Monte Carlo Integration using Variance Reduction methods

Rustam Karpykov

4/9/2020

Programming Monte Carlo Advanced function:

```
Monte_Carlo_Advanced <- function(g, lb, ub, n = 10^5, plot = FALSE,
                                var.reduc = FALSE,   # "var.reduc" which takes
                                                     #  a boolean (True or False).
                                type = "Antithetic", # "type" which takes a string
                                                     # from user as "Antithetic" or "Control".
                                cv = "NULL") {       # I've desided to add extra parameter 'cv'
                                                     # which takes a list of 'Control Variates'
                                                     # ex: c(f1, f2...), whenever user
  #Generatiing X's.------------------------------------------------------|
  x.classic <- runif(n, lb, ub)
  #Monte Carlo Integration, estimator and variance.-----------------------|
  exp.classic <- mean(sapply(x.classic, g)) * (ub - lb)
  var.classic <- mean(sapply(x.classic, g)^2) - mean(sapply(x.classic, g))^2
  #Finding R integrate function's estiomation.---------------------------|
  exp.R.integrate <- integrate(g, lb, ub)
  #Defining (Error) difference between Real and Estimated values.---------|
  error.classic_vs_R.integrate <- abs(exp.classic - exp.R.integrate$value)

  div_line <- strrep("-", 74) # Just a line that will be used to divide output parts.

  if (!var.reduc){
     print(matrix(c(exp.classic, var.classic), nrow = 1,
       dimnames =  list("Classical Approach", c("Estimated Mean", "Variance"))))
     cat(div_line, "\n|Note: In order to reduce variance you may want to use `Antithetic variables` or
  }

  # In case of user defining wrong cv (Control Variates) parameter--------|
  if(var.reduc & type == "Control" & !is.list(cv)) {
     cat("Please specify your Control Variates in a list. ex: cv = c(f1, f2...)")
  }
  #----------------------------------------------------------------------|
  if(var.reduc & type == "Antithetic"){
    x.ant <- runif(n / 2, lb, ub)

    exp.ant.pt1 <- mean(sapply(x.ant, g)) * (ub - lb)
    exp.ant.pt2 <- mean(sapply(ub - x.ant, g)) * (ub - lb)
    exp.ant <- (exp.ant.pt1 + exp.ant.pt2) / 2
```

```r
  z1.1 <- g(x.ant)
  z1.2 <- g(ub - x.ant)

  var.ant <- (var(z1.1) + var(z1.2) + 2 * cov(z1.1, z1.2)) / 4

  result.ant <- matrix(c(exp.classic, exp.ant,
                         var.classic, var.ant),nrow=2,
                         dimnames = list(c("Classical Method","Antithetic variables"),
                                         c("Estimated Mean","Variance")))
  print(result.ant)
  var.red.ant = 100 * ((var.classic - var.ant) / var.classic)
  cat(div_line, "\n|The Antithetic variables approach achived",
      round(var.red.ant, 2), "%", "reduction in variance.|")
}
if(var.reduc & type == "Control" & is.list(cv)){
  x.cv <- runif(floor(n / length(cv)), lb, ub)
  if (length(cv) == 1){
    c <- -cov(g(x.cv), cv[[1]](x.cv)) / var(cv[[1]](x.cv))

    exp.f <- (1/(ub - lb)) * integrate(cv[[1]], lb, ub)$value

    exp.cv.1 <- mean(g(x.cv) + c * (cv[[1]](x.cv) - exp.f)) * (ub - lb)
    var.cv.1 <- var(g(x.cv) + c * (cv[[1]](x.cv) - exp.f))

    result.cv.1 <- matrix(c(exp.classic, exp.cv.1,
                           var.classic, var.cv.1),nrow=2,
                           dimnames = list(c("Classical Approach","Control Variates"),
                                           c("Estimated mean","Variance")))
    var.red.cv = 100 * ((var.classic - var.cv.1) / var.classic)
    print(result.cv.1)
    cat(div_line, "\n|The control variates approach achived",
        round(var.red.cv, 2), "%", "reduction in variance.|")
  }
  else {
    # Matrix of g and f's of X------------------------------------------/
    g_by_fs <- c(g(x.cv))
    for (i in 1:length(cv)) {
      g_by_fs <- cbind(g_by_fs, cv[[i]](x.cv))
    }
    # -----------------------------------------------------------------/
    # Vecror of Estimators of Control Variates----------------------------/
    mu <- c(1)
    for (i in 1:length(cv)) {
      mu <- c(mu, integrate(cv[[i]], lb, ub)$value)
    }
    # -----------------------------------------------------------------/
    # Fitting Linear Model------------------------------------------------/
    L <- lm(g_by_fs[, 1] ~ g_by_fs[, -1])

    exp.cv <- sum(L$coefficients * mu)
    var.cv <- summary(L)$sigma^2

    result.cv <- matrix(c(exp.classic, exp.cv,
```

