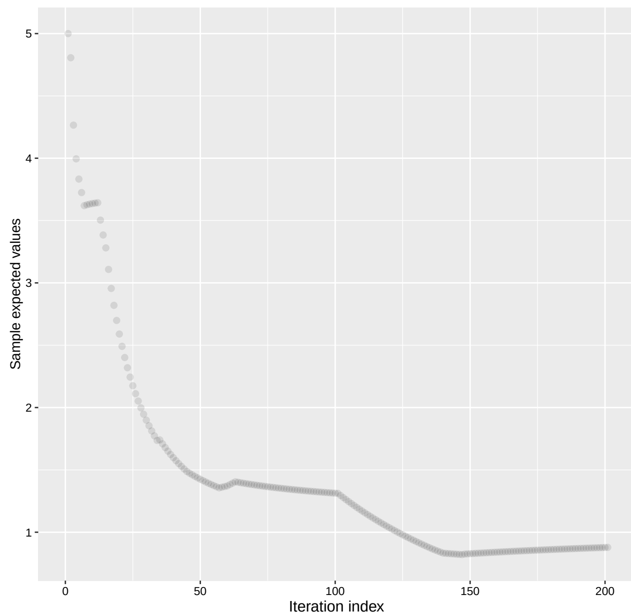
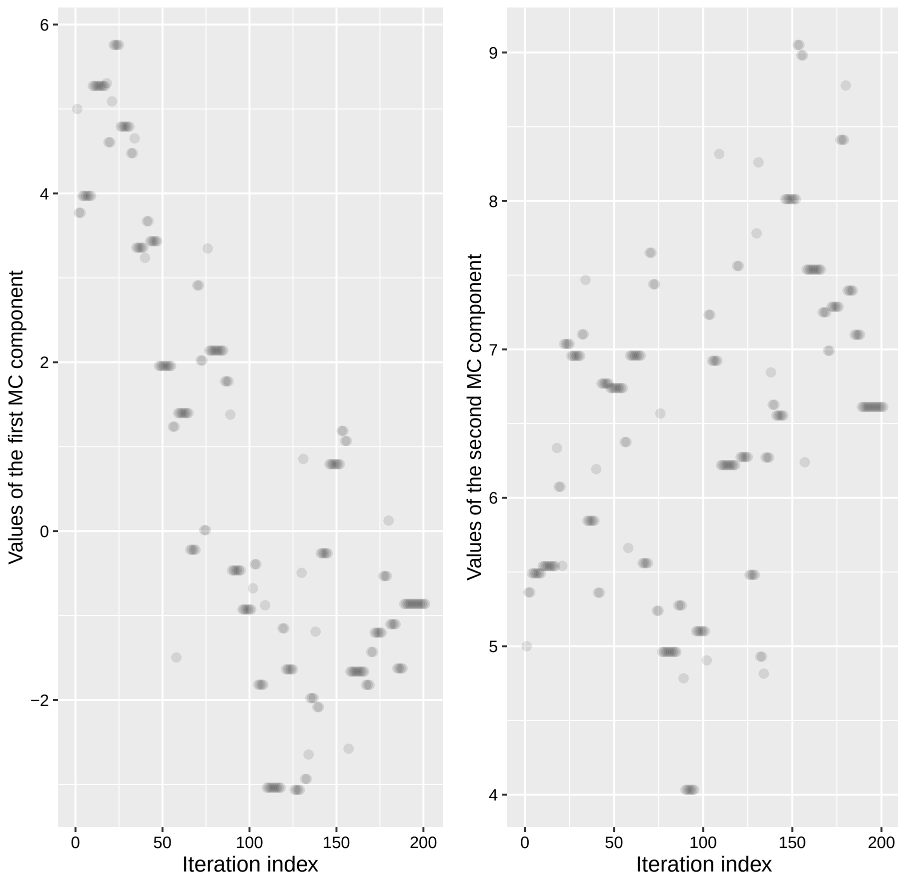
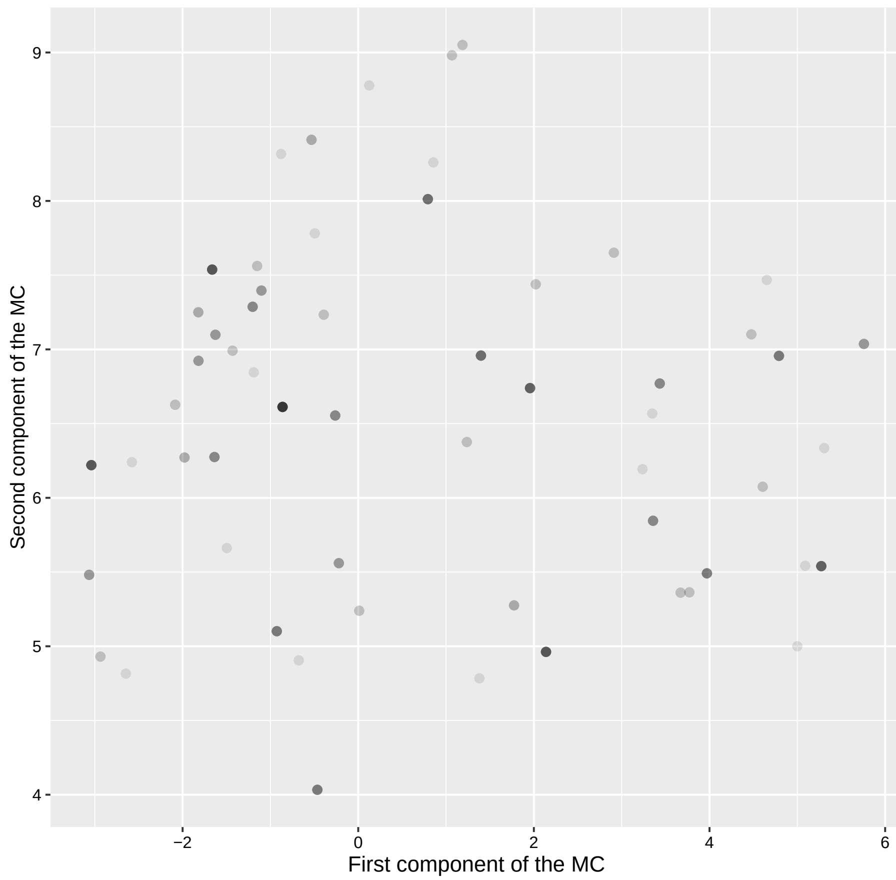


