Monte Carlo, Markov Chains, MCMC

Urša Zrimšek

23/06/2021

Monte Carlo integration

In this task we will compare different methods for approximation of integration. The example, on which we will apply different techniques is the following:

Let $X \sim N(0,1)$ and $f(x) = 10\exp(-5(x-3)^4)$. We are interested in computing the expectation of f with respect to X. On the plot below, we can see the function that we need to integrate to get the expectation, $f_{int}(x) = f(x)\frac{1}{\sqrt{2\pi}}e^{-\frac{1}{2}x^2}$, as a black line.

To get the value we will use as the true value throughout this report, we will approximate the integral using a quadrature based method.

```
## True value of the integral:  0.08939924
```

Monte Carlo method

First we will approximate the integral using Monte Carlo method:

```
## Monte Carlo approximation with 100 samples:  0.1148094
```

```
## Variance:  0.006557007
```

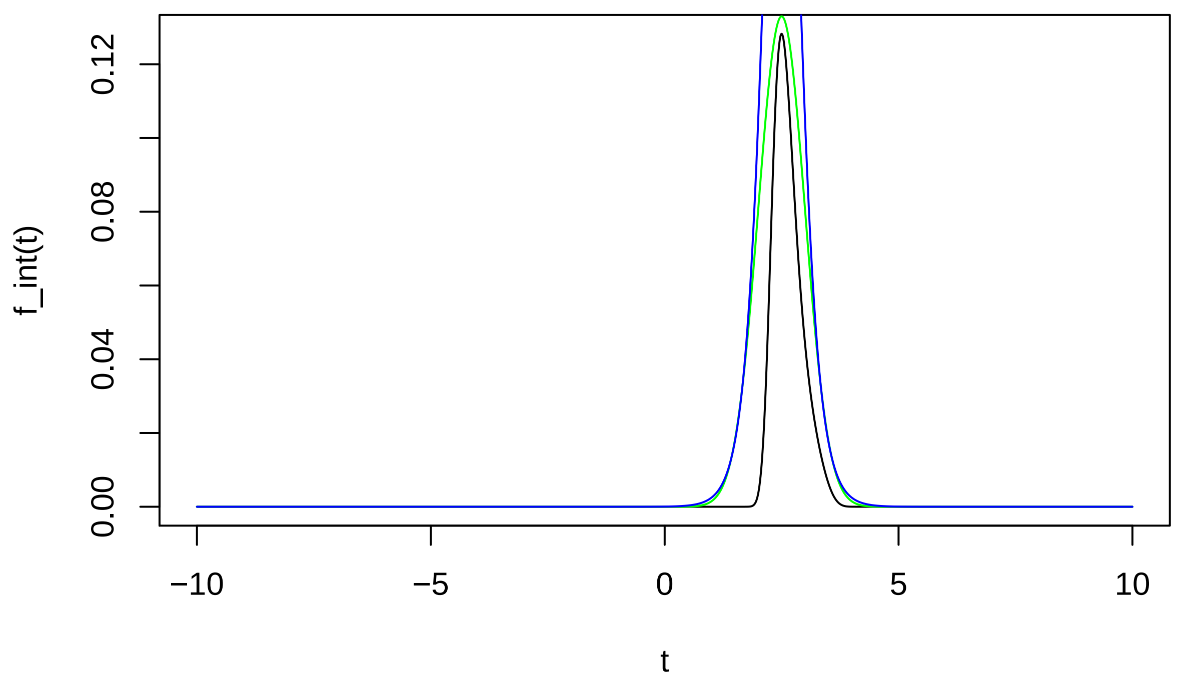
To evaluate uncertainty we will compute the approximation 1000 times, each time computing a 90% confidence interval using the normal approximation, and observe whether the CI contains the 'true' value.

```
## Percentage of repetitions where CI contains true value of the integral:  0.689
```

Since the percentage of confidence intervals that contain the true value is much smaller that 90%, we know that we underestimated the uncertainty of our evaluation.

Importance sampling

Now we will approximate the integral using importance sampling (100 samples).



We chose the surrogate distribution based on the figure above. Our choice is the normal distribution with $\mu = 2.5$ and $\sigma = 0.5$. We can see its pdf on the plot, colored green. We know how to efficiently sample from it, and it has similar shape and thicker tails than our original function.

```
## Importance sampling approximation with 100 samples:  0.09535043
```

```
## Variance:  2.96676e-05
```

We can see that we substantially lowered the error and uncertainty of our estimation with this approach, compared to Monte Carlo method. That happens because with sampling from g, the values that are sampled most often are those where f_{int} is the largest, and those have the most impact on our estimation.