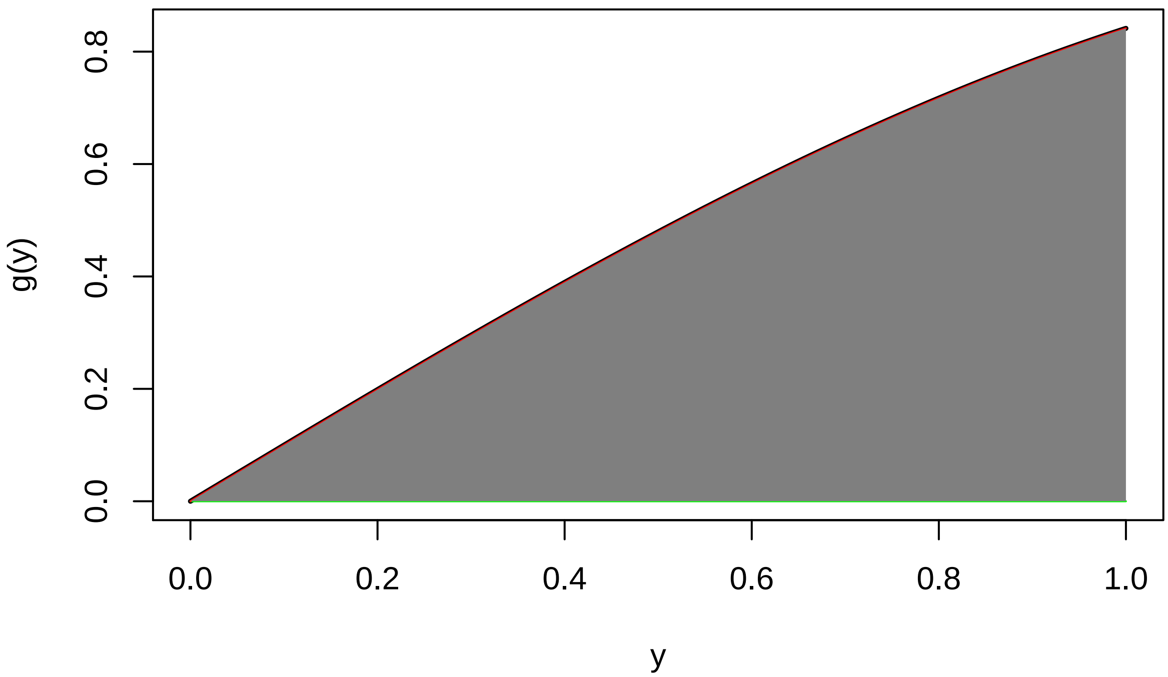
```
                        var.classic, var.cv),nrow=2,
                        dimnames = list(c("Classical Approach","Control variates"),
                                        c("Estimated mean","Variance")))
      var.red.cv = 100 * ((var.classic - var.cv) / var.classic)
      print(result.cv)
      cat(div_line, "\n|The Control Variates approach achived",
          round(var.red.cv, 2), "%", "reduction in variance.|")
    }
  }

  # Drawing plot.-----------------------------------------------------------/
  if(plot){
    y <- seq(lb, ub, 0.001)
    y.low <- rep(0, times = length(y))
    plot(y, g(y), type = "l", lwd = 2.5, main = body(g))
    lines(y, y.low, col = 'green')
    lines(y, g(y), col = 'red')
    polygon(c(y, rev(y)), c(g(y), rev(y.low)),col = "grey50", border = NA)
  }
  # -----------------------------------------------------------------------/
}
```

Testing: Let's test a function we've built:

Try finding integral of sine function on (0, 1) using classical approach

```
Monte_Carlo_Advanced(sin, 0, 1, n = 10^5, plot = TRUE)
```

```
##                     Estimated Mean   Variance
## Classical Approach       0.459542 0.06108161
## ---------------------------------------------------------------------
## |Note: In order to reduce variance you may want to use `Antithetic variables` or `Control Variates` m
```