Figure 4: This is the first graph

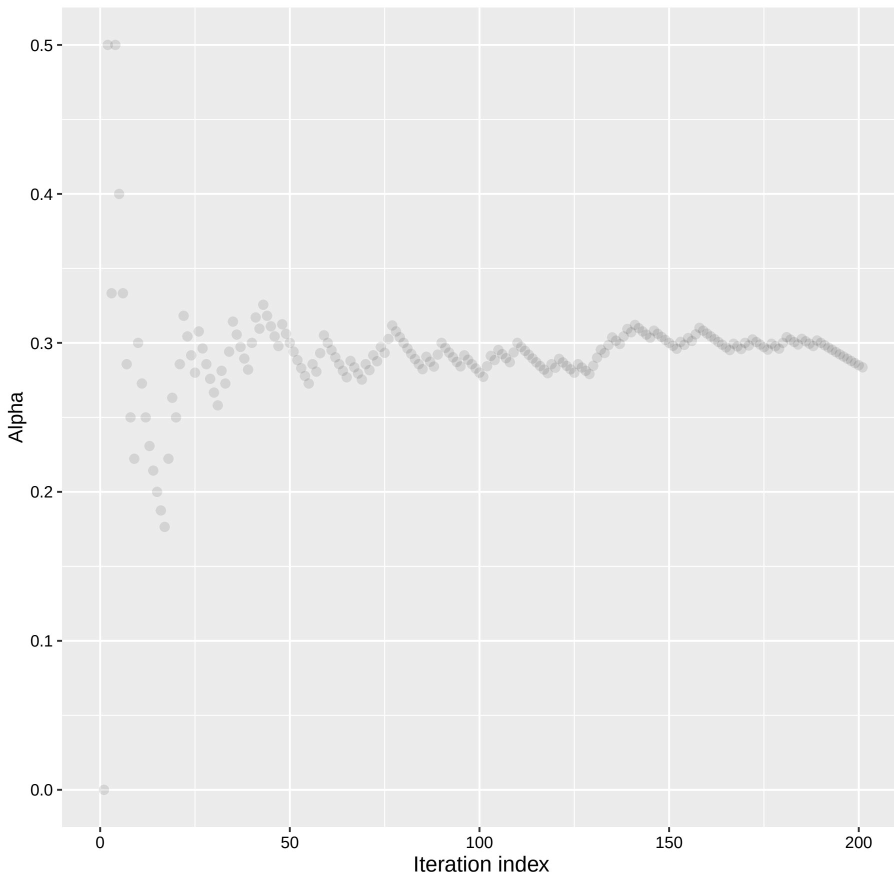
```
## The last acceptance rate was: 0.285

plot6=plotIterations(
                    X_AM$X[,1],
                    n_iter,
                    title="Values of the first MC component"
                    )
plot7=plotIterations(
                    X_AM$X[,2],
                    n_iter,
                    title="Values of the second MC component"
                    )
grid.arrange(plot6, plot7, ncol=2)
```