Rejection sampling

Now we will, instead of sampling from build-in generator for the standard normal distribution, implement rejection sampling using a logistic distribution-based envelope. For that we need to find such constant, that logistic distribution (with correct parameters) multiplied with it will be bigger than the function we're integrating, everywhere. On the plot above we can see the comparison of our function (black) and logistic pdf with mean 2.5 and scale 0.25, multiplied by 0.25 (blue). We can see that this coefficient is enough for the envelope to be bigger everywhere. To be sure, we also checked it numerically. The results would be similar if we take bigger coefficient, but the sampling will be more efficient if we take smaller envelope.

Results of rejection sampling:

```
## Monte Carlo approximation with 100 samples from rejection sampling:  0.08740668
```

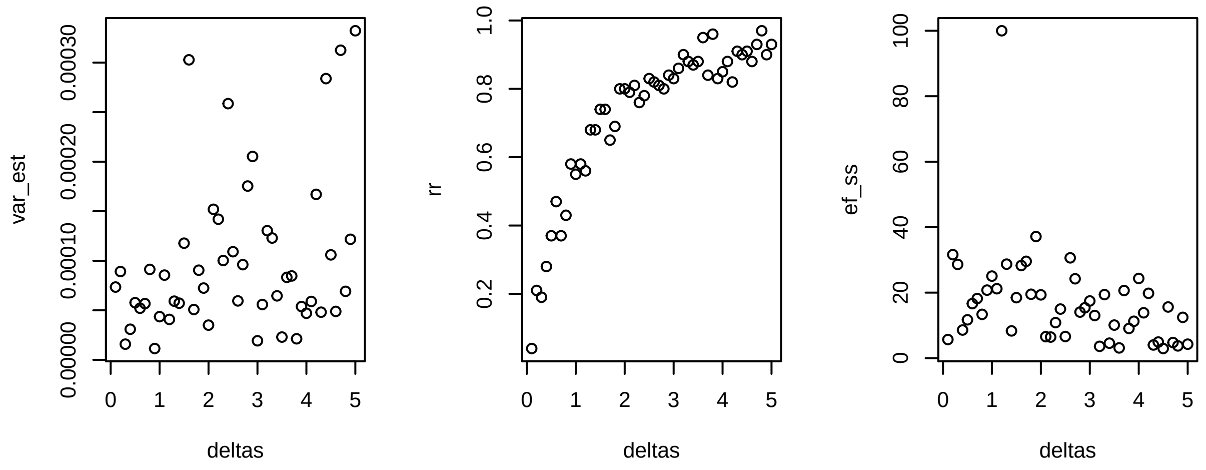
```
## Variance:  1.568755e-05
```

With rejection sampling the estimate is even closer to real value than with both above. Variance is similar than with importance sampling. It is much smaller than the one in Monte Carlo, because we are looking at the function below integral as a whole. With monte carlo we sample around 0, where the value of f is close to 0, and the value of our estimation relies on outliers. With rejection sampling we are sampling directly from an envelope that has high probability where our target function is also high, so we avoid this problem.

Metropolis-Hastings

Now we will again approximate the integral using Monte Carlo (100 samples), but instead of using the build-in generator for the standard normal distribution, we will implement Metropolis-Hastings with a $U(x_i - \delta, x_i + \delta), \delta > 0$ proposal distribution.

```
## Delta with biggest effective sample size:  1.2 ; estimation at that delta:  0.08981351
## ; variance at that delta:  4.081212e-05 ; ESS:  100
```



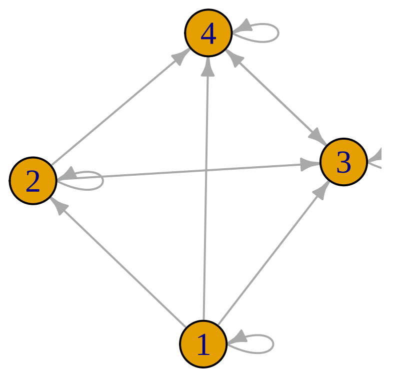
This approach again came closer to the real value as all above. The variance is in the same range as with rejection and importance sampling. We can see how the rejection rates, variance and ESS change with δ on the plots above.