Try finding integral of sine function on (0, 1) using classical approach + Antithetic Variables method of Variance Reduction

```r
Monte_Carlo_Advanced(sin, 0, 1, n = 10^5, plot = FALSE, var.reduc = TRUE)
```

```
##                         Estimated Mean     Variance
## Classical Method             0.4596318 0.0613197094
## Antithetic variables         0.4596893 0.0003066266
## ------------------------------------------------------------------------
## |The Antithetic variables approach achived 99.5 % reduction in variance.|
```

Try finding integral of sine function on (0, 1) using classical approach + Single Control Variate method of Variance Reduction

```r
Monte_Carlo_Advanced(sin, 0, 1, n = 10^5, plot = FALSE,
                     var.reduc = TRUE, type = "Control", cv = c(cos))
```

```
##                       Estimated mean    Variance
## Classical Approach         0.4604406 0.061339749
## Control Variates           0.4595421 0.005540762
## ------------------------------------------------------------------------
## |The control variates approach achived 90.97 % reduction in variance.|
```

Try finding integral of sine function on (0, 1) using classical approach + Multiple Control Variate method of Variance Reduction

```
Monte_Carlo_Advanced(sin, 0, 1, n = 10^5, plot = FALSE,
                     var.reduc = TRUE, type = "Control", cv = c(cos, tan))
```

```
##                      Estimated mean    Variance
## Classical Approach      0.4592420 0.061316291
## Control variates        0.4599208 0.002099817
## ------------------------------------------------------------------------
## |The Control Variates approach achived 96.58 % reduction in variance.|
```

Last but not least lets check whether [In case of user defining wrong cv (Control Variates) parameter] is working correctly

```
Monte_Carlo_Advanced(sin, 0, 1, n = 10^5, plot = FALSE,
                     var.reduc = TRUE, type = "Control", cv = cos)
```

```
## Please specify your Control Variates in a list. ex: cv = c(f1, f2...)
```

Great! It works as well

Objective: ***Since we already have function which evaluates any integral using Monte Carlo integration technique in cope with Variance reduction techniques, we can directly calculate given integral. Also we want to reduce variance of estimation, and all we need for that is to define 3 control variates. We will call Monte_Carlo_Advanced() several times. One for each of functions negatively correlated with g(x), and one with all 3 cv's simmultaniously) and finally for a few last tests***

```
x <- runif(10^5, 0, 1)
g <- function(x) {exp(x) / (1 + x^2)}
f1 <- function(x) {1 - x}
f2 <- function(x) {-log(x)}
f3 <- function(x) {1/exp(x)}
```

