Report

Margarita, Orlova, Kenneth Ruiter, Patrick Walker

November 3, 2018

Metropolis-Hastings

In this part of the report, we will use the Metro-polis Hastings algorithm to draw samples from a Beta distribution ϕ, with parameters $(6, 4)$. Of course, this is usually not necessary as most languages are already able to draw from a Beta distribution, but this way we are able to compare our results easily with the actual distribution. However, before we start, we will describe the algorithm itself.

Algorihm Description

The Metropolis-Hastings algorithm is a widely used statistical algorithm to generate samples from a distribution with a given probability density function (pdf). The algorithm uses a proposal distribution, that we are able to draw from, to obtain candidates for the actual sample draws. These candidates are then either accepted or rejected, according to a certain acceptance ratio. This ratio is calculated as follows:

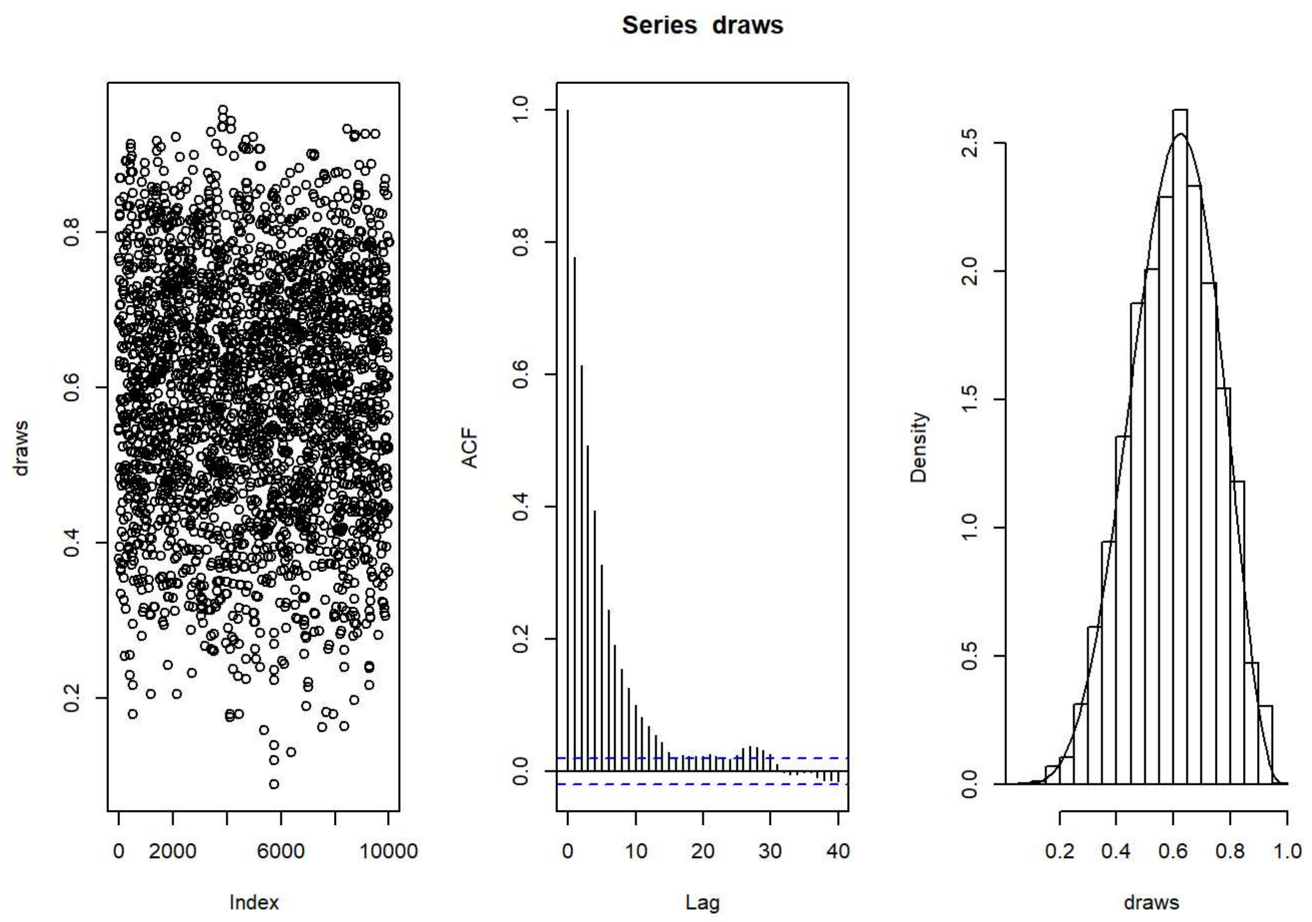
$$r = \frac{p(\phi_{prop}|y)/J_{+}(\phi_{prop}|\phi_{old})}{p(\phi_{old}|y)/J_{+}(\phi_{old}|\phi_{prop})}$$

