732A90 - Exam - March 2016

Assignment 1

1.1 Inverse CDF

We can use the inverse CDF method to sample from the distribution. Assuming we have a PDF of the target distribution we can follow the following schema:

1. If the target distribution is a proper distribution (area under curve = 1) go to step 3, otherwise to step 2.
2. Integrate the function and calculate the area under the curve, lets call this value β; The normalization factor is then given by $\frac{1}{\beta}$; Multiply the original distribution with $\frac{1}{\beta}$ to get a proper density function.
3. Integrate the density function to get its primitive function (cumulative distribution function (CDF)).
4. Calculate the inverse of the CDF.
5. Sample from the inverse with $u \sim Unif(0, 1)$

$$F(x) = \int_{-\infty}^{x} 1.5\sqrt{s}\ ds = \int_{0}^{x} 1.5\sqrt{s}\ ds = \left[s^{\frac{3}{2}}\right]_0^x = x^{\frac{3}{2}}$$

Now the inverse can be calculated:

$$F(x)^{-1} : x = y^{\frac{3}{2}} y = x^{\frac{2}{3}}$$

We can now sample from this distribution.

```
# Define the function of the inverse cdf
inverse_cdf <- function(x) {
  return(x^(2/3))
}

# Generate 10000 random values uniform(0, 1) and
# sample from the inverse cdf
u <- runif(10000)
res_11 <- inverse_cdf(u)

# Generate the values of the real distribution
x <- seq(0.001, 0.999, 0.001)
y <- 1.5 * sqrt(x)

# Plot the distributions in a histogram
hist(res_11, freq = FALSE,
     breaks = 30,
     xlab = "x", ylab = "Density",
     main = "Distribution of f(x)",
     col = "grey")

# Plot the density estimate of the sample (blue) and the true values
# of the function (red)
lines(density(res_11), col = "blue", lwd = 2)
lines(x, y, col = "red", lwd = 2)
```

```r
# Add a legend
legend(x = 0, y = 1.5,
       legend = c("sample", "sample density", "true density"),
       col = c("grey", "blue", "red"),
       pch = c(16, 16, 16),
       cex = 0.8)
```

