project

JD Bunker

11/25/2018

Algorithm:

Inputs: w ~ Unif(0,1)
$l_k(x^*) = log(g_l(x^*))$
$u_k(x^*) = log(g_u(x^*))$
$h(x^*) = log(g(x^*))$
$s_k(x) = exp(u_k(x))/\left(\int_D u_k(x')\,dx'\right) = g_u(x)/\left(\int_D g_u(x')\,dx'\right)$

The lower bound of h(x) is $l_k(x)$, which connects the values of function h on abscissaes. The function of $l_k(x)$ between two consecutive abscissaes x_j and x_{j+1} is $l_k(x) = \frac{(x_{j+1}-x)h(x_j)+(x-x_j)h(x_{j+1})}{x_{j+1}-x_j}$

Let T be the domain of abscissaes, H be the domain of the realized function H at abscissaes, H_prime be the domain of the realized first derivative of function H at abscissaes.

$h'(x) = \frac{dlog(g(x))}{dx} = \frac{g'(x)}{g(x)}$

Step 1: If $w < exp(l_k(x^*) - u_k(x^*))$
- Accept x^* when the condition is satisfied. Draw another x^* from $s_k(x)$
- Reject x^* when the condition is not satisfied.

Step 2: These two procedures can be done in parallel.
- Evaluate $h(x^*), h'(x^*)$. Update $l_k(x), u_k(x), s_k(x)$, which are now include x^* as an element. - Accept x^* if $w < exp(h(x^*) - u_k(x^*))$. Otherwise, reject.

Example: Start with g(x) = 3*N(0,1).
g(x) = $\frac{3}{\sqrt{2\pi}}e^{-(x)^2/2}$

```
#Create
library(Ryacas)
g <- function(x){
  sigma <- 1
  mu <- 0
  return(1/sqrt(2*pi*sigma^2)*exp(-(x-mu)^2/(2*sigma^2)))
}
g(0)
```

```
## [1] 0.3989423
```

```
g(1)
```

```
## [1] 0.2419707
```

```
g(-1)
```

```
## [1] 0.2419707
```

```
sigma <- 1
mu <- 0
h <- D(expression(1/sqrt(2*pi*sigma^2)*exp(-(x-mu)^2/(2*sigma^2))),'x')
print(h)
```

```
## -(1/sqrt(2 * pi * sigma^2) * (exp(-(x - mu)^2/(2 * sigma^2)) *
```

```
##     (2 * (x - mu)/(2 * sigma^2))))
```

```r
dx <- deriv(expression(1/sqrt(2*pi*sigma^2)*exp(-(x-mu)^2/(2*sigma^2))),Sym('x'))
sigma <- 1
mu <- 0
#install.packages("Ryacas")

test <- Simplify(dx);
print(test)
```

```r
test
```

```r
Initial <- function(k_start,x.Bound,h,h_d){
  X=c() #Initialize X
  H=
  if (x.Bound[0] %in% c(-Inf,Inf)){
    print("hey")
  }
}
```

```r
#Draw x* from s
#install.packages("MCMCpack")
library(MCMCpack)
samples = MCMCmetrop1R(fun=s, mcmc = 100, theta.init=1,V=as.matrix(1))
```

Start with h(x) = log(g(x))