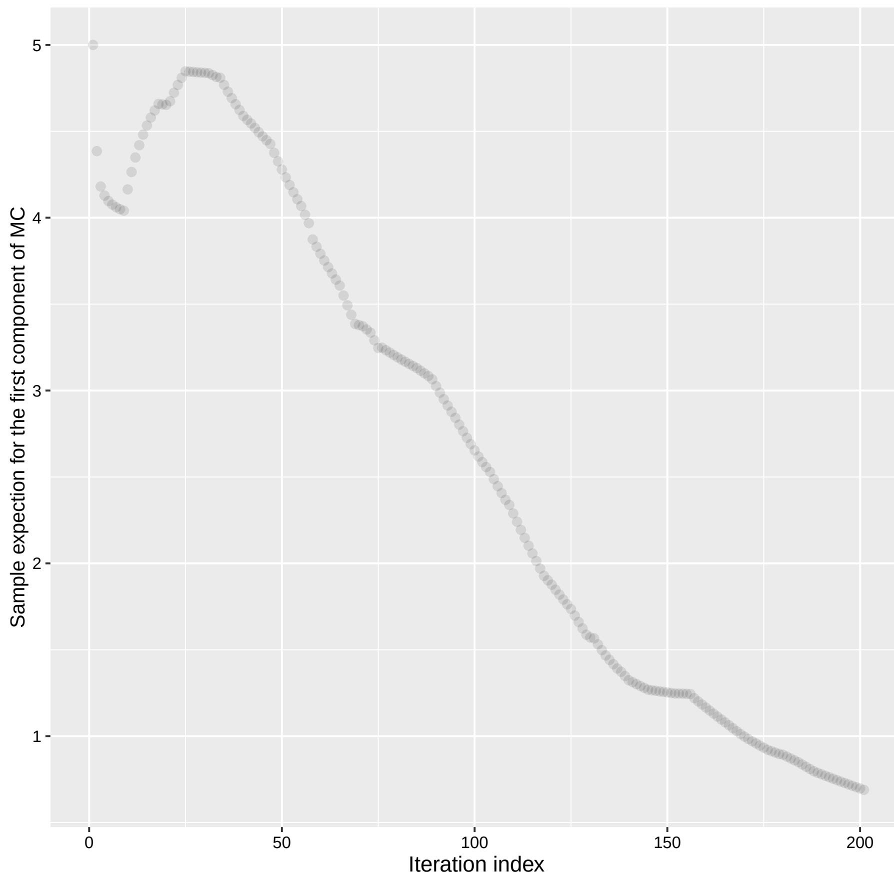

```
plot8=plotComponents(
                    X_AM$X[,1],
                    X_AM$X[,2],
                    Xtitle = "First component of the MC",
                    Ytitle = "Second component of the MC"
                    )
plot8
```




```r
plot9=plotIterations(X_AM$acceptance_rates, n_iter, "Alpha")
plot9
```




```
plot10=plotIterations(X_AM$sample_mean[,1], n_iter, "Sample expection for the first componen
plot10
```

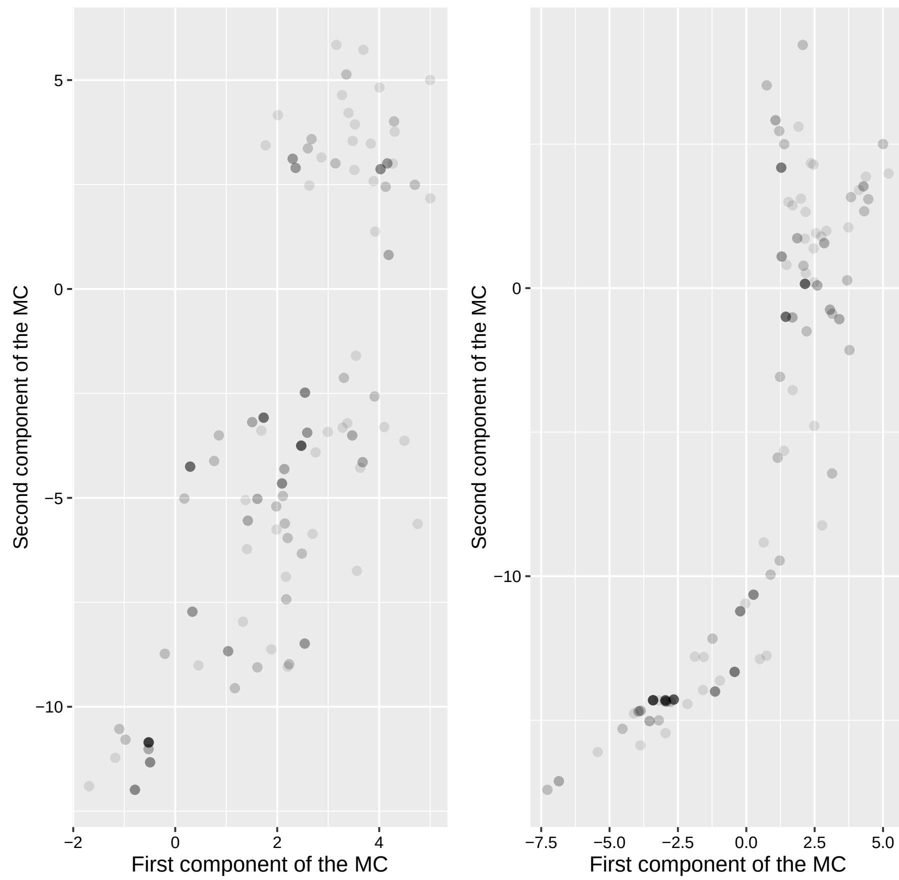



```
x_1 = rep(5,2)       #  Vector of inital values
target = pi_banana


X_MH_banana = mcmc(target = target,
         n_iter = n_iter,
         x_1 = x_1,
         adapt="None"
         )
```

```
## Running MCMC targeting  target with  200 iterations.
## Adaptation algorithm is:  None .
## Covariance matrix estimator is:  Sample covariance .
## MCMC finished in 0.07853293 seconds.
## The last acceptance rate was: 0.44
```

```
plot11=plotComponents(
```

```
                    X_MH_banana$X[,2],
                    X_MH_banana$X[,1],
                    Xtitle = "First component of the MC",
                    Ytitle = "Second component of the MC"
                    )

X_AM_banana = mcmc(target = target,
         n_iter = n_iter,
         x_1 = x_1,
         t_adapt = t_adapt,
         adapt="AM"
         )

## Running MCMC targeting  target with  200 iterations.
## Adaptation algorithm is:  AM .
## Covariance matrix estimator is:  Sample covariance .
## MCMC finished in 1.830635 seconds.
## The last acceptance rate was: 0.42

plot12=plotComponents(
                    X_AM_banana$X[,2],
                    X_AM_banana$X[,1],
                    Xtitle = "First component of the MC",
                    Ytitle = "Second component of the MC"
                    )
grid.arrange(plot11, plot12, ncol=2)
```