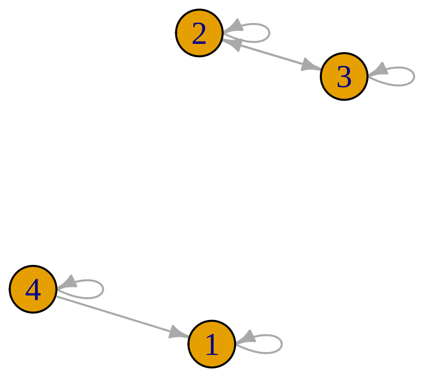
Markov Chains

In this section we will analyze some Markov Chains.

We will start with two chains whose graph representation can be seen below. The first one has three communicating classes, $\{1\}$, $\{2\}$ and $\{3, 4\}$, therefore the chain is not irreducible. They all have a period of 1, as we can directly return to each state from itself. From 1 and 2 we will sooner or later leave and we can't return, so they are transient. We can easily see that we can return to 3 and 4 and that expected positive return time is finite, and since positive recurrence is a class property, we know that it holds for states 3 and 4. For stationary distribution, we know that first two components need to be 0, because we can't return to them. If we calculate $\pi K = \pi$, where $\pi = (0, 0, a, b)$, we get a stationary distribution $\pi = (0, 0, \frac{1}{8}, \frac{7}{8})$.

The second one also has three communicating classes, $\{1\}$, $\{4\}$ and $\{2, 3\}$, therefore the chain is not irreducible. They all have a period of 1. When we leave, we can't return to 4, so it is transient, and others are positive reccurent - can be seen from the graph. We can quickly see that $(1, 0, 0, 0)$ and $(0, \frac{1}{2}, \frac{1}{2}, 0)$ are stationary distribution, and from them we can combine $(a, \frac{b}{2}, \frac{b}{2}, 0)$, where $a + b = 1$.





Other two chains we will inspect have infinite state space:

- $S = \mathbb{Z}$. If we are in state z, we have 0.5 probability to move to $z - 1$ and 0.5 probability to move to $z + 1$.
- $S = \mathbb{Z}$. If we are in state $z > 0$, we have 0.8 probability to move to $z - 1$ and 0.2 probability to move to $z + 1$. If we are in state $z < 0$, we have 0.2 probability to move to $z - 1$ and 0.8 probability to move to $z + 1$. In $z = 0$ we have equal probability to go to 1 or -1.

For both we can see that we have one communicating class, therefore they are irreducible. They both have period of 2, because we can return to any state in even number of steps.

We will show that the first chain is recurrent by using random variable X, that tells us where we will move from current state. It has two possible values, -1 and 1. The walk from current point can be described as the sum of X_i. If we calculate the expectation of the sum, $E(sum(X_i)) = sum(E(X_i)) = nE(X_i) = n(0.5 \cdot (-1) + 0.5 \cdot 1) = 0$, we see that we expect to return to this point, so the chain is recurrent. But we can't find a stationary distribution, because we would need $\pi(i)K = 0.5\pi(i-1) + 0.5\pi(i+1) = \pi(i)$ to hold. If we sum all $\pi(i)$ under this conditions, they wouldn't sum to 1. Therefore the chain is null-recurent.

For second chain we will show that it is positive-recurrent, by finding a stationary distribution (this holds because it is irreducible). We will find a distribution that satisfies detailed balance, as it is then stationary. All states that are more than 1 apart have $K(x, y) = 0$. So we only need to look at $\pi(i)K(i, i+1) = \pi(i+1)K(i+1, i)$. From this we can derive that $\pi(1) = \pi(-1) = \frac{5}{8}\pi(0)$. For $i > 0$: $\pi(i+1) = \frac{1}{4}\pi(i)$, and symmetrically for negative values. If we set $\pi(0) = c$ and sum all probabilities, we get $c = \frac{3}{8}$.

Markov Chain Monte Carlo

We will implement a Metropolis-Hastings sampler for Bayesian logistic regression.

First we will generate a toy dataset where we have 5 predictors, that are standard normal, and binary dependent variable, $Y_i \sim Bernoulli(\frac{1}{1+e^{-\mu_i}})$, where $\mu_i = 3x_1 - 2x_2 + \frac{1}{10}x_3 + 1$.