The value of r can actually be bigger than 1, but that just means the candidate will definitely be accepted. Here, $p(x|y)$ is the pdf of the distribution that we want to draw from, with the parameters y (possibly a vector). The function $J_{+}(x|y)$ is the pdf of the proposal (or jumping) distribution, given the parameters y. Finally, ϕ_{prop} is the candidate and ϕ_{old} is the value of ϕ used to find the candidate. For this algorithm we need a starting value, which we will randomly draw from a uniform distribution in between 0 and 1. We also need a proposal distribution, which in this case will be a Beta distribution with parameters $(c\phi_{old}, c(1-\phi_{old}))$, where c is a constant. The number of times a candidate is drawn is a formerly specified number of iterations. Both the constant c and the number of iterations are used as an input of the function. Every accepted value of ϕ is stored in a chain, which is the output of the algorithm. The code for the function, including comments is shown below.

```
Metropolis <- function(c,iteration){ #Setting constant c and number of iterations

  if(c <= 0){stop("Input c must be greater than 0")} #prevents nonsensical user inputs

  if(iteration <= 0){stop("Input iterations must be greater than 0")}
      #prevents nonsensical user inputs

  fi=runif(1,0,1) #starting value for "Phi"

  chain = rep(0,iterations + 1) #initialzing array

  chain[1] = fi #putting initial value into chain

  for (i in 1:iteration){ #for loop for each iteration of the chain

    proposal = rbeta(1,c*chain[i],c*(1-chain[i])) #selects a random draw from the beta
          #distribution using parameter from chain as the proposal value


    acceptance_ratio = (dbeta(proposal,6,4)/dbeta(proposal,c*chain[i],c*(1-chain[i])))/
      (dbeta(chain[i],6,4)/dbeta(chain[i],c*proposal,c*(1-proposal)))
       #the probability of acceptance, acceptance ratio has been corrected for
       #asymmetry of jumping function
    if (runif(1) < acceptance_ratio){
       #this combined if/else statment accepts the proposal if the acceptance ratio
       #is greater than a random draw from the uniform function.
      chain[i+1] = proposal}

    else{

      chain[i+1] = chain[i]

      }
  }
  return(chain) #done
}
```

Results

Now that the algorithm itself has been discussed, we will run the function with $c = 1$ and $10,000$ total iterations. We will not discard any initial samples, which is sometimes done to prevent bias introduced by the starting value. The following plots show the results.



The trace of the data is roughly what we expect, namely that the data is clustering around some central value. This is qualitatively what we expect; in essence this plot only tells us that there is nothing obviously wrong with the data. The auto correlation test shows us what we would expect; namely that initial values are highly correlated but rapidly drop off. This emphasizes the importance of the burn in since we didn’t experience low auto correlation until we were about a quarter of the way through the iterations. The histogram seems to match well with the intended beta function. There are some issues near the top but this may be simply due to the limitations of trying to map a continuous function onto a discrete one.

```
## Warning in ks.test(draws, "pbeta", 6, 4): ties should not be present for
## the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  draws
## D = 0.023443, p-value = 3.366e-05
## alternative hypothesis: two-sided
```

```
## Warning in ks.test(draws, "pbeta", 6, 4): ties should not be present for
## the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  draws
## D = 0.023079, p-value = 4.722e-05
## alternative hypothesis: two-sided
```

```
## Warning in ks.test(draws, "pbeta", 6, 4): ties should not be present for
## the Kolmogorov-Smirnov test
```

