Statistical Computing HW4

Wei-Chen Chang r12227118

Due: 2024-05-08

We want to sample from the double gamma distribution ($\alpha = 2$) with pdf

$$f(x|\alpha) = \frac{1}{2\Gamma(\alpha)}|x|^{\alpha-1}e^{-|x|};\ -\infty < x < \infty$$

1. Use the acceptance-rejection algorithms with the proposal distribution $Y \sim t(2)$. Plot the scaled histogram and matches it up with the theoretical PDF.
2. Use the Metropolis-Hastings algorithm with the proposal distribution $Y \sim N(X_t, 10)$. Plot the scaled histogram and matches it up with the theoretical PDF.
3. Compare the acceptance rate and performance of two algorithms.

Ans:

Setup

```
library(tidyverse)
library(glue)
Nsim <- 5000
doub_gamma <- function(x, alpha=2){
  return(1/(2*gamma(alpha)) *abs(x)^(alpha-1)*exp(-abs(x)))
}
```

Set number of iteration = 5000, and define the function `doub_gamma()` of double gamma pdf.
For visualization purpose, see Fig.1.

```
curve(doub_gamma, -10, 10, col="red", ylab= "density", ylim = c(0,.35),
      main = "Double Gamma and t Distribution")
curve(dt(x, df=2),-10, 10, lty=1, add =TRUE)
curve(3*dt(x, df=2),-10, 10, lty=2, add =TRUE, col = "grey20")
legend("topright",
       legend = c("doub_gamma",expression(t(df=2)),expression(3%*%t(df=2))),
       col = c("red", "black", "grey20"), lty = c(1,1,2), cex=.6,xjust = 0, bty="n")
```

1. A-R method

After trial and error, set c = 3 seems to be appropriate (see Fig.1). The result can be seen in Fig.2.

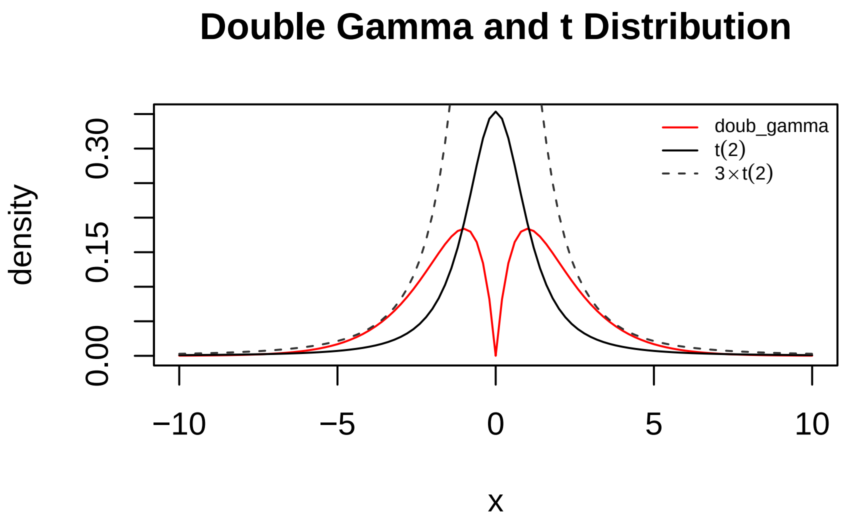

Figure 1: Target and Proposed Distribution in AR Method

```
AR_sim <- function(N = Nsim, c=3){
  env_samp <-  rt(N, df = 2) # envelop
  u <- runif(Nsim)
  prop <- doub_gamma(env_samp)/(c*dt(env_samp, df=2)) # acceptance crit
  samp <- env_samp[u<prop] #valid sample
  result <- list(sample = samp,
       accept_rate = length(samp)/Nsim)
  return(result)
}
plot_sim <- function(sample, title = "Sample",n=Nsim){
  sample |> hist(breaks = 100, freq=FALSE, xlab="x",
                  main = glue("Histogram of {title},(n={n})"))
  curve(doub_gamma, -10, 10, col="red", add= TRUE)
}
AR_samp <- AR_sim()
plot_sim(AR_samp$sample, "AR method")
```

2. Metropolis-Hastings algorithm

Accept Criterion:

$$\alpha = \min\{\frac{f(Y)q(Y|X_{i-1})}{f(X_{i-1})q(X_{i-1}|Y)}, 1\}$$

where $q(x|y)$ is the pdf of $N(y, \sigma = 10)$.

```
MH_sim <- function(N=Nsim){
  MH_samp <- 1:Nsim
  MH_samp[1] <- rnorm(1,sd=10) # x0
  u <- runif(Nsim)
```

