

Tutorial 2 – Giovanni Filomeno - 12315325

Exercise 7

a) Weibull random variable

My code for the Weibull random variable is presented in Figure 1.

```
6 ▾ rweibull_inv <- function(n, alpha, beta) {  
7   U <- runif(n) # Suggested as Hint  
8   (-log(U) / alpha)^(1 / beta)  
9 ▸ }  
10
```

Figure 1: Weibull algorithm

b) Simulate M=1000 Weibull points and compare CDFs

A way to compare the two CDFs is to plot them in the same graph. This can be easily achieved with the code in Figure 2 which leads to the graph in Figure 3. The empirical step function comes close to the theoretical CDF, which is exactly the behavior I expect from a realization of 1000 points.

```
11 x <- rweibull_inv(M, alpha, beta)  
12 F <- function(t) 1 - exp(-alpha * t^beta) # true CDF  
13  
14 plot(ecdf(x), do.points = FALSE, main = "Weibull(α=2, β=4): ecdf vs. true CDF",  
15      xlab = "x", ylab = "F(x)")  
16 curve(F, from = 0, to = max(x), add = TRUE, lwd = 2)  
17 legend("bottomright", legend = c("empirical CDF", "true CDF"),  
18      lty = c(1,1), lwd = c(1,2), col = c("black", "black"))  
19 |
```

Figure 2: Comparison code

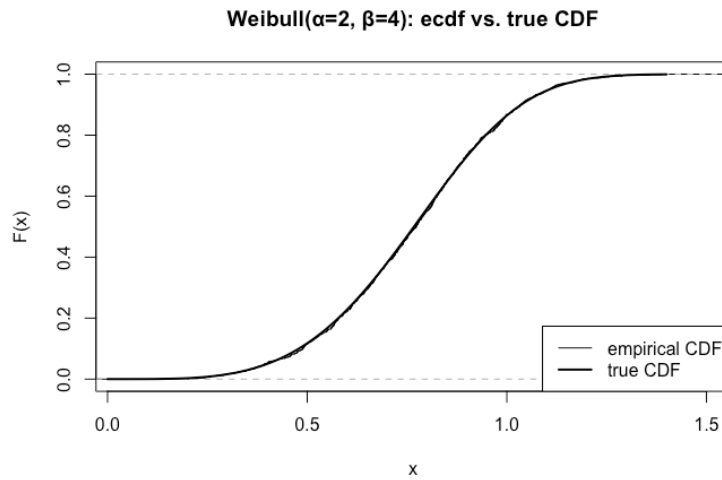


Figure 3: Comparison graph b)

c) Comparison with Exponential(λ)

Using the same approach used in a) and b), I can construct the exponential distribution imposing the Weibull parameters as $\beta = 1$ and $\alpha = \lambda$. Figure 4 shows the differences between the two CDFs which seem to be more marked than the exercise b).

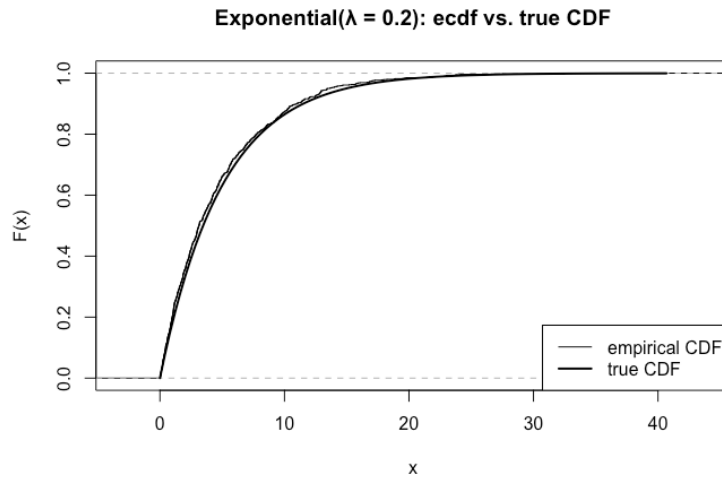


Figure 4: Comparison graph c)