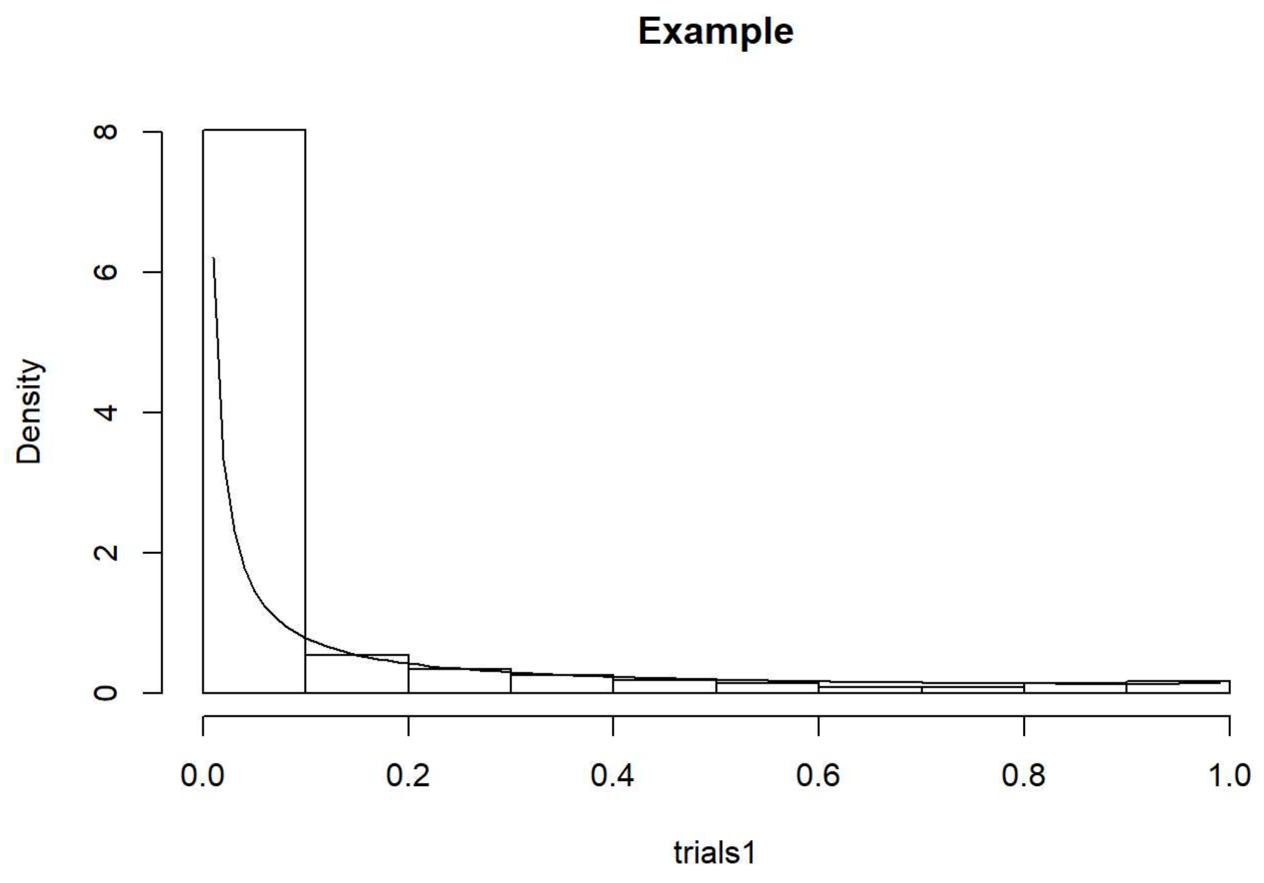
```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  draws
## D = 0.015642, p-value = 0.01498
## alternative hypothesis: two-sided
```