We will implement the logistic regression likelihood with an arbitrary number of coefficients and combine it with a weakly-informative standard normal prior $\beta_i \sim N(0, 100)$ to get a function that is proportional to the posterior distribution of the logistic regression model.

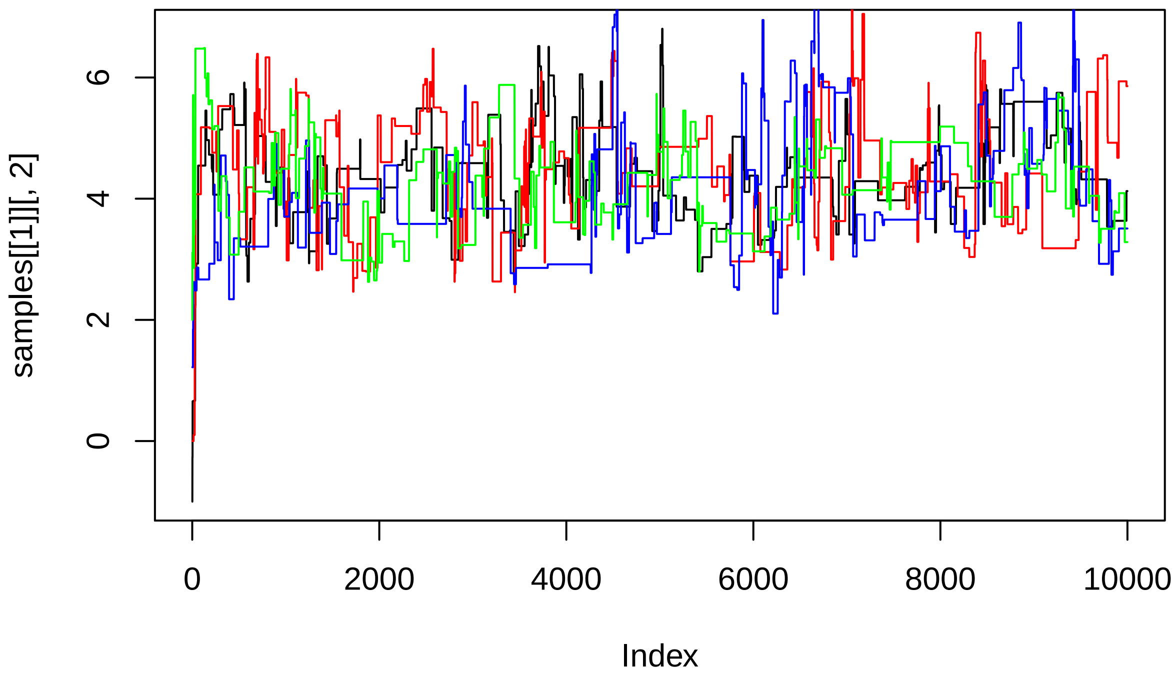
Now we will use Metropolis-Hastings with a standard normal proposal to infer the parameters on the toy dataset.

```
## Mean ESS over chains for all parameters:  70.5733 38.26006 49.38581 78.7878 107.8144 73.97963
```

```
## Rejection rates:  0.9792 0.9748 0.9783 0.98
```

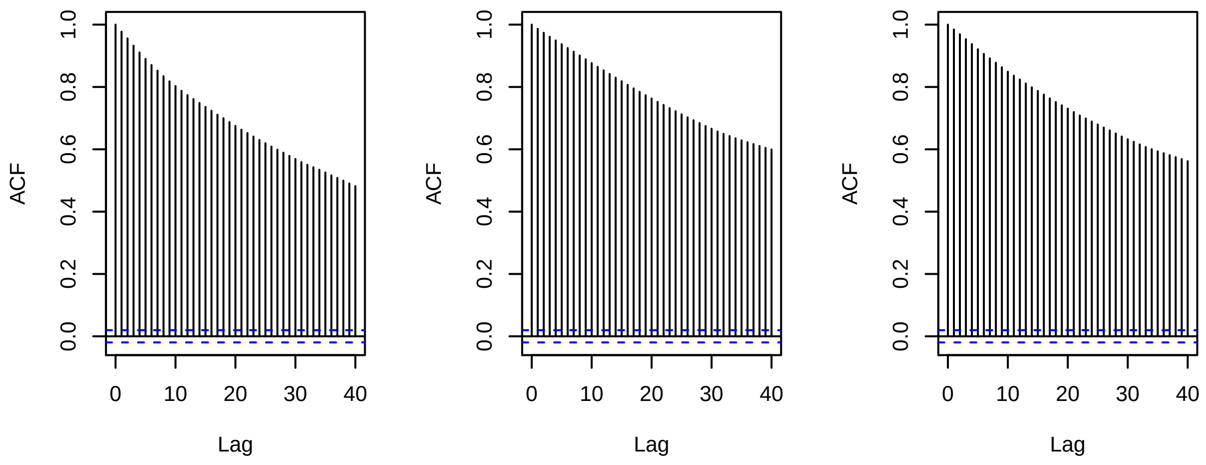
If we inspect the chains, the mean values for coefficients are roughly the same (and similar to true values, but let's pretend we don't know that). But the thing that should worry us is the effective sample size, as we can see that with 10000 we have the same amount of information as in less than 100 independent samples. The reason for this is in the rejection rate, as we keep less than 3 of the samples.