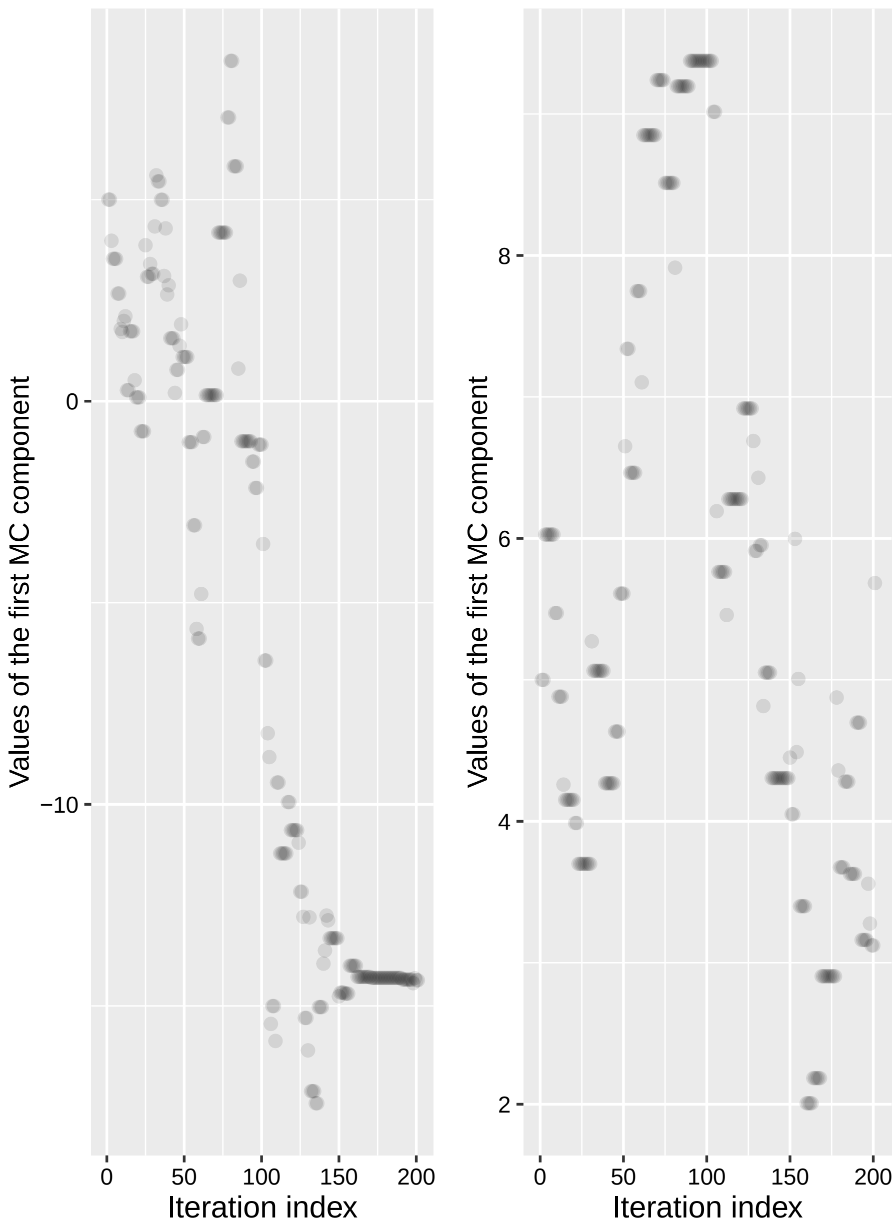



```
x_1 = rep(5,8)       #  Vector of inital values
target = pi_banana8


X_AM2_banana = mcmc(target = target,
          n_iter = n_iter,
          x_1 = x_1,
          adapt="AM2"
          )
```

```
## Running MCMC targeting  target with  200 iterations.
## Adaptation algorithm is:  AM2 .
## Covariance matrix estimator is:  Sample covariance .
## MCMC finished in 0.04424238 seconds.
## The last acceptance rate was: 0.275
```

```
plot13=plotIterations(
```

```
                    X_AM_banana$X[,1],
                    n_iter,
                    title="Values of the first MC component"
                    )
plot14=plotIterations(
                    X_AM2_banana$X[,1],
                    n_iter,
                    title="Values of the first MC component"
                    )

grid.arrange(plot13, plot14, ncol=2)
```




```
x_1 = rep(5,8)       #  Vector of inital values
cov_estimator1="Shrinkage estimator"
cov_estimator2="Thresholding estimator"
target = pi_banana8
```

```
X_MH_banana = mcmc(target = target,
         n_iter = n_iter,
         x_1 = x_1,
         adapt="None"
         )

## Running MCMC targeting  target with  200 iterations.
## Adaptation algorithm is:  None .
## Covariance matrix estimator is:  Sample covariance .
## MCMC finished in 0.09531641 seconds.
## The last acceptance rate was: 0.09

X_AM_banana = mcmc(target = target,
         n_iter = n_iter,
         x_1 = x_1,
         t_adapt = t_adapt,
         adapt="AM"
         )

## Running MCMC targeting  target with  200 iterations.
## Adaptation algorithm is:  AM .
## Covariance matrix estimator is:  Sample covariance .
## MCMC finished in 2.15562 seconds.
## The last acceptance rate was: 0.15

X_AM_banana_sh= mcmc(target = target,
         n_iter = n_iter,
         x_1 = x_1,
         adapt="AM",
         t_adapt = t_adapt,
         cov_estimator=cov_estimator1
         )

## Running MCMC targeting  target with  200 iterations.
## Adaptation algorithm is:  AM .
## Covariance matrix estimator is:  Shrinkage estimator .Estimating optimal shrinkage intens
##
##
## MCMC finished in 2.563684 seconds.
## The last acceptance rate was: 0.265

X_AM2_banana_sh = mcmc(target = target,
         n_iter = n_iter,
         x_1 = x_1,
         adapt="AM2",
         cov_estimator=cov_estimator1
         )
```

```
## Running MCMC targeting  target with  200 iterations.
## Adaptation algorithm is:  AM2 .
## Covariance matrix estimator is:  Shrinkage estimator .
## MCMC finished in 0.04282379 seconds.
## The last acceptance rate was: 0.275
```

```
X_AM_banana_th= mcmc(target = target,
         n_iter = n_iter,
         x_1 = x_1,
         adapt="AM",
         t_adapt = t_adapt,
         cov_estimator=cov_estimator2
         )
```