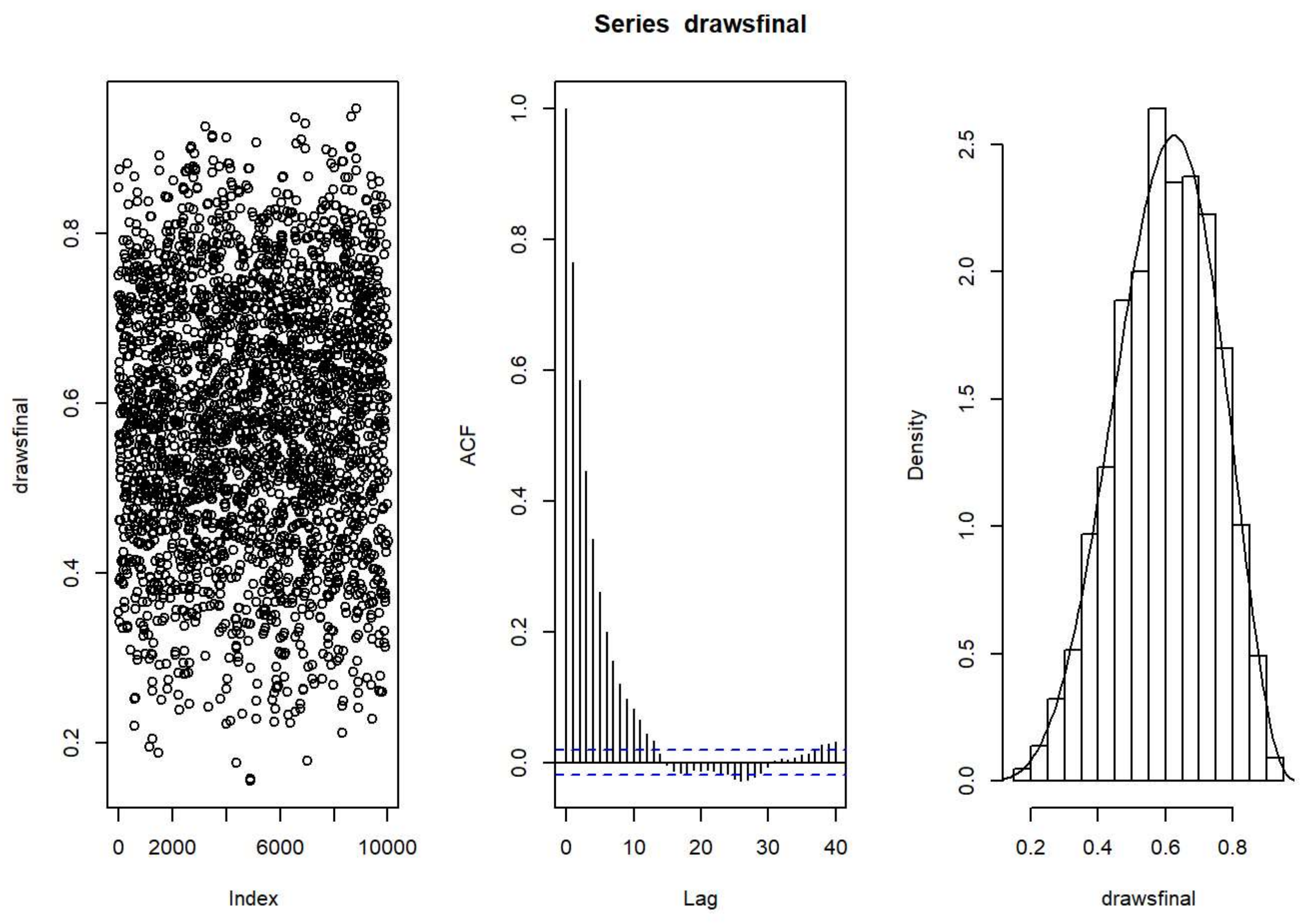
```
## Warning in ks.test(draws, "pbeta", 6, 4): ties should not be present for
## the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  draws
## D = 0.02009, p-value = 0.0006235
## alternative hypothesis: two-sided
```

```
## Warning in ks.test(draws, "pbeta", 6, 4): ties should not be present for
## the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  draws
## D = 0.02009, p-value = 0.0006235
## alternative hypothesis: two-sided
```


Example
Density
trials1
0
2
4
6
8
0.0
0.2
0.4
0.6
0.8
1.0


Series drawsfinal
drawsfinal
Index
0
2000
6000
10000
ACF
Lag
Density
drawsfinal

What we see is that for c = 1, the K-S statistics return p values so incredibly low we must reject the null hypothesis. This is because at a low value of c, our proposal function will return essentially a pareto distribution. Therefore, when we do get a sample near the center (with relatively low probability mind you) it is likely to get stored in the chain multiple times since the following draws for the proposal are very likely to come from the heavy tail, which are themselves likely to be rejected. This effect can be seen in the above Example histogram.

The trace plot may appear difficult to understand at first, but if we examine it in comparison to subsequent plots we can infer some things. If we have a high rate of acceptance from the proposal function, (such as in c = 10), then we would expect to see an almost uniform black band across the graph. This would indicate that we rarely had to have a point duplicated in the chain due to rejection. Here we see that we had medium rate of rejection, where the darker clusters indicate a sequence of duplicates.

The autocorrelation plot converges to 0 relatively quickly, but it is difficult to make more subtle estimations.