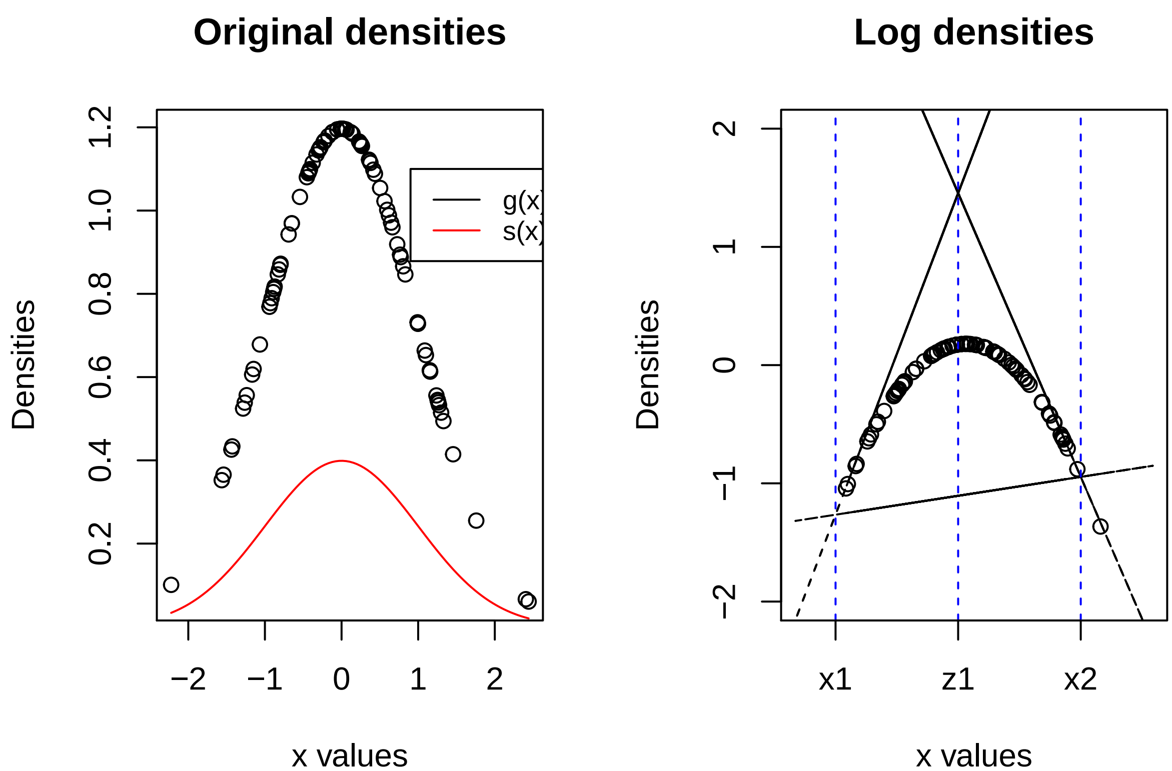
```r
#install.packages("Deriv")
library(Deriv)
```

```
##
## Attaching package: 'Deriv'

## The following object is masked from 'package:Ryacas':
##
##     Simplify
```

```r
g <- function(x) {
  (3/sqrt(2*pi))*exp(-(x^2)/2)
}
g_prime <- Deriv(g)
set.seed(0)
x <- rnorm(100)
g_x <- g(x)
s_x <- g(x)/integrate(g, lower = "-Inf", upper = "Inf")$value
h_x <- log(g_x)
#Work with the log densities
#Calculate the lower bounds:
#Initialize two abscissaes, X is a vector that includes abscissaes
X <- c(-1.7,1.5)
X <- sort(X)
Z <- c()
#H is a vector that includes density h at abscissaes
#H_prime is a vector that includes first derivative of density h at abscissaes
update_H <- function(x) {
 log(g(x))
}
```

```
update_H_prime <- function(x) {
g_prime(x)/g(x)
}
H <- sapply(X, update_H)
H_prime <- sapply(X, update_H_prime)
#Calculate initial lower bounds
l <- function(x,i) {
  ((X[i+1]-x)*H[i] + (x-X[i])*H[i+1])/(X[i+1]-X[i])
}
u <- function(x,i) {H[i] + (x-X[i])*H_prime[i]}
#Calculate initial intersection of two tangent lines at abscissaes
z <- function(i) {
(H[i+1]-H[i]-X[i+1]*H_prime[i+1]+X[i]*H_prime[i])/(H_prime[i]-H_prime[i+1])
}
z_order <- 1
Z[z_order] <- z(z_order)
par(mfrow=c(1,2))
plot(x,g_x, type = "p", col = "black", xlab = "x values", ylab = "Densities", main = "Original densitie
curve(g(x)/integrate(g, lower = -Inf, upper = Inf)$value,add = TRUE,col="red")
legend(0.9,1.1,legend=c("g(x)","s(x)"),lty=1:1,col=c("black","red"),cex=0.8)
plot(x,h_x, xlab = "x values", ylab = "Densities", main = "Log densities", ylim = c(-2,2),xaxt='n')
lines(x,l(x,1),lty=2)
lines(x,u(x,1),lty=2)
lines(x,u(x,2),lty=2)
abline(v=c(X[1],Z[1],X[2]), lty = 2, col="blue")
axis(1, c(X[1],Z[1],X[2]),c("x1","z1","x2"))
```