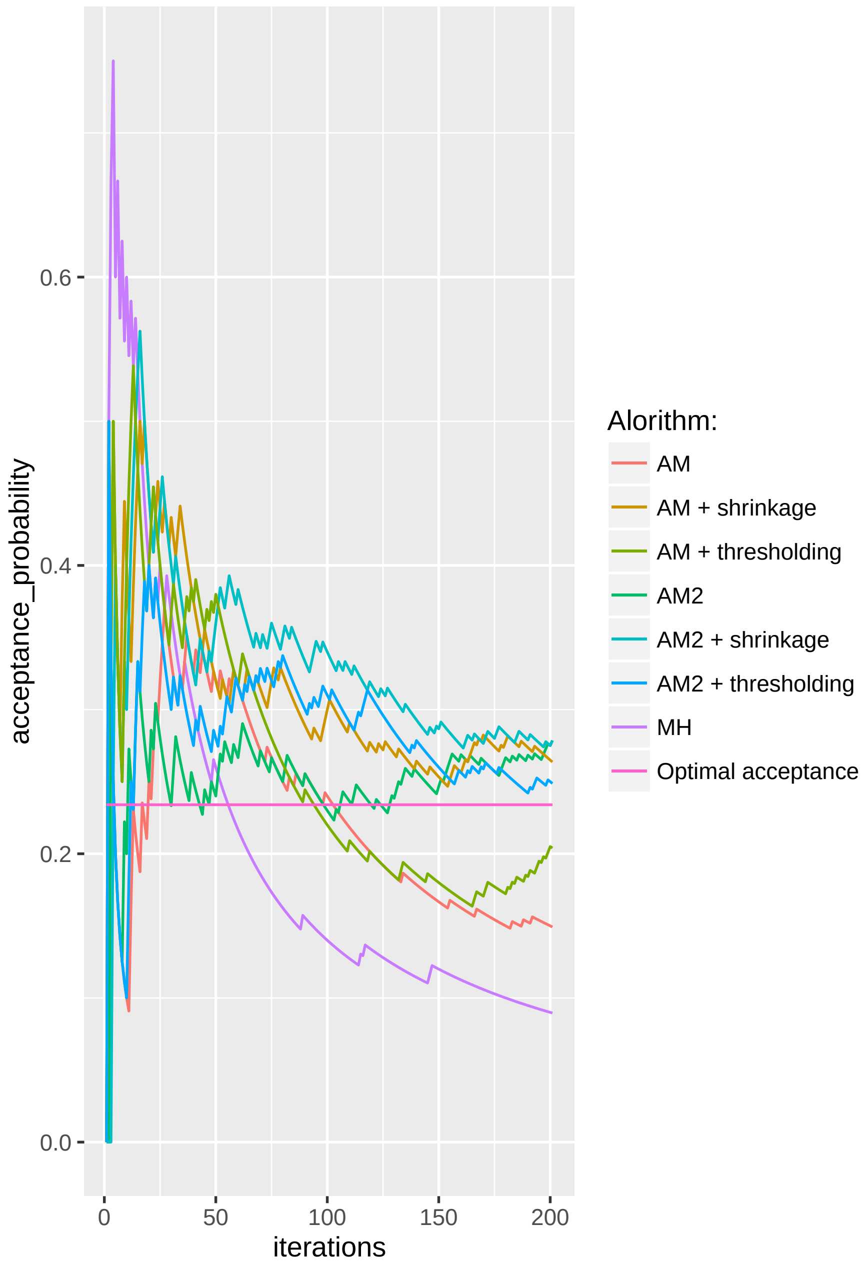
```
## Running MCMC targeting  target with  200 iterations.
## Adaptation algorithm is:  AM .
## Covariance matrix estimator is:  Thresholding estimator .
## MCMC finished in 0.2850504 seconds.
## The last acceptance rate was: 0.205
```

```
X_AM2_banana_th = mcmc(target = target,
         n_iter = n_iter,
         x_1 = x_1,
         adapt="AM2",
         cov_estimator=cov_estimator2
         )
```

```
## Running MCMC targeting  target with  200 iterations.
## Adaptation algorithm is:  AM2 .
## Covariance matrix estimator is:  Thresholding estimator .
## MCMC finished in 0.05075908 seconds.
## The last acceptance rate was: 0.25
```

```
iteration=seq(from= 1, to=n_iter+1, by=1)
#data=data.frame(iteration_count=iteration, am_alpha=X_AM_banana_sh£acceptance_rates)


df1<-data.frame(iterations=iteration,acceptance_probability=X_MH_banana$acceptance_rates)
df2<-data.frame(iterations=iteration,acceptance_probability=X_AM_banana$acceptance_rates)
df3<-data.frame(iterations=iteration,acceptance_probability=X_AM2_banana$acceptance_rates)
df4<-data.frame(iterations=iteration,acceptance_probability=X_AM_banana_sh$acceptance_rates)
df5<-data.frame(iterations=iteration,acceptance_probability=X_AM2_banana_sh$acceptance_rates
df6<-data.frame(iterations=iteration,acceptance_probability=X_AM_banana_th$acceptance_rates)
df7<-data.frame(iterations=iteration,acceptance_probability=X_AM2_banana_th$acceptance_rates
df8<-data.frame(iterations=iteration,acceptance_probability=rep(0.234,n_iter+1))
```

```
ggplot(df1,aes(iterations,acceptance_probability))+geom_line(aes(color="MH"))+
      geom_line(data=df2,aes(color="AM"))+
      geom_line(data=df3,aes(color="AM2"))+
      geom_line(data=df4,aes(color="AM + shrinkage"))+
      geom_line(data=df5,aes(color="AM2 + shrinkage"))+
      geom_line(data=df6,aes(color="AM + thresholding"))+
      geom_line(data=df7,aes(color="AM2 + thresholding"))+
      geom_line(data=df8,aes(color="Optimal acceptance"))+