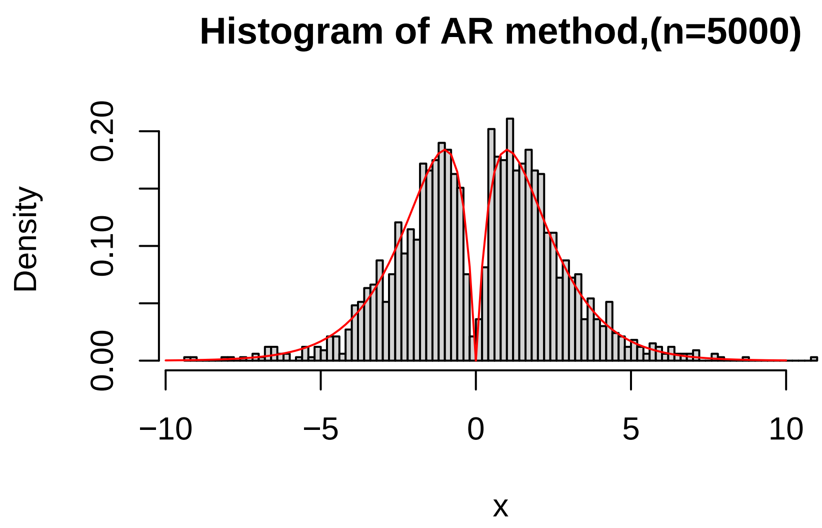

Figure 2: AR Sampler and Theoretical pdf

```
  accept <- 1 # accept first sample
  for (i in 2:Nsim){
    x <- MH_samp[i-1]
    y <- rnorm(1, mean = x, sd=10)
    num <- doub_gamma(y) * dnorm(y, mean=x, sd=10)
    den <- doub_gamma(x) * dnorm(x, mean=y, sd=10)
    if (u[i] < (num/den)){
      MH_samp[i] <- y
      accept <- accept+1
    }else MH_samp[i] <- x # reject
  }
  result <- list(sample = MH_samp, accept_rate=accept/Nsim)
  return(result)
}
MH_samp <- MH_sim()
```

First check its convergence through caterpillar plot. As the left panel of Fig.3 shows, the convergence seems to be OK. The histogram of the M-H sampler can be seen in the right panel of Fig.3.

```
#Check for Convergence
par(mfrow=c(1,2))
MH_samp$sample |> plot(type = 'l', main=glue("n={Nsim}, burnin=0"),
                       ylab=expression(x[i]), xlab="i")
plot_sim(MH_samp$sample, title = "M-H method")
```

One can see M-H performs fine when we set number of iteration (i) as 5000, even no burn-in sample were discard.

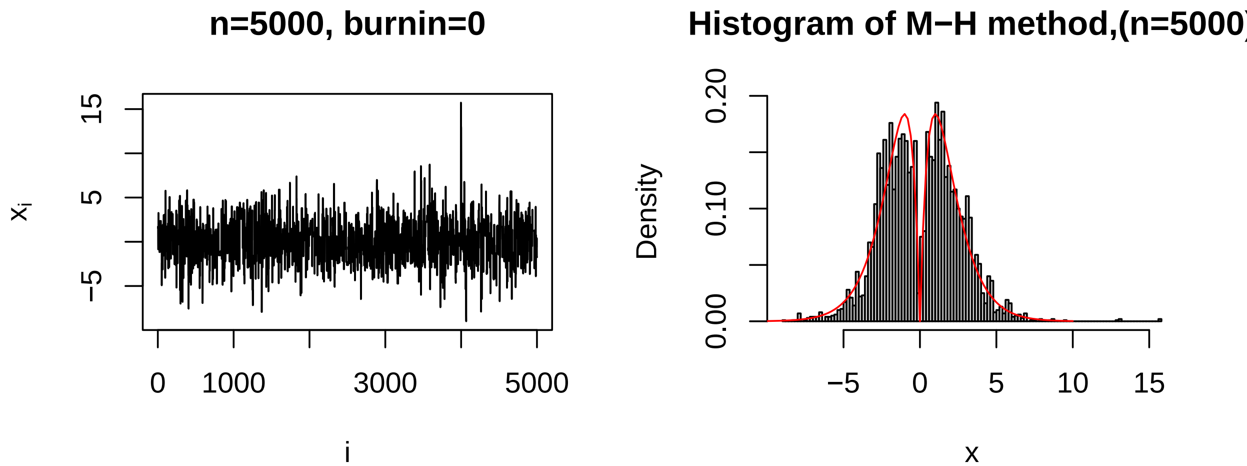

Figure 3: M-H sampler

3. Comparison b/t 2 mehtods

Acceptance rate in 2 methods are shown below. One can notice M-H method has lower acceptance rate compared to AR method.

```
cat(glue("Acceptance rate in AR method: {AR_samp$accept_rate*100}%\n
         Acceptance rate in M-H method: {MH_samp$accept_rate*100}%"))
```

Acceptance rate in AR method: 33.18%

Acceptance rate in M-H method: 27.1%

For performance evaluation, repeating the sampling procedure for 100 times.

```
cat(glue("Time taken in AR method (N = {Nsim}) for 100 times: \n
         "))
system.time(for (i in 1:100) AR_sim())
cat(glue("Time taken in M-H method (N = {Nsim}) for 100 times: \n
         "))
system.time(for (i in 1:100) MH_sim())
```

```
## Time taken in AR method (N = 5000) for 100 times:
##    user  system elapsed
##    0.25    0.00    0.25
## Time taken in M-H method (N = 5000) for 100 times:
##    user  system elapsed
##    2.43    0.10    2.52
```

One can see M-H is about 10 times slower than AR method in this case.