The histogram looks about comparable to the beta function plot, although we do tend to see a slightly higher peak, as discussed above. Again, unless the histogram is very bad, it is difficult to make inferences.

```
## Warning in ks.test(draws, "pbeta", 6, 4): ties should not be present for
## the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  draws
## D = 0.024449, p-value = 1.284e-05
## alternative hypothesis: two-sided
```

```
## Warning in ks.test(draws, "pbeta", 6, 4): ties should not be present for
## the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  draws
## D = 0.016311, p-value = 0.009773
## alternative hypothesis: two-sided
```

```
## Warning in ks.test(draws, "pbeta", 6, 4): ties should not be present for
## the Kolmogorov-Smirnov test
```

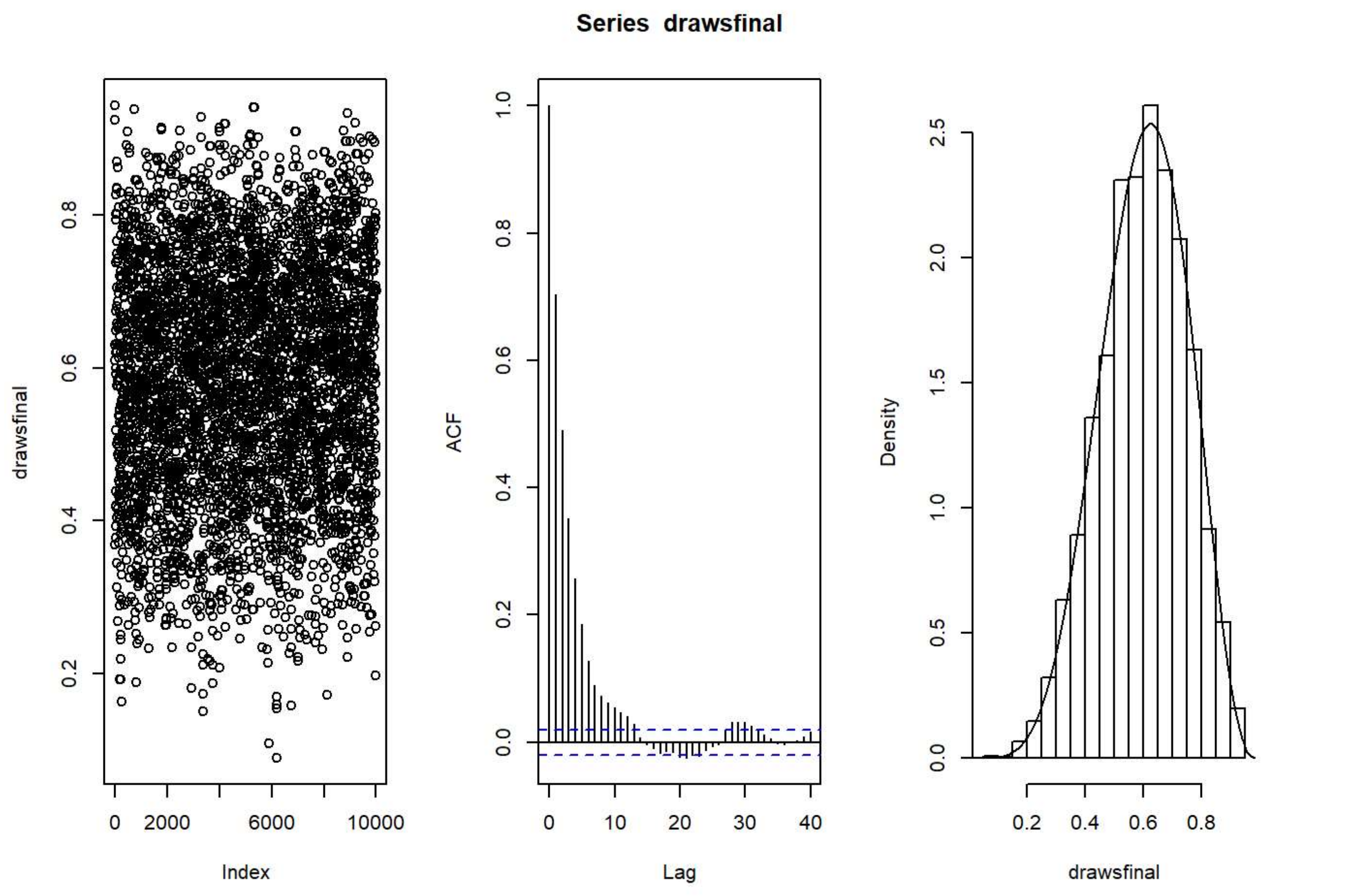
```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  draws
## D = 0.026793, p-value = 1.162e-06
## alternative hypothesis: two-sided
```

```
## Warning in ks.test(draws, "pbeta", 6, 4): ties should not be present for
## the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  draws
## D = 0.022771, p-value = 6.265e-05
## alternative hypothesis: two-sided
```

```
## Warning in ks.test(draws, "pbeta", 6, 4): ties should not be present for
## the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  draws
## D = 0.022771, p-value = 6.265e-05
## alternative hypothesis: two-sided
```