Exercise 8

a) Implement the algorithm

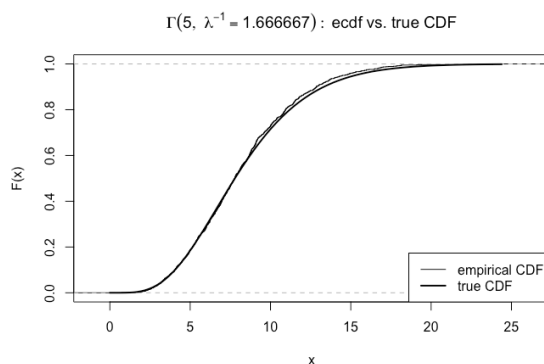
My code for the Γ -distributed random variable is presented in Figure 5.

```
Ex7.R x Ex8.R x
Source on Save
1 rgamma_intshape_inv <- function(M, shape, rate = 1) {
2   # Sanity check
3   if (shape <= 0 || rate <= 0)
4     stop("Both 'shape' and 'rate' must be positive.")
5   if (abs(shape - round(shape)) > .Machine$double.eps^0.5)
6     stop("'shape' must be an integer for this generator.")
7
8   k <- as.integer(round(shape))
9
10  U <- runif(M * k)
11  mat <- matrix(-log(U) / rate, nrow = k)
12  colSums(mat)
13 }
14
```

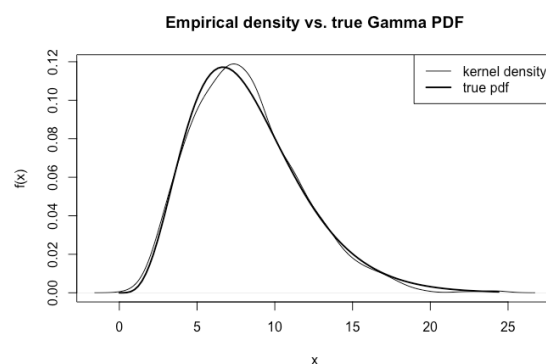
Figure 5: Gamma-distributed algorithm

b) Comparison of CDFs + c) Comparison of the PDFs

Similarly to the previous exercise, it is possible to construct the empirical and theoretical CDF and PDF. For the CDF comparison (cf., Figure 6 (a)), the two curves follow the same path, with a small vertical gap that happens in the range $7 < x < 15$ which push the probability into the right tail. For the PDF comparison (cf., Figure 6 (b)), the shift appears around $x \approx 3$. Both effects shrink with a larger M .



(a)



(b)

Figure 6: CDF (a) and PDF (b) comparison

Exercise 9

a) Implement the algorithm

My code for the $\text{simx}(M)$ is presented in Figure 7.

```
1 simx <- function(M) {  
2   replicate(M, {  
3     # Initialization  
4     s <- 0  
5     n <- 0  
6     repeat {  
7       s <- s + log(runif(1)) # add log(U_i)  
8       if (s > -3) {  
9         n <- n + 1           # if above threshold -> count it  
10      } else {  
11        return(n)           # threshold crossed -> stop  
12      }  
13    }  
14  })  
15 }  
16
```

Figure 7: $\text{simx}(M)$

b) Estimate of statistical parameters based on $M = 10^4$ realisations

The results of the simulation show $\mathbb{E}[X] = 2.9856$ and $\text{Var}(X) = 2.961689$. The following table shows the results for $\mathbb{P}(X = i)$ and, in combination with the part c) also the Poisson distribution with parameter $\lambda = 3$ (named P_theory)

i	$\mathbb{P}(X = i)$	P_theory
0	0.0520	0.0497900
1	0.1504	0.1494000
2	0.2196	0.2240000
3	0.2246	0.2240000
4	0.1722	0.1680000
5	0.1009	0.1008000
6	0.0490	0.0504100
7	0.0203	0.0216000
8	0.0075	0.0081020
9	0.0024	0.0027010
10	0.0009	0.0008102
11	0.0002	0.0002210

c) Do the estimated values agree with the true ones?

Yes. The mean and variance are close to 3 (the defining property of a Poisson with $\lambda = 3$). Additionally, as shown in b), the shift between the two probabilities is small. Last but not least, the p-value has been evaluated and it has value of 0.69 which means no evidence against the null hypothesis.

