University of Cape Town
Department of Statistical Sciences
Honours: R & Statistical Computing Assignment 1

Instructions

- The following assignment is the first of two hand-ins for me.
- Save both the Rmd and the html/pdf version of your output as 'StudentNumber.Rmd' and 'StudentNumber.nb.html'/'StudentNumber.pdf'. Create an R object in your workfile named 'stdnumbers' to store your student number.
- Submit your assignment on Vula under the Assignments tab by 9 am on 17 March 2024. Late hand-ins will be penalised 10% per day or part there of a day!
- Hand in both an R markdown file and a compiled pdf version of your R markdown file.
- In this assignment you SHOULD include your code in the main part of your submission. You need to describe the algorithms/code used and briefly explain the logic of your code where needed. You do not have to explain what each and every line of your code does, only the logic where appropriate should be explained. Code without explanations will receive a low mark.
- Below you are explicitly told to create certain objects with specific names. Do so! The names of some of the required objects are shown using single quotes in brackets after the question.
- Ensure that you add comments to your code to explain the logic of your code where necessary!
- **At the end of the assignment ensure that your R code compiles!!!**
- Where needed, interpret your results.
- Where needed, provide mathematical derivations although keep this to a minimum. Reference all works used.
- Attach a SIGNED plagiarism declaration.
- Reference your work where needed! Marks will be deducted if you do not reference work appropriately.
- **Do not include any `rm` or any `setwd` commands in your Rmd file!**

Question 1

Suppose that the probability density function of a random variable X is

$$f(x) = a \times xe^{-4x} \quad \text{for } 0 < x < \infty.$$

When answering the questions below, ensure that you create an `R function` named 'fx' to calculate the value of $f(x)$.

(a) Use the 'integrate' function to show that $a = 16$. ('a') If needed read the help file.

```
?integrate
```

(b) Use numerical integration to calculate var(X). ('varX')

(c) Evaluate $f(x)$ at the following x values: x=0.5, x=1 and x=2 and store the object as a vector named 'xeval'.

(d) Plot the function over the range (use $x \in [0, 2]$) of x.

(e) What value of x maximises $f(x)$? ('xopt')

(f) Refer to insert (a) of Figure 1. Obtain the roots of $g(x) = f(x) - 0.55$. i.e. where does $g(x)$ cross the x-axis? This can easily be obtained using the `uniroot` function. I however want you to use optimisation to obtain the solution. Refer to insert (b) of Figure 1 and note that the roots of $g(x)$ are the points that minimises $h(x) = |g(x)|$.

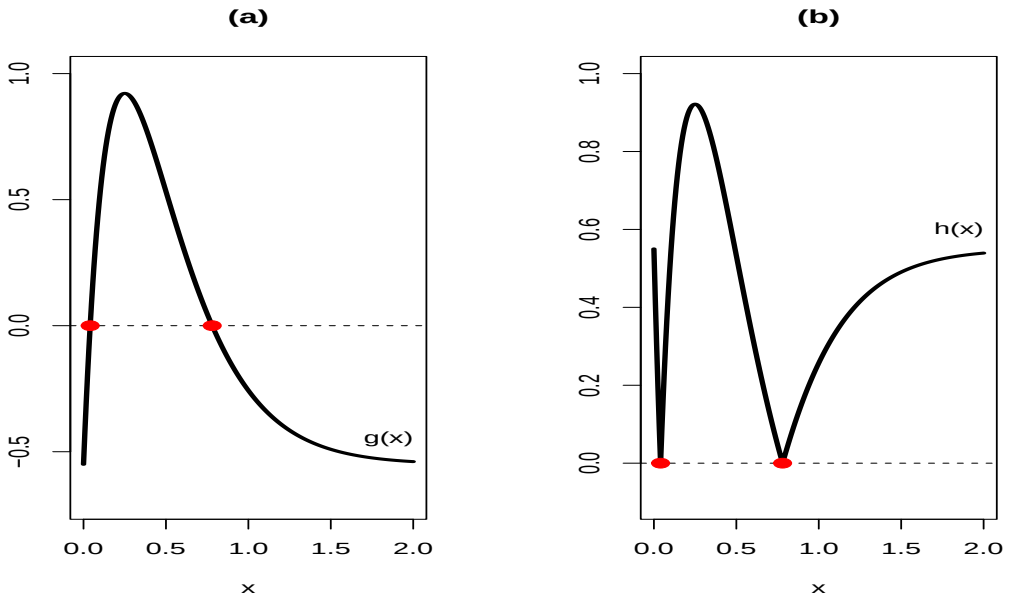

Figure 1: (a) $g(x)$ and (b) $h(x) = |g(x)|$. The roots of $g(x)$ are highlighted in red.

(g) In the following question we use the following steps to obtain samples from $f(x)$:

1. Set a range for x. In this example set 'xlo=0' and 'xhi=5'. i.e. $x \in [0, 5]$. Here we are assuming the $\mathbb{P}(X > 5) \approx 0$.

2. Generate a uniform random number between 'xlo' and 'xhi'. Denote the draw as x_0.
3. Calculate the value of $f(x_0)$.
4. Generate a uniform random number between 0 and $f(x_0)$. Denote the draw as y.
5. Now identify the x-coordinates of where the line $y = y$ intersects with $f(x)$ as per the figure below. Name the x-coordinates as 'xl' and 'xr'.