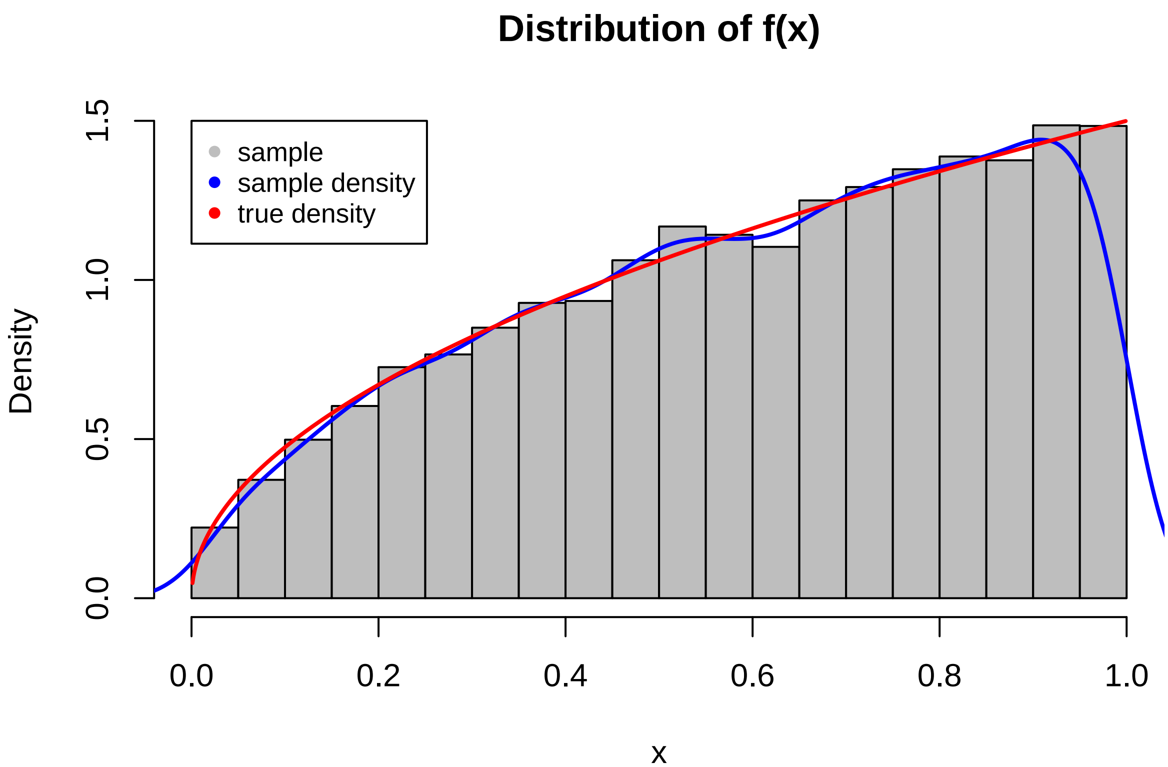


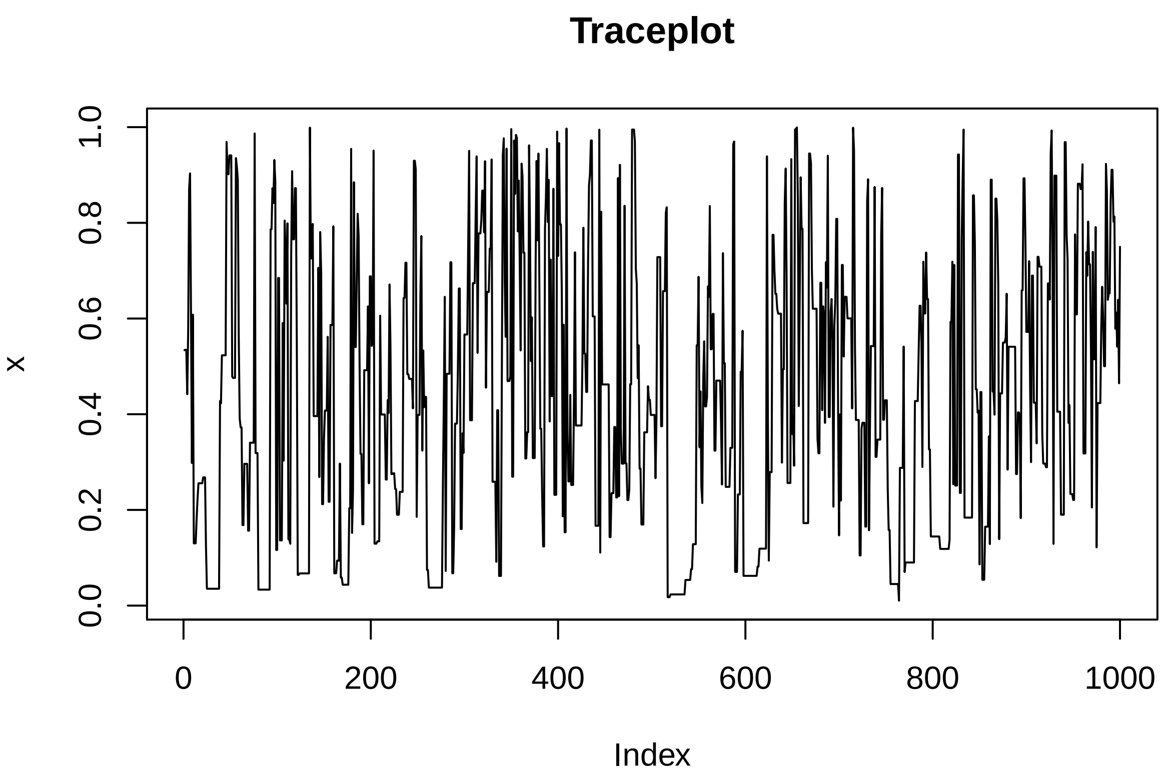
Figure 1: Histogram of the sampled values.