This can be also seen on the trace plots. Let's look at the plot for coefficient β_1, as others look similar.



We can see that it moves around 4, similarly for all chains (each started at different value, for them to be truly independent), which confirms that with a chain that long, starting point is not influential. But this trace plot shows that the values stay constant for a long time, before we find a new value to accept.

Next we will look at lag-k covariances. They are again similar for all β_i, we will illustrate this with showing first three. For all of them covariances fall with k, but stay high, still above 0.5 for $k = 40$. This also shows that we have highly autocorrelated chain.

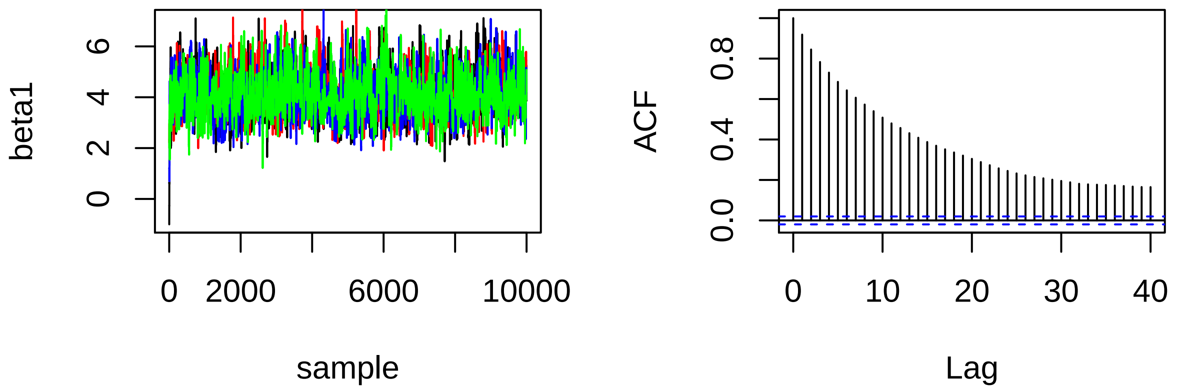


We tried to improve the efficiency of the sampler by changing the covariance matrix of the proposal distribution. Because the rejection rate is so high, we know that the values we are sampling are too far away from the correct values, and we need a long time to get good value. From the mean values of betas above, we concluded that β_1 should have the biggest variance, and last three should have the smallest, as their absolute values coincide with that. We tweaked the values a bit and came to the best ESS with variances for all coefficients (first one for intercept) equal to: $0.5, 1, 0.7, 0.3, 0.3, 0.3$. Below you can see mean ESS over all chains and rejection rates for all 4 chains.

```
## Mean ESS over chains:  347.6126 261.7259 320.2123 318.4949 144.2819 265.7411
```

```
## Rejection rates for all chains:  0.767 0.777 0.7678 0.7665
```

With different variances we came from less than 3% to almost 25% of accepted samples. Mean of ESS over all coefficients is now over 250, instead fo 70 as before. Below we will show a trace plot and autocorrelation plot for β_1, to see the difference from before. The trace plot doesn't show high reject rate as before, and lag-k covariances fall much faster.



Now we can estimate the posterior means β_i. With MLE fit of logistic regression we get: $\beta_0 = 1.01 \pm 0.38, \beta_1 =$

$3.62 \pm 0.76, \beta_2 = -1.83 \pm 0.47, \beta_3 = 0.05 \pm 0.30, \beta_4 = 0.32 \pm 0.37$ and $\beta_5 = -0.15 \pm 0.35$.

