ACM

Anders, Christian and Markus

26 10 2019

First Draft of project:

We decided to create two different classes (and corresponding methods):

1) Copulas (Anders' focus) and

2) Simulations (Christian's and my focus)

2) First ideas regarding the class "Simulations" (based MC methods)

We are going to use the class "Simulations" together with subclasses:

1) Implementing MC estimates for integrals

Input: Function you want to integrate

Output: List (Class: "Simulations", Subclass:"MC_Integral") containing the value of the integral, the 95%-confidence interval, number of simulations and the function f.

```
int <- function(f, a, b, n = 100000){
  U <- runif(n)
  U_ab <- a + (b-a)*U
  sim <- f(U_ab)
  estimate <- (b-a)*sim
  value <- mean(estimate)
  halfwidth <- sd(estimate)/sqrt(n)*qnorm(0.975)
  confidence_interval <- c(value - halfwidth, value + halfwidth)
  structure(list("integrand" =f, "lower bound" = a, "upper bound" = b, "Value of integral" = value,
"asymptotic 95% confidence interval" = confidence_interval, "number of simulations" = n),
class = c("MC_integral", "simulation"))

}

#example
f <- function(x){x^2}

MCint <- int(f, -2, 2)

#simulated value
MCint$'Value of integral'
```

```
## [1] 5.326157
```

```
#theoretical value
2*2^3/3
```

```
## [1] 5.333333
```

2) Simulating random variables

2.1) via inverse distribution

Input: inverse distribution function

Output: List (Class: "Simulations", Subclass:"invsample") containing the samples and the number of simulations.

Suppose we know the inverse distribution function of a random variable (with the distribution F), called F^{-1}.

```
inv_sample <- function(Finv, N = 1, F_distribution = NA){
  U <- runif(N)
  structure(list(samples=Finv(U), "number of simulations"=N, "distribution function" = F_distribution),
class = c("invsample", "simulation"))
}
```

Example: The distribution function of an exponential with rate 1 is given by $F(x) = 1 - e^{-x}$ and thus $F^{-1}(x) = -ln(1-x)$

```
Finv <- function(x){-log(1-x)}
F_dist <- function(x){1-exp(-x)}

inv_samples <- inv_sample(Finv, N = 10000)
inv_samples_withF <- inv_sample(Finv, N = 10000, F_distribution = F_dist)
inv_5samples <- inv_sample(Finv, N = 5)
```

2.2) acceptance rejection

Input: proposal and density we want to sample from

Out: List (Class: "Simulations", Subclass:"invsample") containing the samples, the proposal, the target density and the number of simulations.

In acceptance rejection sampling, we want to sample from a density f. Furthermore, we are not able to find an explicit expression of the inverse distribution. In this case, we can us acceptance rejection sampling.