      labs(color="Alorithm:")
```




```
# all lines of the mean in the same plot with a legend
```

```
target=pi_norm_corr


X_MH = mcmc(target = target,
         n_iter = n_iter,
         x_1 = x_1,
         adapt=adapt)

## Running MCMC targeting  target with  200 iterations.
## Adaptation algorithm is:  AM .
## Covariance matrix estimator is:  Sample covariance .
## MCMC finished in 0.1930785 seconds.
## The last acceptance rate was: 0.36

X_AM = mcmc(target = target,
         n_iter = n_iter,
         x_1 = x_1,
         adapt=adapt,
         t_adapt = t_adapt,
         cov_estimator=cov_estimator
         )

## Running MCMC targeting  target with  200 iterations.
## Adaptation algorithm is:  AM .
## Covariance matrix estimator is:  Sample covariance .
## MCMC finished in 2.201681 seconds.
## The last acceptance rate was: 0.335

X_AM2 = mcmc(target = target,
         n_iter = n_iter,
         x_1 = x_1,
         adapt="AM2"
         )

## Running MCMC targeting  target with  200 iterations.
## Adaptation algorithm is:  AM2 .
## Covariance matrix estimator is:  Sample covariance .
## MCMC finished in 0.1027937 seconds.
## The last acceptance rate was: 0.35

X_AM_sh= mcmc(target = target,
         n_iter = n_iter,
         x_1 = x_1,
         adapt="AM",
         t_adapt = t_adapt,
         cov_estimator=cov_estimator1
         )
```

```
## Running MCMC targeting  target with  200 iterations.
## Adaptation algorithm is:  AM .
## Covariance matrix estimator is:  Shrinkage estimator .Estimating optimal shrinkage intens
##
##
## MCMC finished in 2.58172 seconds.
## The last acceptance rate was: 0.37
```

```
X_AM2_sh = mcmc(target = target,
         n_iter = n_iter,
         x_1 = x_1,
         adapt="AM2",
         cov_estimator=cov_estimator1
         )
```

```
## Running MCMC targeting  target with  200 iterations.
## Adaptation algorithm is:  AM2 .
## Covariance matrix estimator is:  Shrinkage estimator .
## MCMC finished in 0.09118152 seconds.
## The last acceptance rate was: 0.37
```