For c = 2.5, we no longer see p-values nearly as low as those we saw for c = 1. This is because the variance of our proposal function is now approximating that of the one we are sampling from. This means we are getting fewer extreme values which need to be tossed out, which in turn means fewer repeated values near the center.

For the trace plot, we see a higher rate of coverage and fewer areas of pure black spots (representing a sequence of the exact same value). This is to be expected since we should have a lower rate of rejection with c = 2.5.

Again, we have difficulty inferring anything from the autocorrelation plot beyond that it seems similar to c = 1.

The histogram doesn’t seem to have quite the same peak that the c = 1 histogram had.

```
## Warning in ks.test(draws, "pbeta", 6, 4): ties should not be present for
## the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  draws
## D = 0.01756, p-value = 0.004191
## alternative hypothesis: two-sided
```

```
## Warning in ks.test(draws, "pbeta", 6, 4): ties should not be present for
## the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  draws
## D = 0.02825, p-value = 2.335e-07
## alternative hypothesis: two-sided
```

```
## Warning in ks.test(draws, "pbeta", 6, 4): ties should not be present for
## the Kolmogorov-Smirnov test
```

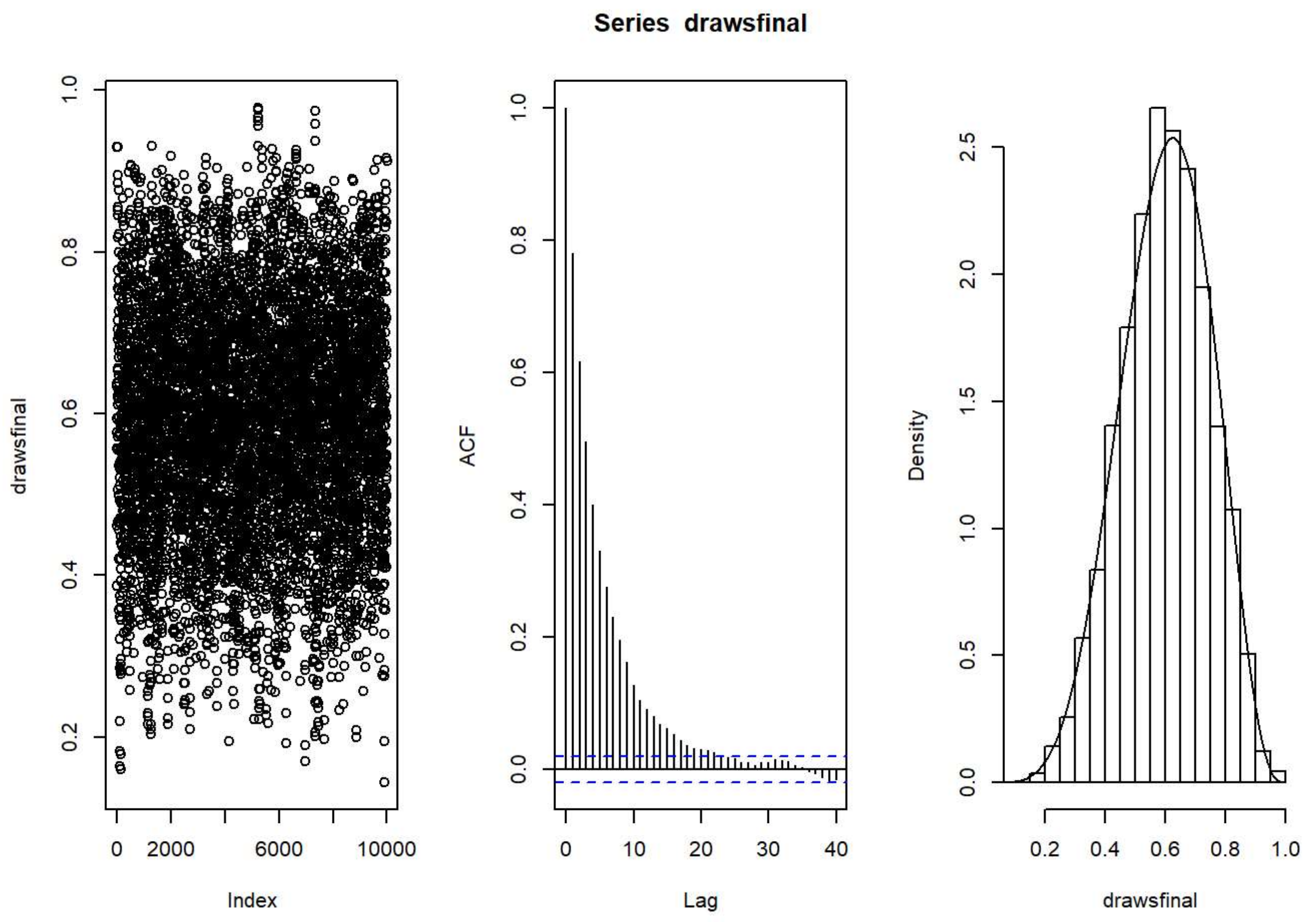
```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  draws
## D = 0.022415, p-value = 8.639e-05
## alternative hypothesis: two-sided
```

```
## Warning in ks.test(draws, "pbeta", 6, 4): ties should not be present for
## the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  draws
## D = 0.014349, p-value = 0.03254
## alternative hypothesis: two-sided
```

```
## Warning in ks.test(draws, "pbeta", 6, 4): ties should not be present for
## the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  draws
## D = 0.014349, p-value = 0.03254
## alternative hypothesis: two-sided
```