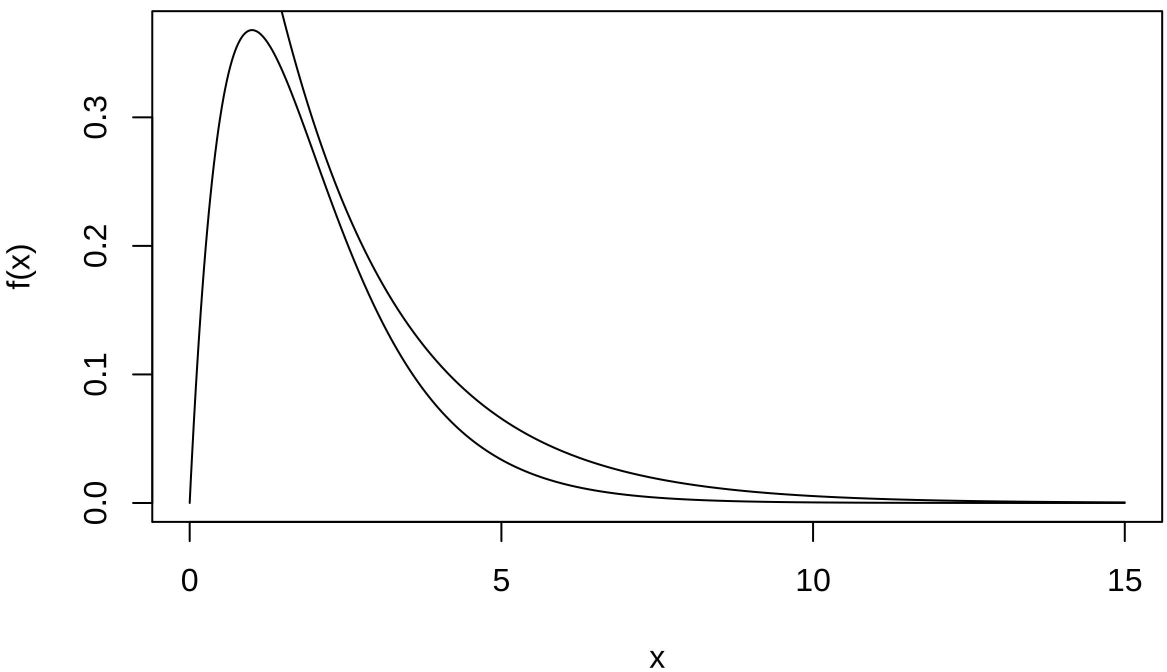
We therefore need a density g such that $f(x) \leq Cg(x)$. You have to provide this C in the following function. Per default, we use the density $g(x) = \lambda e^{-\lambda x}$ (exponential distributed with rate λ) as proposal in our function. Otherwise you have to set `exponential`to `FALSE` and provide your own proposal density together with the corresponding inverse distribution function.

```
ARsim <- function(f, rate = 1, C, N, exponential = TRUE, propdensity = NA, inveresepropdensity = NA){
  if(exponential == TRUE){
    expdensity <- function(x){exp(-rate*x)}
    sum <- 0
    Y <- rep(0,N)
    while(sum < N){
        X <- rexp(1,rate = rate)
        U <- runif(1)
        if(f(X)/(C*expdensity(X))>U){
          sum <- sum + 1
          Y[sum] <- X
        }
    }
    structure(list(samples = Y, "number of simulations" = N, "proposal density" = expdensity,
"target density" = f),class = c("AR", "simulation"))
  } else{
    sum <- 0
    Y <- rep(0,N)
    while(sum < N){
        U_X <- runif(1)
        X <- inveresepropdensity(U_X)
        U <- runif(1)
```

```
        if(f(X)/(C*propdensity(X))>U){
          sum <- sum + 1
          Y[sum] <- X
        }
    }
    structure(list(samples = Y, "number of simulations" = N, "proposal density" = propdensity,
"target density" = f),class = c("AR", "simulation"))
  }

}
```

Example: Suppose we want to sample from $f(x) = xe^{-x}$ for $x > 0$. One can find graphically that $f(x) \leq 1.6g(x)$ for $\lambda = \frac{1}{2}$, since

```
C <- 1.6
lambda <- 0.5
x <- seq(0,15,0.01)
f <- function(x) {x*exp(-x)}
plot(x, f(x), type = "l")
lines(x, C*lambda*exp(-0.5*x))
```



