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("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,length(X))
    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,0)
    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]  2.4119015 -0.1000012
##
## [[2]]
## [1] -2.7289608  0.1746736
##
## [[3]]
## [1] -2.4119015  0.1000012
##
## [[4]]
## [1] -Inf  Inf
```

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

```
##    user  system elapsed
##   0.287   0.003   0.305
```

Exponential distribution (2 starting pts)

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

```r
#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] 0.8800005 1.1956156
##
## [[2]]
## [1] -0.8800005 -1.1956156
##
## [[3]]
## [1] -1 -1
##
## [[4]]
## [1]   0 Inf
```

Normal distribution (5 starting pts)

```r
  #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]]
## [1]  0.07065779 -1.46264688 -0.66467338 -0.56187684  0.33905853
##
## [[2]]
## [1]  0.17717749 -0.88999420 -0.04122160  0.02182096  0.12219341
##
## [[3]]
## [1] -0.07065779  1.46264688  0.66467338  0.56187684 -0.33905853
##
## [[4]]
## [1] -Inf  Inf
```

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

```
##    user  system elapsed
##   0.484   0.035   0.587
```

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] 0.3903494 0.5107521 0.5438679 0.9327843 2.2745782
##
## [[2]]
## [1] -0.3903494 -0.5107521 -0.5438679 -0.9327843 -2.2745782
##
## [[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] 0.06867557 0.07863842 0.21166391 0.21881569 0.33182613 0.37132615
##  [7] 0.71521893 0.73841366 0.77456309 0.91411376
##
## [[2]]
##  [1]  0.9563171  0.9348068  0.6229507  0.6047240  0.2921986  0.1703269
##  [7] -1.4134568 -1.5833695 -1.8808176 -3.8108511
##
## [[3]]
##  [1]  -2.147479  -2.170700  -2.536989  -2.560215  -2.993233  -3.181300
##  [7]  -7.022939  -7.645659  -8.871662 -23.286618
##
## [[4]]
## [1] 0 1
```

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

```
##    user  system elapsed
##   0.838   0.015   0.879
```