Now, with c = 10, we see another step closer to convergence of the variation. We still have some rejections, but even the smallest p-values aren't quite as small and we have a greater probability of getting p-values > 0.05. It should be noted however, that if we were to continue to much greater c's, such as c=100, we will have such a narrow variance that we aren't fully exploring the space. This means that we will have a very tightly clustered distribution, which will also return extremely small p-values.

The trace plot of c = 10, is the best we've seen so far. Namely, that we've seen good coverage of the area with an almost uniform black bar. We can also infer from this that there is relatively limited chains of the exact same value in the data, although the areas has become to dark to easily examine this. However, if we go significantly higher values of c, we would see a very narrow band of black, representing the limited variance and limited exploration of the sample space. This would also be unideal.

The autocorrelation plots look similar to those for c = 1 and c = 2.5.

```
## Warning in ks.test(draws, "pbeta", 6, 4): ties should not be present for
## the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  draws
## D = 0.051233, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
## Warning in ks.test(draws, "pbeta", 6, 4): ties should not be present for
## the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  draws
## D = 0.042287, p-value = 5.551e-16
## alternative hypothesis: two-sided
```

```
## Warning in ks.test(draws, "pbeta", 6, 4): ties should not be present for
## the Kolmogorov-Smirnov test
```

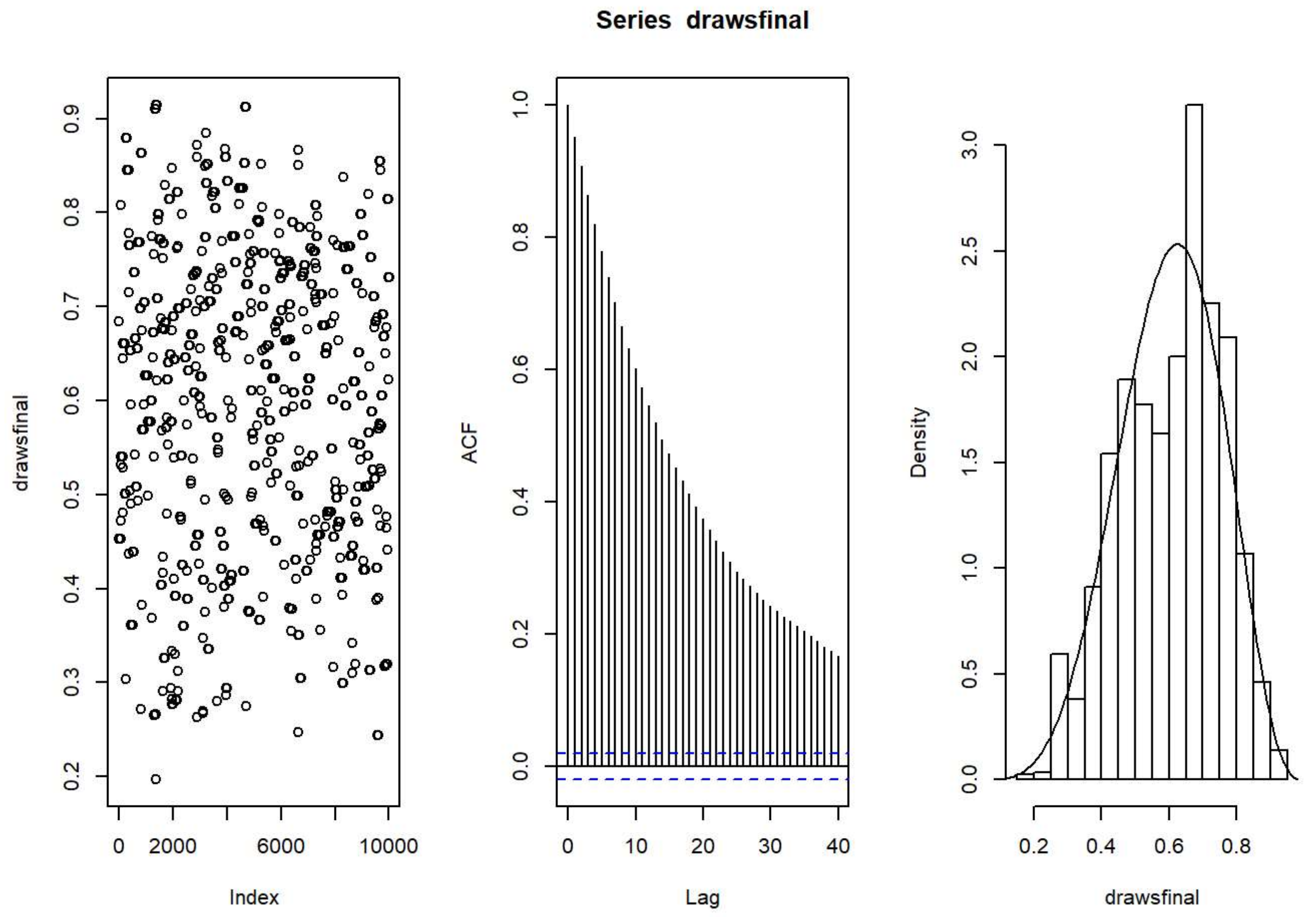
```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  draws
## D = 0.047998, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
## Warning in ks.test(draws, "pbeta", 6, 4): ties should not be present for
## the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  draws
## D = 0.056051, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
## Warning in ks.test(draws, "pbeta", 6, 4): ties should not be present for
## the Kolmogorov-Smirnov test
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  draws