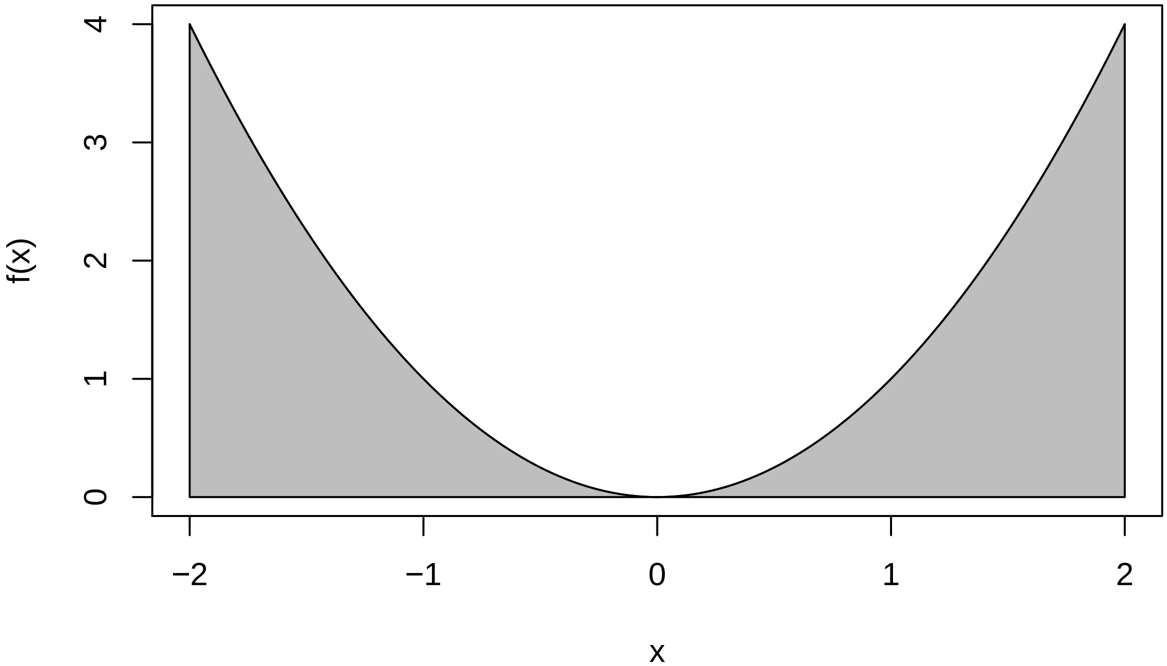

```
ARsimulation <- ARsim(f, lambda, C, 100000)
```

Methods for the class "simulations"

We want to define specific plot functions:

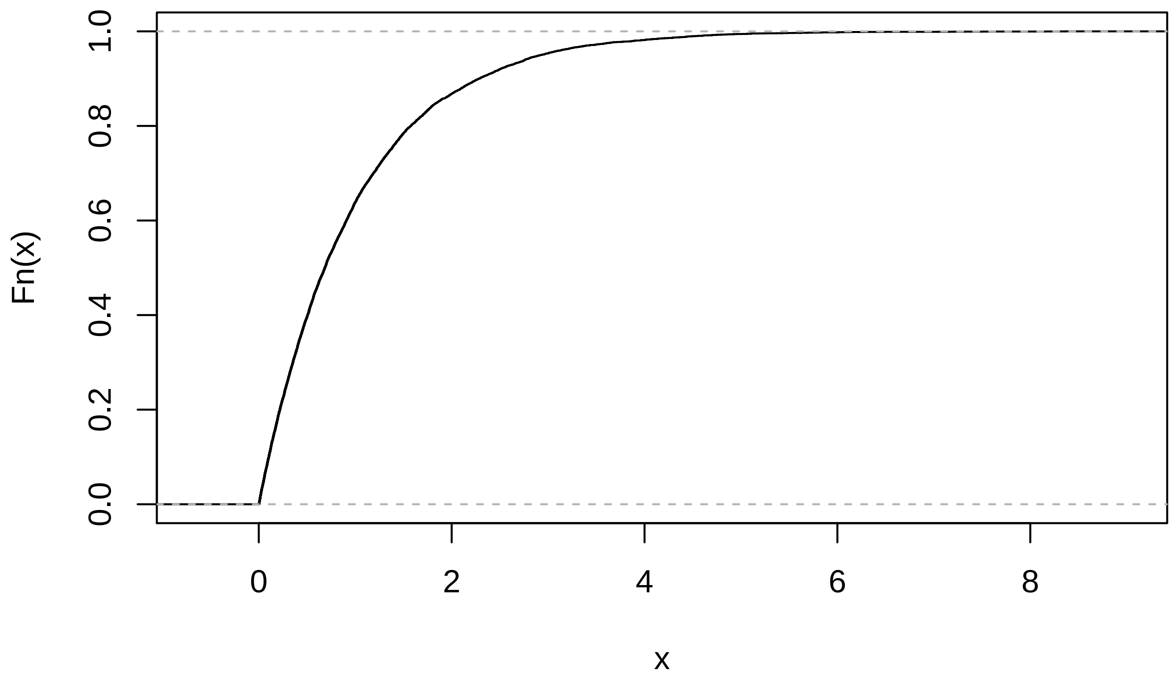
```
plot.MC_integral <- function(MC){
  x <- seq(MC$'lower bound',MC$'upper bound', 0.01)
  y <- MC$'integrand'(x)
  plot(x,y, type = "l", xlab = "x", ylab = "f(x)")
  polygon(x = c(x,MC$'upper bound', MC$'lower bound'),
        y = c(y,0,0),
```

```r
        col = "grey")
}

plot(MCint)
```