As we can see in figure 1 the histogram of the sampled values is very close to the true values. We can assume that the sample comes from $f(x)$.

1.2 Metropolis Hastings

```r
# Template for metropolis hastings algorithm:

dt <- function(x) {
  if (x <= 0 || x >= 1) {
    stop("x has to be greater than 0 and smaller than 1.")
  }
  return(sqrt(x) / (x+0.1))
}

dp <- function(x, xt) {
  dbeta(x, xt, 0.5)
}
```

```r
rp <- function(x) {
  rbeta(1, x, 0.5)
}

metropolis_hastings <- function(x_0, t_max, dt, dp, rp) {
  # Perform Metropolis-Hastings sampling of the specified density.
  #
  # Args:
  #   x_0   starting value.
  #   t_max maximum numer of iterations.
  #   dt    density function from which we want to sample.
  #   dp    proposal density function, should accept 2 arguments:
  #           x:  value at which to compute density.
  #           x_t: value on which the rq is conditioned.
  #   rp    proposal random number generator, should accept 1 argument:
  #           x_t: value on which the rq is conditioned.
  #
  # Returns:
  #   Vector of sampled points of length t_max.

  x_t <- x_0
  x <- vector("numeric", t_max) # pre-allocate

  # Metropolis-Hastings
  for (t in 1:t_max) {
    # Generate a candidate
    y <- rp(x_t)
    # Generate U
    u <- runif(1, 0, 1)
    # Compute alpha
    alpha <- min(1, (dt(y) * dp(x_t, y)) / (dt(x_t) * dp(y, x_t)))
    # Accept the jump or stay in x_t
    if (u < alpha) {
      x_t <- y
    }
    # Sace x_t in vector
    x[t] <- x_t
  }

  return(x)
}


set.seed(12345)
res_12 <- metropolis_hastings(0.1, 1000, dt, dp, rp)

plot(res_12, type = 'l',
     main = "Traceplot",
     xlab = "Index", ylab = "x")
```




```
# Optional code for plotting the histogram

# hist(res_12, breaks = 28, freq = FALSE,
#      col = "grey",
#      xlim = c(0, 1), ylim = c(0, 2),
#      xlab = "x", ylab = "Density",
#      main = "Metropolis hastings sample")
# x <- seq(0.01, 0.99, 0.01)
# y <- sapply(x, dt)
# lines(x, y, col = "red", lwd = 2)
# lines(density(res_12), col = "blue", lwd = 2)
# legend("topright",
#        legend = c("Sampled", "Sampled density", "True density"),
#        col = c("grey", "blue", "red"),
#        pch = c(16, 16, 16),
#        cex = 0.8)
```

Convergence: The values of x converge between 0 and 1.

Mixing: Despite of some areas (where the values stuck for a few iterations) the mixing seems to be good.

Burn-In: The starting value $x0 = 0.1$ is already in the convergence interval. So we can basically say that there is no burn in or only a very short burn in period.

2.3 Estimating the integral

```
f <- function(x) {
  (x*sqrt(x)) / (x+0.1)
}
```

```r
true_integral_value <- integrate(f, 0, 1)$value
sample1_integral_value <-
sample2_integral_value <- round(mean(res_12), 2)
```

Samples from 1 I DO NOT HAVE A CLUE HOW TO DO THAT!

Samples from 2 NOT SURE IF THIS IS CORRECT: The value of the integral is the mean value of the samples values and can be derived as 0.43.

Using the integrate function The value of the integral function is 0.5466419.