Initialize Step (Step 1/3)

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
#install.packages("Ryacas")
#install.packages("Deriv")
#install.packages("distr")
#install.packages("truncnorm")
library(truncnorm)
library(Ryacas)
library(Deriv)
```

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

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

```
#library(distr)


#####################################
#User supplied parameters

#User supplied g-function
g <- function(x){
```

```r
  sigma <- 1
  mu <- 0
  val <- 3/sqrt(2*pi*sigma^2)*exp(-(x-mu)^2/(2*sigma^2))
}

#Examples for function g
g(0)
g(1)
g(-1)

#Number of starting points
k_start=2

#Bounds for x
x.Bound=c(-Inf,Inf)


######################################
#Test default deriv() function + Simplify() function
sigma <- 1
mu <- 0

x <- Sym("x")
f <- 1/sqrt(2*pi*sigma^2)*exp(-(x-mu)^2/(2*sigma^2))
#print(f)
dx <- eval(deriv(g(x),Sym("x"))); #Derivative of f in terms of x
print(dx)
```

```
## expression(-1.1968268412 * (exp(-x^2/2) * (2 * x))/2)
```

```r
dxSimp <- Ryacas::Simplify(deriv(g(x),"x")) #Simplified version of derivative (collapse terms)...
print(dxSimp)
```

```
## expression(-2.3936536824 * (exp(-x^2/2) * x)/2)
```

```r
s = deriv(~a^2,'a',func = T) #Create function s(a)=a^2
#s(5) #5^2
```

```r
######################################
#Internally created functions
g_d <- function(x){
  as.numeric(eval(Deriv(~g(X),"X")))
}
#h function
h <- function(x){
    log(g(x))
}

#Examples for function h
#h(1)
#h(-1)
#h(0)

#h' function
h_d <- function(X){
```

```r
    #print("X:")
    #print(X)

    #print("Original:")
    #print(h(X))

    #print("Derivative - deriv()")
    #print(deriv(log(g(x)),"x"))

    #print("Derivative - Deriv() Unevaluated")
    #print(Deriv(~log(g(X)),"X"))

    #print("Derivative - Deriv() Evaluated")
    #print(eval(fun))
    #deriv(log(g(x),1))


    #return(as.numeric(eval(Deriv(~log(g(X)),"X"))))
  return(as.numeric(1/g(X)*eval(Deriv(~g(X),"X"))))
}

#Examples for function h'...
#h_d(0)
#h_d(-0.5)
```

```r
#Initial function (main)
Initial <- function(k_start,x.Bound,g,QC=FALSE){
  #print(all.equal(x.Bound,c(-Inf,Inf)))

    upBd <- x.Bound[length(x.Bound)]
    lowBd <- x.Bound[1]
    if (lowBd != -Inf){
      needPosHprime <- FALSE
    } else{
      needPosHprime <- TRUE
    }
    if (upBd != Inf){
      needNegHprime <- FALSE
    } else{
      needNegHprime <- TRUE
    }
    upBdInit <- upBd #Upper bound for initialization points
    lowBdInit <- lowBd #Lower bound for initialization points
    if (0==1){

      if (QC==TRUE){
        print("Lower & upper bound pairs")
        print(lowBd)
        print(upBd)

        print(upBdInit)
        print(lowBdInit)
      }
    }
```

```r
k <- 0
X <- c()


#print(paste0("Begin: Find initialization points"))
while (k < k_start){
  samp <- rtruncnorm(1,a=lowBdInit,b=upBdInit)
  hPrime <- h_d(samp)
  if (QC==TRUE){
    print(paste0("Current X vector: ",X))
    print(paste0("Initialization bounds: ",lowBdInit," ",upBdInit))
    print(paste0("Need positive / negative h'? ",needPosHprime," ",needNegHprime))

    print(paste0("Sampled point x: ",samp))
    print(paste0("h'(x): ",h_d(samp)))
  }

  if (hPrime >0 & needPosHprime==TRUE){
    needPosHprime <- FALSE
    X <- append(X,samp,0)
    k <- k+1
    #print(paste0("Appended X with h'(X) > 0"))

    if (needNegHprime==TRUE) {
      lowBdInit <- samp
      #print(paste0("Changed lower initialization bound to sampled point"))
    }
    else {
      lowBdInit <- lowBd
      #print(paste0("Changed lower initialization bound to distribution lower bound"))
    }
  }
  else if (hPrime < 0 & needNegHprime==TRUE) {
    needNegHprime <- FALSE
    X <- append(X,samp,length(X))
    k <- k+1
    #print(paste0("Appended X with h'(X) < 0"))

    if (needPosHprime == TRUE) {
      upBdInit <- samp
      #print(paste0("Changed upper initialization bound to sampled point"))
    }
    else {
      upBdInit <- upBd
      #print(paste0("Changed upper initialization bound to distribution upper bound"))
    }
  }
  else if (k < (k_start-needPosHprime-needNegHprime)){
    X <- append(X,samp,length(X[X<samp]))
    k <- k+1
    #print(paste0("Appended X; no h' requirements met"))
  }
```

```
      else {
        #print(paste0("Sampled point does not satisfy any initialization requirements. Sample again."))
      }
    }
    #print(paste0("Finished: Find initialization points"))


    H <- h(X)
    H_d <- h_d(X)
    Z <- c(lowBd,upBd)
  return(list(X,H,H_d,Z))
}
```

Normal distribution (2 starting pts)

```
  #User supplied g-function
g <- function(x){
  sigma <- 1
  mu <- 0
  val <- 3/sqrt(2*pi*sigma^2)*exp(-(x-mu)^2/(2*sigma^2))
}