From Bayesian approach we get:

```
## Mean values for betas:  1.160274 4.134233 -2.110266 0.04603866 0.3789555 -0.1509104
```

```
## With SD of posterior:  0.4301668 0.8798644 0.4961595 0.3287447 0.3857325 0.3859714
```

We can see that the results are comparable, and both vary a bit from true values. Bayesian approach doesn't have true value of β_1 in the range of one standard deviation from mean value, but that could be also a consequence of only 100 datapoints. We can see that also with MLE approach the estimated value is higher than true value.

To end this task we will estimate the posterior probability $P(|\beta_3| > \frac{1}{10}|\text{data})$. We can do that easily by looking only at the percentage of values from posterior where this holds, and we get the value 0.77.

MH sampler for the Banana function

For the next task we will tune the MH sampler for the Banana function. We will take three different proposal distributions:

- uniformly from a square with side of length 1 and centered on the current state,
- uniformly from a square with side of length 20 and centered on the current state,
- third one, designed to outperform first two: normal with covariance matrix based on distance from mode.

For first proposal distribution, ESS and rejection rates are:

```
## Mean ESS over chains for coordinates:  2.19118 6.282969
```

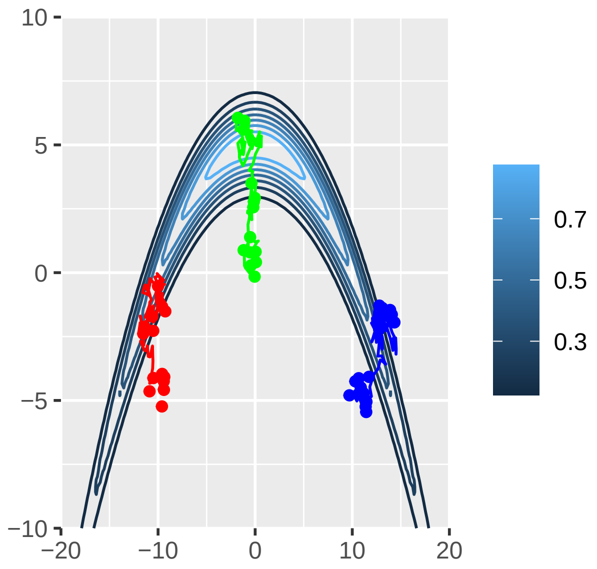
```
## Rejection rates for all chains:  0.185 0.163 0.092
```

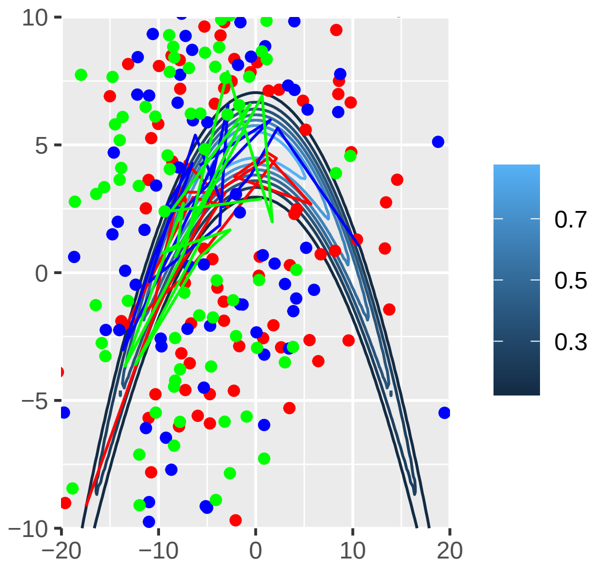
And for the second:

```
## Mean ESS over chains for coordinates:  17.76796 22.22702
```

```
## Rejection rates for all chains:  0.891 0.863 0.853
```

To better understand what's going on we will look at the first 100 points for both proposal distributions. On the left plot we see that all samples stay close to each other, but also close to the higher values of banana function, which leads us to low rejection rate and high autocorrelation. On the other hand, on the right side, where we have wider proposal distribution, we see that the accepted samples are further away, but we have a problem that most of the proposed points fall far away from the target distribution and they are rejected. This again leads us to high autocorrelation, as the samples in our chain are repeated.