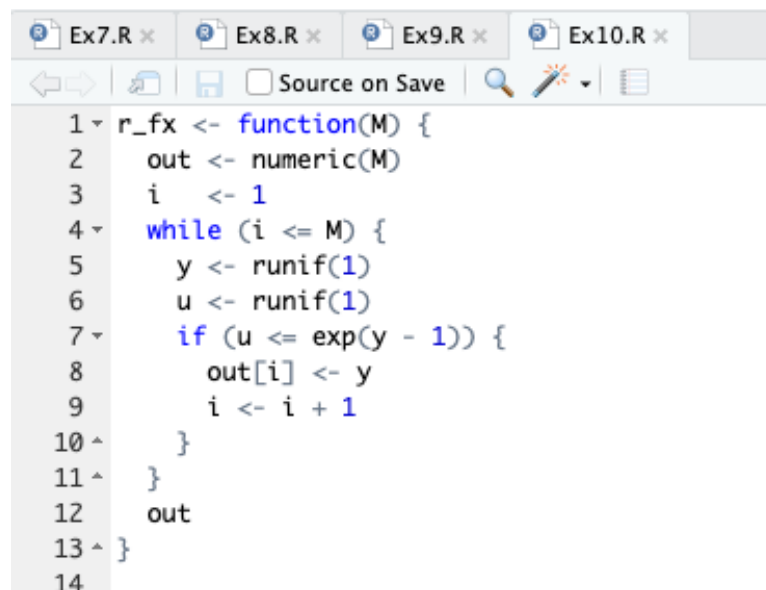
Exercise 10

a) Implement the algorithm

Given the interval $[0,1]$, the most convenient PDF is $g(x) = 1$ for $0 < x < 1$. This means that the formula $f(x) \leq C \cdot g(x)$ simplifies to $f(x) \leq C$, which leads to find the maximum of $f(x)$. A standard way to do so is to impose derivative to zero, however in this case the derivative is always strictly positive, which means that the maximum is at $x = 1$ (border of the interval). The max is at $f(1) = e/(e - 1)$.

Now I can define my rejection condition as $U \leq \frac{f(Y)}{C \cdot g(Y)} = \frac{e^Y/(e-1)}{(e/(e-1)) \cdot 1} = e^{Y-1}$, where $Y \sim U(0,1)$.

My code for the rejection sampler algorithm is presented in Figure 8.



```
Ex7.R x Ex8.R x Ex9.R x Ex10.R x
Source on Save
1 r_fx <- function(M) {
2   out <- numeric(M)
3   i <- 1
4   while (i <= M) {
5     y <- runif(1)
6     u <- runif(1)
7     if (u <= exp(y - 1)) {
8       out[i] <- y
9       i <- i + 1
10    }
11  }
12  out
13 }
14
```

Figure 8: Rejection sampler algorithm

b) Empirical and Theoretical Mean and Variance

The empirical mean and variance are 0.58161352 and 0.07923278 respectively, while the theoretical value for mean is 0.58197671 and for variance is 0.07932641. Both sample estimates match the exact values up to three decimal places, which is what I expect with 10^5 draws.

Exercise 11

a) Acceptance probability: simulation vs true value

The hint of the exercise suggests using the uniform distribution which has density function of $g(x) = 1$ for $0 < x < 1$. This means that the formula $f(x) \leq C \cdot g(x)$ simplifies to $f(x) \leq C$. To find C , I have to find the max of the function by imposing its derivative equal to zero. The derivative can be easily calculated as $f'(x) = 20(1-x)^2(1-4x)$ which is equal to zero in $x = 1$ (not included in the range) and $x = 1/4$ (included in the range). Substituting $x = 1/4$ back to the original function, I can find the max which is $f(1/4) = 135/64$.

Using $M = 10^5$, both empirical and theoretical probability are 0.4741.

b) Compare empirical and true PDF

Figure 9 shows the comparison between the empirical and true PDF. Also in this case, the two curves overlap.