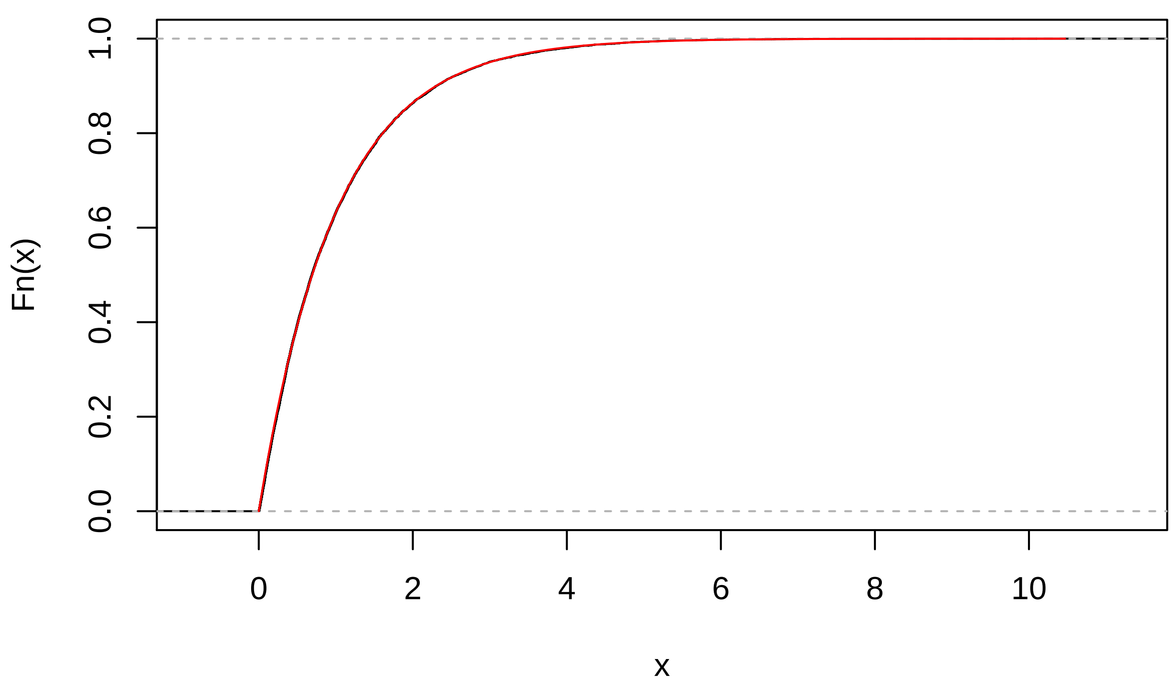

```r
plot.invsample <- function(inv_sample){
  if(inv_sample$"number of simulations" < 10){
        warning("number of simulations is to small")
     } else{
  if(is.function(inv_sample$"distribution function")) {
    y <- inv_sample$samples
    plot(ecdf(y), main = "Empirical cumularive distribtion function compared to theoretical")
    x <- seq(min(y), max(y), 0.01)
    lines(x, inv_sample$"distribution function"(x), col = "red")
  } else {
    y <- inv_sample$samples
    plot(ecdf(y), main = "Empirical cumularive distribtion function")
  }
 }
}

plot(inv_samples)
```