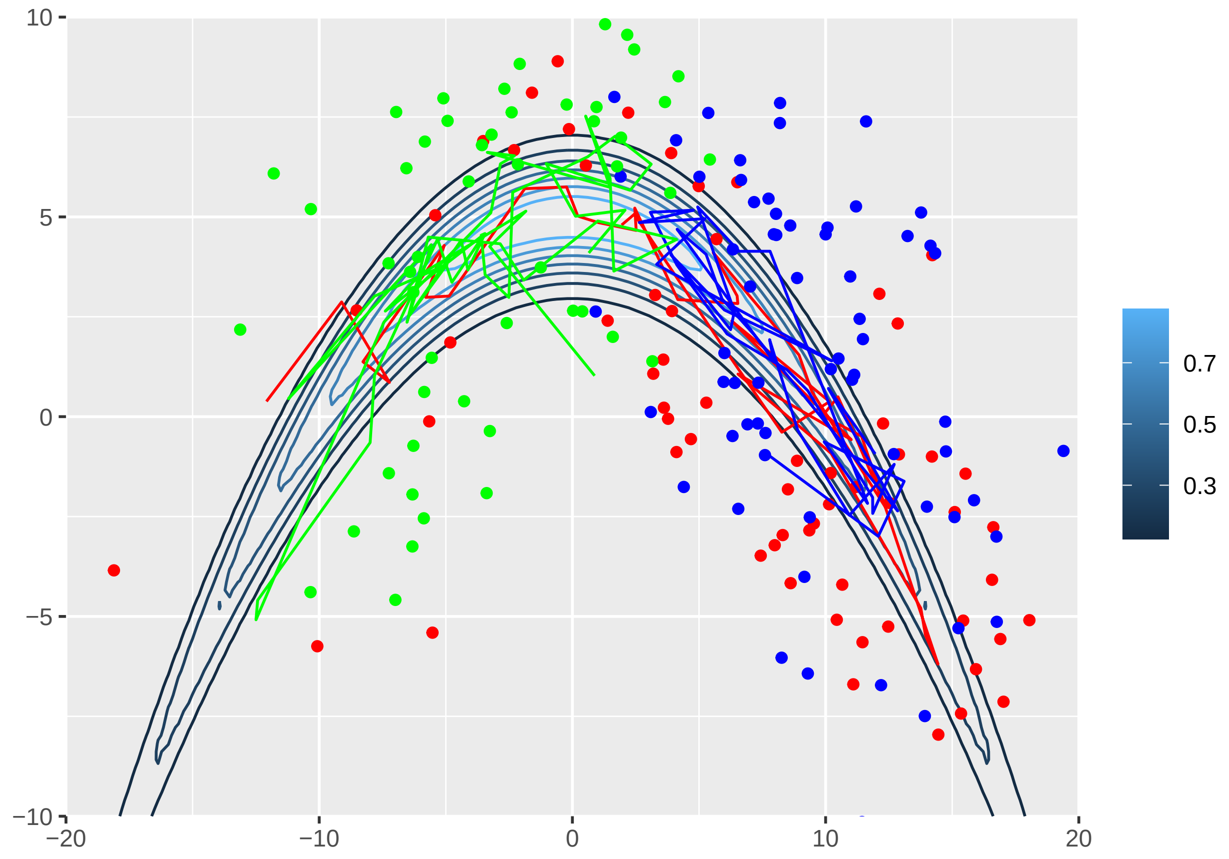
It is hard to find an efficient proposal distribution for this function because we would want it to be different for different positions. We will try to improve the proposal distribution by sampling from normal distribution, with covariance matrix dependent on the distance from maximum, obtained like this:

```
banana_cov <- function (x) {
  d <- max(5, sqrt(sum((x - c(0, 5))^2)))
  if (x[1] == 0) {cv <- 0} else if (x[1] > 0) {cv <- -d/3} else {cv <- d/3}
  matrix(c(d, cv, cv, d), 2, 2)
}
```

```
## Mean ESS over chains for coordinates:  892.4285 65.86944
```