Correletion between g(x) and assumed Control Variates

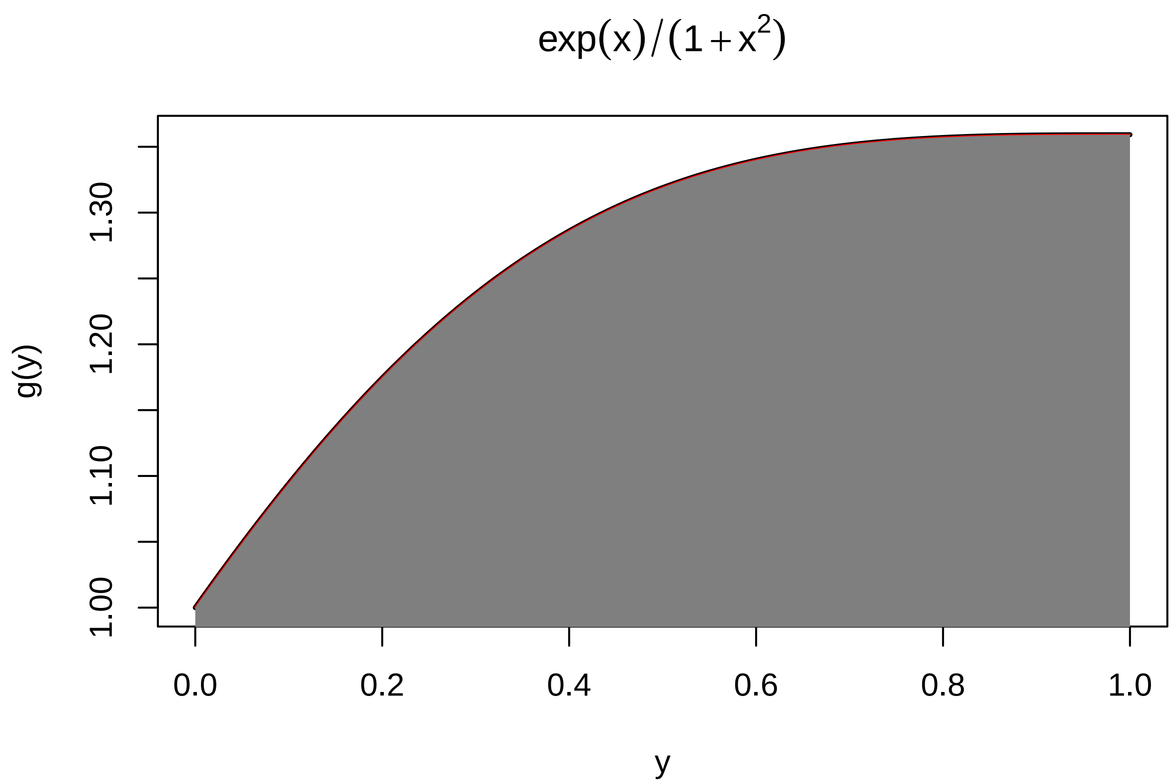
```
matrix(c(cor(g(x), f1(x)),cor(g(x), f2(x)), cor(g(x), f3(x))), nrow = 1,
       dimnames =  list("value", c("Cor(g & f1)",
                                   "Cor(g & f2)",
                                   "Cor(g & f2)")))
```

```
##       Cor(g & f1) Cor(g & f2) Cor(g & f2)
## value  -0.9097286  -0.9534682  -0.9551291
```

Several Control variates approach (Using 3 Control variates)

```
Monte_Carlo_Advanced(g, 0, 1, n = 10^5, plot = TRUE,
                     var.reduc = TRUE, type = "Control",
                     cv = c(f1, f2, f3))
```

```
##                      Estimated mean     Variance
## Classical Approach       1.270591 1.072413e-02
## Control variates         1.270708 1.171771e-05
## ----------------------------------------------------------------------
## |The Control Variates approach achived 99.89 % reduction in variance.|
```



Single control variate f1() approach

```
Monte_Carlo_Advanced(g, 0, 1, n = 10^5, plot = FALSE,
                     var.reduc = TRUE, type = "Control", cv = c(f1))
```

```
##                      Estimated mean    Variance
## Classical Approach       1.270621 0.010709595
## Control Variates         1.270678 0.001838766
## ----------------------------------------------------------------------
## |The control variates approach achived 82.83 % reduction in variance.|
```

Single control variate f2() approach

```r
Monte_Carlo_Advanced(g, 0, 1, n = 10^5, plot = FALSE,
                     var.reduc = TRUE, type = "Control", cv = c(f2))
```

```
##                      Estimated mean     Variance
## Classical Approach       1.270824 0.0106697306
## Control Variates         1.270834 0.0009807774
## ---------------------------------------------------------------------------
## |The control variates approach achived 90.81 % reduction in variance.|
```

Single control variate f3() approach

```r
Monte_Carlo_Advanced(g, 0, 1, n = 10^5, plot = FALSE,
                     var.reduc = TRUE, type = "Control", cv = c(f3))
```

```
##                      Estimated mean     Variance
## Classical Approach       1.270407 0.0106692200
## Control Variates         1.270651 0.0009288119
## ---------------------------------------------------------------------------
## |The control variates approach achived 91.29 % reduction in variance.|
```

Antithetic Variable approach just for sake of comparison

```r
Monte_Carlo_Advanced(g, 0, 1, n = 10^5, plot = FALSE,
                     var.reduc = TRUE)
```

```
##                        Estimated Mean    Variance
## Classical Method           1.270808 0.010633068
## Antithetic variables       1.270638 0.001781637
## ---------------------------------------------------------------------------
## |The Antithetic variables approach achived 83.24 % reduction in variance.|
```

Final check of classical approach

```r
Monte_Carlo_Advanced(g, 0, 1, n = 10^5, plot = FALSE)
```

```
##                      Estimated Mean  Variance
## Classical Approach       1.270848 0.0106732
## ---------------------------------------------------------------------------
## |Note: In order to reduce variance you may want to use ‘Antithetic variables‘ or ‘Control Variates‘ r
```