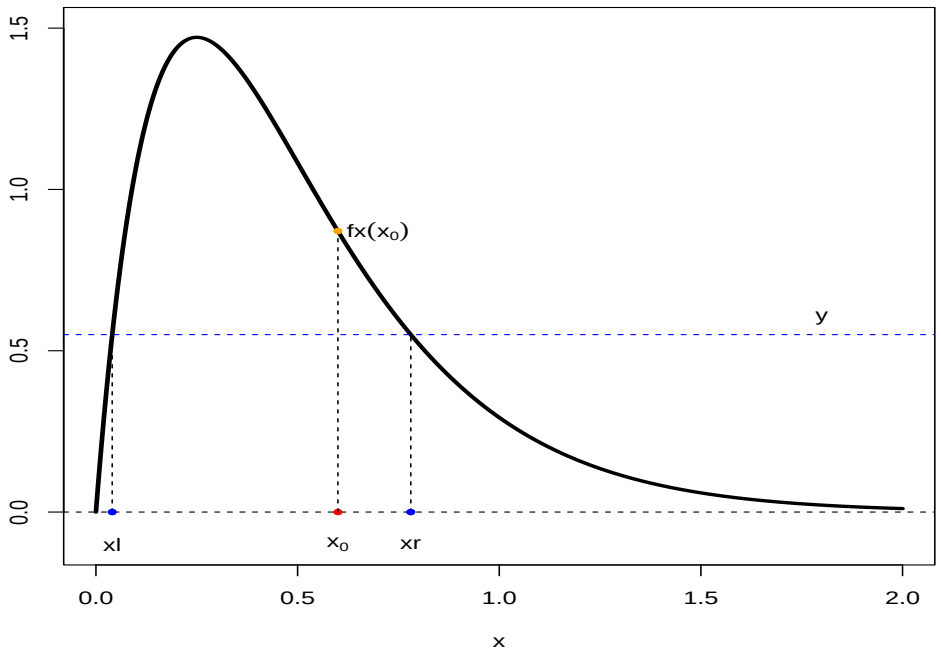

Figure 2: $f(x)$ represents the probability distribution function of X. x_0 is a random draw from the range of X, y is a uniform draw from the range $[0, f(x_0)]$ while 'xl' and 'xr' are the x-coordinates of where the line $y = y$ intersects with $f(x)$.

6. Now obtain a uniform draw between 'xl' and 'xr'. Name this draw x_{new}.
7. Store x_{new} and repeat steps 3 - 6 (many times, say 50 000 times) using the value of x_{new}.

Write a function to do steps 3-6 given a value of x_0. Your function should have the following format:

```
sampler <- function(x0, xopt, xlo, xhi){
 #x0 is some random x value in the allowable range  of X
 #xopt is the x location of the maxima of fx
 #xlo is set to 0
 #xhi is set to 5

 #put instructions here

 xnew #return xnew
}
```

The above algorithm produces a Markov chain. i.e. the stored x_{new} values. Discard the first half of the samples and store the resulting vector as an object named 'samples'.

Use the following code as a guide.

```
#step 1: sample a random starting point in the domain of x
x0 <-

#store the generated x values in 'samples'
samples <- NULL

#the first x_new value
samples[1] <-

#the number of iterations of the algorithm
nsims <- 50000

for (i in 2:nsims){
 samples[i] <-
}

#retain the second half of samples
samples <- samples[-c(...)]
}
```

(h) Plot the histogram of your retained samples. Superimpose $f(x)$ on your plot. Calculate the variance of X using your samples and compare this value to the one obtained when using numerical integration. How do these values compare to the analytical value?

(i*) An estimator for the mean of X can be calculated as

$$\hat{\mu} = \frac{1}{N}\sum_{t=1}^{N} s_t$$

where s_t is the value of the t^{th} retained sample value and $N = 25000$. Suggest an estimate for the variance of $\hat{\mu}$. Here you might have to do some maths. Think of $\{s_t\}$ as a time series!

Question 2

Let

$$f(x,y) = (3x-5)^2 + y^2 + \frac{y^3 - x^3}{1000}.$$

Write your own **gradient-descent algorithm** to obtain the value of x and y that minimises $f(x,y)$. Store the result in an R object named 'opt_own'. Set the starting value of your algorithm to $(x^{(0)}, y^{(0)})^T = (15,\ 20)^T$ and let $\alpha = 0.01$.

Hints:

1. Code a gradient function that returns $\left(\frac{df}{dx}, \frac{df}{dy}\right)$ as a vector.

2. Recall that $\dfrac{d(x^n)}{dx} = nx^{n-1}$ if n is an integer. We thus have

$$\frac{df}{dx} = 18x - 30 - \frac{3x^2}{1000}$$
$$\frac{df}{dy} = 2y + \frac{3y^2}{1000}.$$

3. Code the gradient descent iterative step. Define $\boldsymbol{Z}_i$ as the location of the parameter vector at the i^{th} iteration. The parameter vector is updated as,

$$\boldsymbol{Z}_{i+1} = \boldsymbol{Z}_i - \alpha F(\boldsymbol{Z}_i)$$

   where $\alpha = 0.01$ and $F(\boldsymbol{Z}_i)$ are the gradients at point $\boldsymbol{Z}_i$.

Compare your solution to the one obtained using R. Provide a figure or two explaining the above iterative algorithm and how the solution is obtained. You might have to consult some of the optimisation notes on Vula.