Empirical cumularive distribtion function




```r
plot(inv_samples_withF)
```

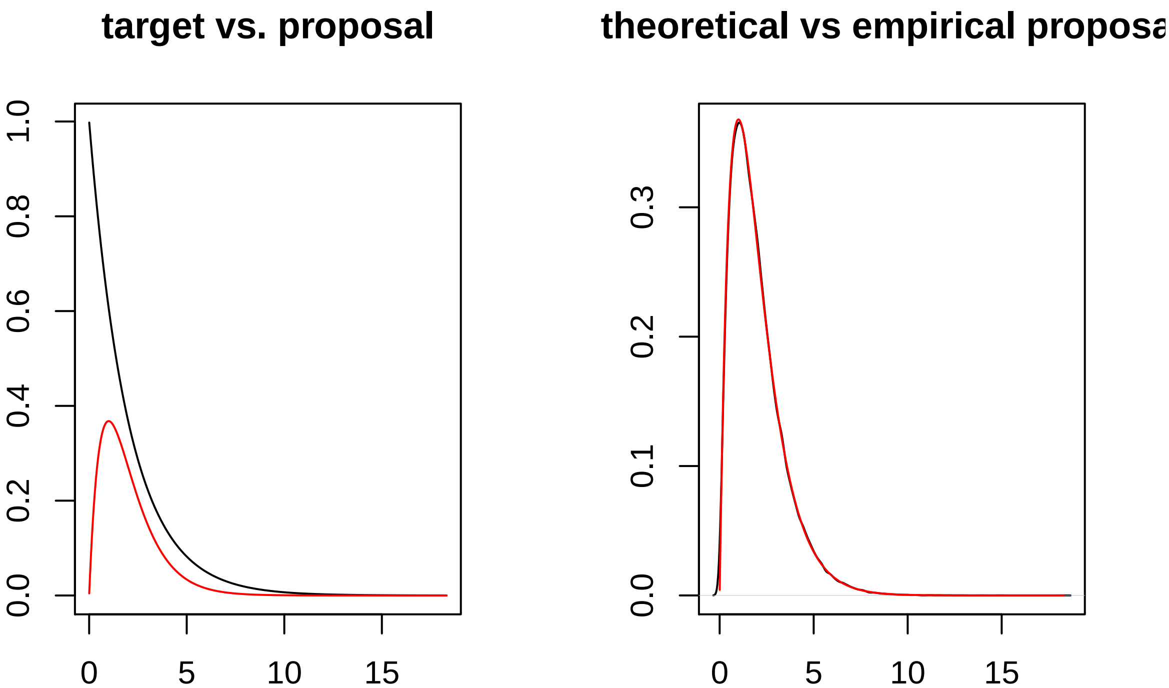
Empirical cumularive distribtion function compared to theoretical




```r
plot(inv_5samples)
```

```
## Warning in plot.invsample(inv_5samples): number of simulations is to small
```

```r
plot.AR <- function(ARsample){
  par(mfrow = c(1,2))
  x <- seq(min(ARsample$samples),max(ARsample$samples),0.01)
  plot(x, ARsample$"proposal density"(x), type = "l", main = "target vs. proposal",
xlab = "", ylab = "")
  lines(x, ARsample$"target density"(x), col = "red")
  plot(density(ARsample$samples), main = "theoretical vs empirical proposal", xlab = "", ylab = "")
  lines(x, ARsample$"target density"(x), col = "red")
}

plot(ARsimulation)
```



Specific print methods:

Other interesting methods: summary, …