Initial(k_start=2,x.Bound=c(-Inf,Inf),g
        #,QC=TRUE
        )
```

```
## [[1]]
## [1] -1.809737  1.025240
##
## [[2]]
## [1] -1.4579007 -0.3458851
##
## [[3]]
## [1]  1.809737 -1.025240
##
## [[4]]
## [1] -Inf  Inf
```

```
system.time(Initial(k_start=2,x.Bound=c(-Inf,Inf),g
        #,QC=TRUE
        ))
```

```
##    user  system elapsed
##   0.095   0.000   0.095
```

Exponential distribution (2 starting pts)

```
g <- function(x){
  lam <- 1
  value <- lam*exp(-lam*x)
```

```
}
#a <- AbscontDistribution(d=g,low=0,low1=0)
#print(g(1))
Initial(k_start=2,x.Bound=c(0,Inf),g
        #,QC=TRUE
        )
```

```
## [[1]]
## [1] 0.3173624 1.4007340
##
## [[2]]
## [1] -0.3173624 -1.4007340
##
## [[3]]
## [1] -1 -1
##
## [[4]]
## [1]    0 Inf
```

Normal distribution (5 starting pts)

```
  #User supplied g-function
g <- function(x){
  sigma <- 1
  mu <- 0
  val <- 3/sqrt(2*pi*sigma^2)*exp(-(x-mu)^2/(2*sigma^2))
}
Initial(k_start=5,x.Bound=c(-Inf,Inf),g
        ,QC=TRUE
        )
```

```
## [1] "Current X vector: "
## [1] "Initialization bounds: -Inf Inf"
## [1] "Need positive / negative h'? TRUE TRUE"
## [1] "Sampled point x: 0.827263581205055"
## [1] "h'(x): -0.827263581205055"
## [1] "Current X vector: 0.827263581205055"
## [1] "Initialization bounds: -Inf 0.827263581205055"
## [1] "Need positive / negative h'? TRUE FALSE"
## [1] "Sampled point x: -0.55359391686273"
## [1] "h'(x): 0.55359391686273"
## [1] "Current X vector: -0.55359391686273"
## [2] "Current X vector: 0.827263581205055"
## [1] "Initialization bounds: -Inf 0.827263581205055"
## [1] "Need positive / negative h'? FALSE FALSE"
## [1] "Sampled point x: -0.146730868658299"
## [1] "h'(x): 0.146730868658299"
## [1] "Current X vector: -0.55359391686273"
## [2] "Current X vector: -0.146730868658299"
## [3] "Current X vector: 0.827263581205055"
## [1] "Initialization bounds: -Inf 0.827263581205055"
## [1] "Need positive / negative h'? FALSE FALSE"
```

```
## [1] "Sampled point x: 0.093344679387136"
## [1] "h'(x): -0.093344679387136"
## [1] "Current X vector: -0.55359391686273"
## [2] "Current X vector: -0.146730868658299"
## [3] "Current X vector: 0.093344679387136"
## [4] "Current X vector: 0.827263581205055"
## [1] "Initialization bounds: -Inf 0.827263581205055"
## [1] "Need positive / negative h'? FALSE FALSE"
## [1] "Sampled point x: -0.390368273406002"
## [1] "h'(x): 0.390368273406002"

## [[1]]
## [1] -0.55359392 -0.39036827 -0.14673087  0.09334468  0.82726358
##
## [[2]]
## [1]  0.02644064  0.10348006  0.16890878  0.17531714 -0.16250876
##
## [[3]]
## [1]  0.55359392  0.39036827  0.14673087 -0.09334468 -0.82726358
##
## [[4]]
## [1] -Inf  Inf
```

```r
print(system.time(Initial(k_start=5,x.Bound=c(-Inf,Inf),g
       #,QC=TRUE
       )))
```

```
##    user  system elapsed
##   0.193   0.000   0.193
```

Gamma distribution (5 starting pts)

```r
#User supplied g-function
g <- function(x){
  alpha <- 1
  beta <- 1
  val <- beta^alpha/gamma(alpha)*x^(alpha-1)*exp(-beta*x)
}

Initial(k_start=5,x.Bound=c(0,Inf),g
       #,QC=TRUE
       )
```

```
## [[1]]
## [1] 0.5857307 0.8094329 0.8118139 1.1094600 1.2906118
##
## [[2]]
## [1] -0.5857307 -0.8094329 -0.8118139 -1.1094600 -1.2906118
##
## [[3]]
## [1] -1 -1 -1 -1 -1
##
## [[4]]
## [1]   0 Inf
```

Beta distribution (10 starting pts)

```r
  #User supplied g-function
g <- function(x){
  alpha <- 1
  beta <- 3
  val <- x^(alpha-1)*(1-x)^(beta-1)/(gamma(alpha)*gamma(beta)/gamma(alpha+beta))
}

Initial(k_start=10,x.Bound=c(0,1),g
        #,QC=TRUE
        )
```

```
## [[1]]
##  [1] 0.04835229 0.05971081 0.05986599 0.09174070 0.20189613 0.50596560
##  [7] 0.51769779 0.67877738 0.83442538 0.94961335
##
## [[2]]
##  [1]  0.9994916  0.9754767  0.9751466  0.9061616  0.6475792 -0.3116880
##  [7] -0.3597564 -1.1726294 -2.4980543 -4.8774456
##
## [[3]]
##  [1]  -2.101618  -2.127005  -2.127356  -2.202014  -2.505939  -4.048301
##  [7]  -4.146778  -6.226212 -12.079146 -39.693052
##
## [[4]]
## [1] 0 1
```

```r
system.time(Initial(k_start=10,x.Bound=c(0,1),g
        #,QC=TRUE
        ))
```

```
##    user  system elapsed
##   0.444   0.000   0.445
```