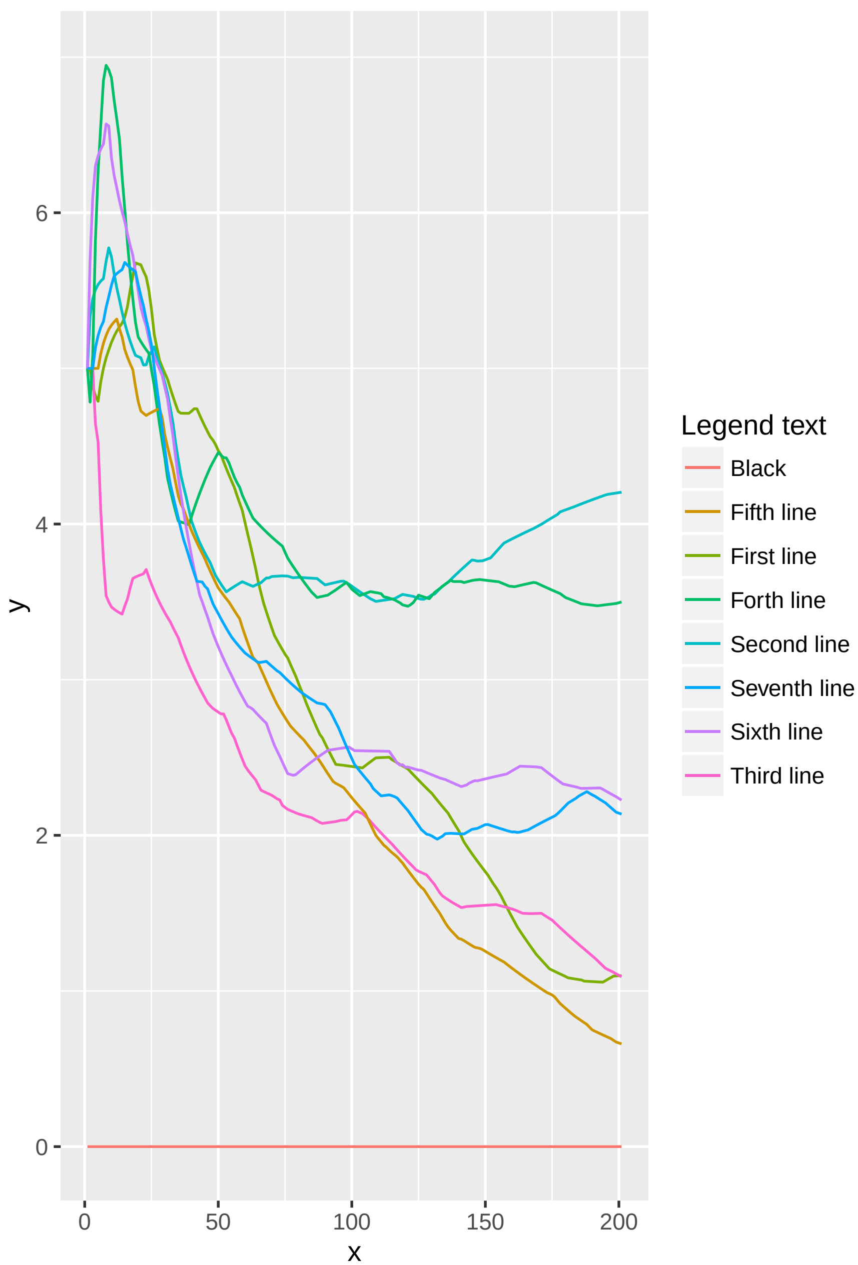
```
X_AM_th= mcmc(target = target,
         n_iter = n_iter,
         x_1 = x_1,
         adapt="AM",
         t_adapt = t_adapt,
         cov_estimator=cov_estimator2
         )
```

```
## Running MCMC targeting  target with  200 iterations.
## Adaptation algorithm is:  AM .
## Covariance matrix estimator is:  Thresholding estimator .
## MCMC finished in 0.297977 seconds.
## The last acceptance rate was: 0.35
```

```
X_AM2_th = mcmc(target = target,
         n_iter = n_iter,
         x_1 = x_1,
         adapt="AM2",
         cov_estimator=cov_estimator2
         )
```

```
## Running MCMC targeting  target with  200 iterations.
## Adaptation algorithm is:  AM2 .
## Covariance matrix estimator is:  Thresholding estimator .
## MCMC finished in 0.09100533 seconds.
## The last acceptance rate was: 0.415
```

```r
df1<-data.frame(x=iteration,y=X_MH$sample_mean[,1])
df2<-data.frame(x=iteration,y=X_AM$sample_mean[,1])
df3<-data.frame(x=iteration,y=X_AM2$sample_mean[,1])
df4<-data.frame(x=iteration,y=X_AM_sh$sample_mean[,1])
df5<-data.frame(x=iteration,y=X_AM2_sh$sample_mean[,1])
df6<-data.frame(x=iteration,y=X_AM_th$sample_mean[,1])
df7<-data.frame(x=iteration,y=X_AM2_th$sample_mean[,1])
df8<-data.frame(x=iteration, y=rep(0,n_iter+1))

ggplot(df1,aes(x,y))+geom_line(aes(color="First line"))+
      geom_line(data=df2,aes(color="Second line"))+
      geom_line(data=df3,aes(color="Third line"))+
      geom_line(data=df4,aes(color="Forth line"))+
      geom_line(data=df5,aes(color="Fifth line"))+
      geom_line(data=df6,aes(color="Sixth line"))+
      geom_line(data=df7,aes(color="Seventh line"))+
      geom_line(data=df8,aes(color="Black"))+
      labs(color="Legend text")
```




```
# all lines of the mean in the same plot with a legend
```