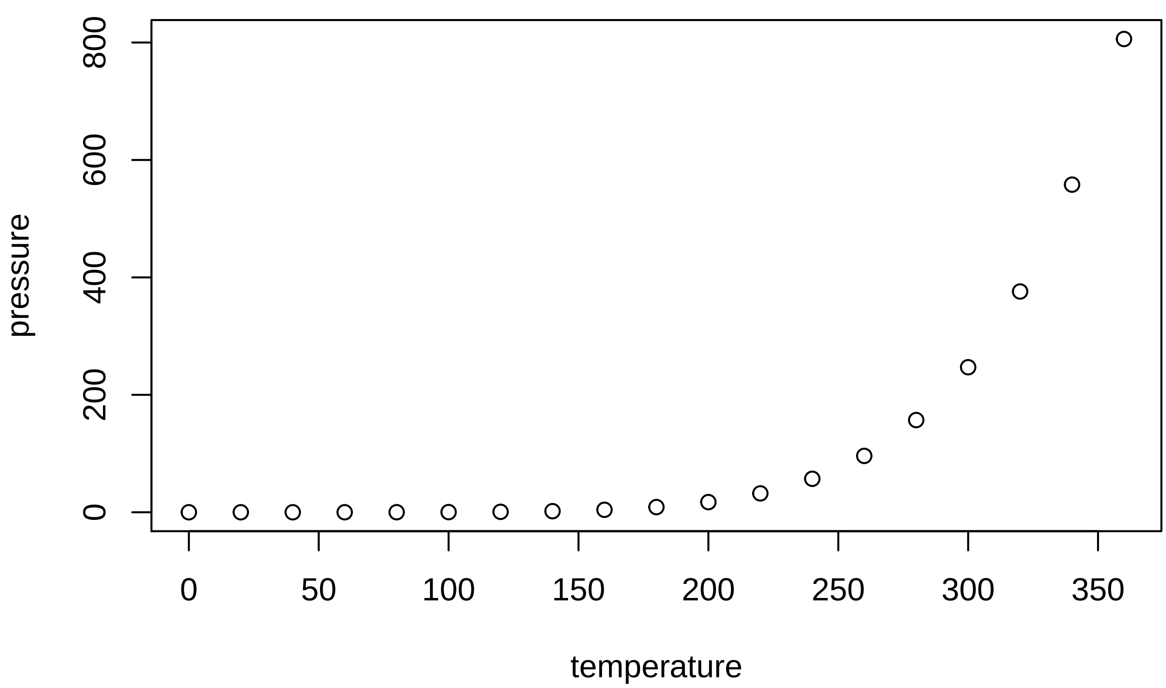



```
x_accept <- c()
length_accept = 1
```

```r
#Step 1: Choose if w < exp(l - u)
w <- runif(1)
x_star <- runif(1,min = X[1], max = X[2])
i_star_x <- which.max(X[X<x_star])
i_star_z <- which.min(Z[x_star<Z])
if(w < exp(l(x_star,i_star_x) - u(x_star,i_star_z))){
    x_accept[length_accept] <- x_star
    length_accept <- length_accept+1
    print(x_accept)
    print(w)
}else {
    if (w < exp(log(g(x_star)) - u(x_star,i_star_z))) {
    x_accept[length_accept] <- x_star
    length_accept <- length_accept+1
    print(X)
    print(H)
    }
  #Update step
    X <- c(X, x_star)
    X <- sort(X)
    H <- sapply(X, update_H)
    H_prime <- sapply(X, update_H_prime)
    for (z_order in 1:length(X)-1) {
      Z[z_order] <- z(z_order)
    }
}
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.