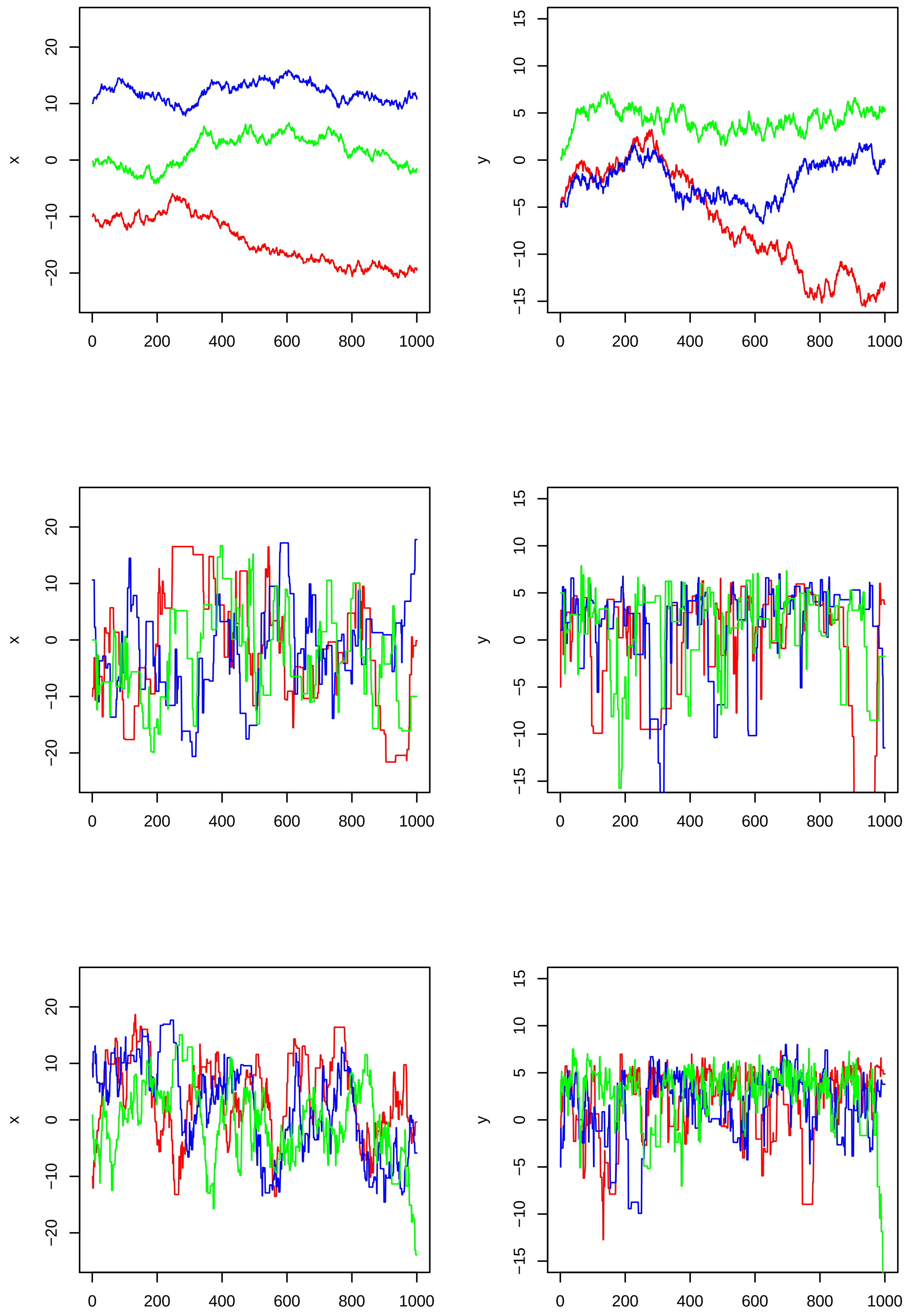
```
## ESS for separate chains:  2649.045 7.771154 20.46943 33.85796 31.29658 132.4538
```

```
## Rejection rates for all chains:  0.646 0.65 0.555
```



Here we see very high value for mean ESS for x, that's why we checked them for separate samples. We can see that in one chain, we managed to get negative autocorrelation, which means that our sampler tends to go where it wasn't before, and that's how it acts even better than with independent samples. Still, even if we wouldn't use this extreme case, the values for other two chains are also better than with other two proposal distributions.

For the end, let's look at the trace plots. The rows correspond to all three proposal distributions, and we can see that in the first one there are too small movements, the second one has big movements, but not often, and the last one is improved as it has bigger movements than the first one and doesn't stay on the same point as long as the middle one.


x
y
20
10
0
−10
−20
15
5
−5
−15
0
200
400
600
800
1000