## D = 0.056051, p-value < 2.2e-16
## alternative hypothesis: two-sided
```



With c = 0.1, we have a distribution so heavily skewed that the distribution is practically all tails. This leads to an incredibly high rejection rate, which is why we no longer see any difference between the p-values; we have maxed out how small the p-values can be in R.

In the trace plot we see a large amount of white space and a handful of dark spots. The white space indicates that we had a large number of rejections, which is reflected by the black points which indicate a sequence of values which were the exact same. This is expected, as was argued in the previous paragraph.

The autocorrelation plot never goes to 0 within the number of iterations. This is also not surprising since the loss of correlation is based on the iterative change in the values of phi. If phi only rarely changes, then it is more difficult to lose the correlation.

The histogram is clearly worse off than what we have seen for the previous c's, but still holds the general shape of the beta function. This suggests that there is a great discrepancy between estimating the Gibb's sampler using the K-S test and simply comparing the histogram to the distribution.

Given what we have seen from the impact of the c variable, c = 10. Essentially, the more appropriate the c value the fewer values will be needed to replicate the proposal distribution. The comparison to the target distribution is a more difficult problem, all the histograms we have seen before are relatively well centered on the beta distribution, meaning that if we were to extract estimates of the beta and alpha terms from this data we would have similar values with similar estimates of the variance. However, what we can tell from the trace plots and the histograms is that poor choices of c will lead to overweighting certain values. This would lead to a higher degree of bias in the estimate of the parameters.