In order to test the increased performance of the AM algorithm, we have compared it with the classical Metropolis-Hastings algorithm. We found the AM algorithm to be faster and to perform better in terms of acceptance rate.

Again for $10,000$ iterations of the MCMC and for a dimension $d = 2$, but this time with no adaptation:

3 A Bit About knitr

Whilst this demonstrates that the algorithms both approximately sample from the correct target distribution, there is no improvement in acceptance rate due to the adaptation steps. We require a less regular shaped target distribution to test whether adaptation improves our acceptance rate towards the optimal 0.234. For this we follow Haario et al in using a banana-shaped distribution. The following results show a significantly lower acceptance rate for the adaptive version of the algorithm.

The AM algorithm with banana target:

The MH algorithm with banana target:

We can see that the acceptance probability is closer to the optimal 0.234 with adaptation according to Haario et al.

We now implement an adaptation scheme that uses stochastic stabilisation rather than numerical. This algorithm (AM2), from Roberts and Rosenthal (2009), differs from AM by using a mixture of Gaussians as the proposal distribution. In proportion β a normal uncorrelated distribution is used, and this is mixed with a correlated normal distribution.

$$Q_n(x, .) = (1 - \beta)N(x, s_d\Sigma_n) + \beta(N(x, (0.1^2)I_d/d)$$

The results for our implementation of AM2 are as follows. We have used a higher dimensional normal distribution for the proposal 20 and the banana target 3. Notice that the point at which we start adapting is determined by d, the dimension of the target distribution. The algorithm performs well when this d is high; we believe this is due to the point at which the adaptation starts. We have thus approximated a 3-dimensional banana with a 20-dimensional normal gaussian distribution. This forces the adaptation to start later. We would like to further investigate this effect, and explore various dimensional spaces.

4 Less naive covariance estimation

Naive empirical covariance estimators are unstable for high-dimensional problems with little data; the literature .

Following these two adaptive MH implementations, we are now looking to use something that is less naive than the vanilla estimator for the covariance matrix. In particular we have began testing the shrinkage and thresholding estimators as modifications to AM2. We will now test whether these estimators translate into better convergence by looking at their trajectories. We then plan to include burn in, and compare all algorithms in terms of the suboptimality factor following Roberts and Rosenthal (2009).

5 LaTeX

LaTeXitself is complicated if you've never used it before, but I'm sure you'll pick it up quickly: there are a lot of guides on the web. I recommend using the `align` environment (in the `amsmath` package) for displayed equations:

$$
\begin{aligned}
f(x) &= x^3 - x - 1 \\
g(y) &= y^4 + 2y
\end{aligned}
$$

You can cite in two ways using the `natbib` package: (Haario et al., 2001) and Haario et al. (2001).

References

A Gelman, GO Roberts, and WR Gilks. Efficient metropolis jumping rules. 1996.

Heikki Haario, Eero Saksman, and Johanna Tamminen. An adaptive metropolis algorithm. *Bernoulli*, 7(2):223–242, 04 2001. URL `http://projecteuclid.org/euclid.bj/1080222083`.

Gareth O. Roberts and Jeffrey S. Rosenthal. Examples of adaptive mcmc. *Journal of Computational and Graphical Statistics*, 18(2):349367, 2009. doi: 10.1198/jcgs.2009.06134. URL `http://dx.doi.org/10.1198/jcgs.2009.06134`.