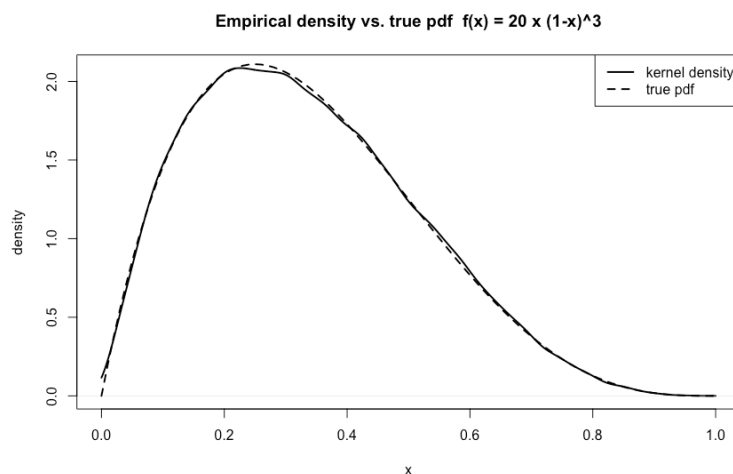


Figure 9: Empirical vs true PDF

Exercise 12

a) Implement Box-Muller algorithm

My code for the Box-Muller is presented in Figure 10.

```
1 rnorm_bm <- function(n) {  
2   u1 <- runif(n) # U(0,1)  
3   u2 <- runif(n) # U(0,1)  
4  
5   r <- sqrt(-2 * log(u1))  
6   z1 <- r * cos(2 * pi * u2) # first  
7   z2 <- r * sin(2 * pi * u2) # second  
8  
9   c(z1, z2)  
10 }
```

Figure 10: Box-Muller algorithm

b) Compare empirical density vs true

As for the previous exercises, the comparison is presented in Figure 11.

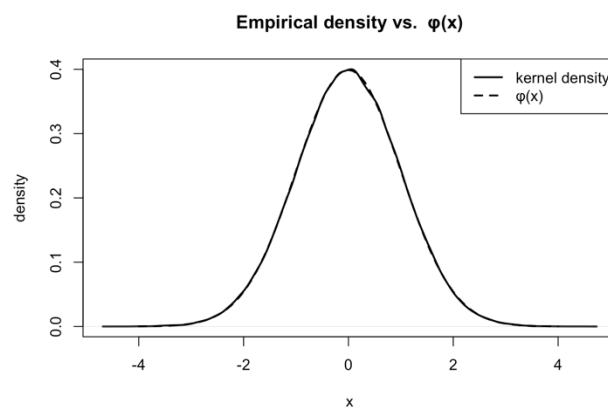


Figure 11: Empirical density vs true for Box-Muller

c) Perform a Kolmogorov-Smirnov test

I performed the Kolmogorov-Smirnov test as suggested in the exercise. Running `ks$statistic` and `ks$p.value` lead to 0.001127206 and 0.9612609 respectively.

Exercise 13

a) Implement the polar algorithm

My code for the Polar algorithm is presented in Figure 12.

```
1 polar_norm <- function(n, seed = NULL) {  
2   if (!is.null(seed)) set.seed(seed)  
3  
4   # 2.n uniforms up front  
5   u1 <- runif(n, -1, 1)  
6   u2 <- runif(n, -1, 1)  
7  
8   s <- u1^2 + u2^2  
9   ok <- which(s > 0 & s < 1) # acceptance condition --> circle  
10  
11  # Transform accepted pairs  
12  factor <- sqrt(-2 * log(s[ok]) / s[ok])  
13  z1 <- u1[ok] * factor  
14  z2 <- u2[ok] * factor  
15  Z <- c(z1, z2)  
16  
17  attr(Z, "pairs_used") <- n  
18  attr(Z, "pairs_accepted") <- length(ok)  
19  
20  Z  
21 }
```

Figure 12: Polar algorithm

b) Compare empirical density vs true

As for the previous exercises, the comparison is presented in Figure 13.

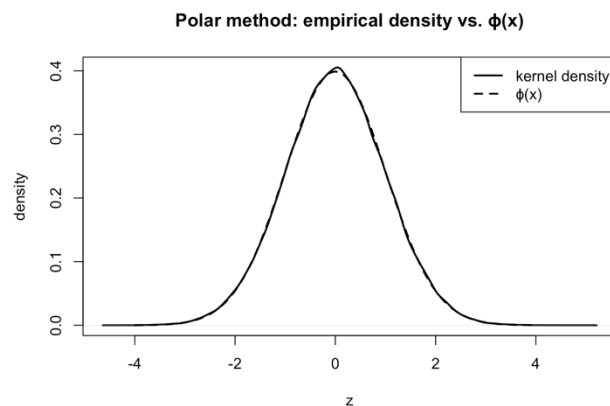


Figure 13: Empirical density vs true for Polar algorithm

c) Perform a Kolmogorov-Smirnov test

I performed the Kolmogorov-Smirnov test as suggested in the exercise. Running `ks$statistic` and `ks$p.value` lead to 0.002247065 and 0.404904 respectively.

d) Calculate the number of generated realizations.

I run the code considering the exercise parameters obtaining 78660 pairs accepted with 157320 normals obtained. The empirical acceptance probability then leads to 0.7866, very close to the theoretical which is 0.7854.