Zool 710: HW6: Numerical Integration, Derivatives, and Roots

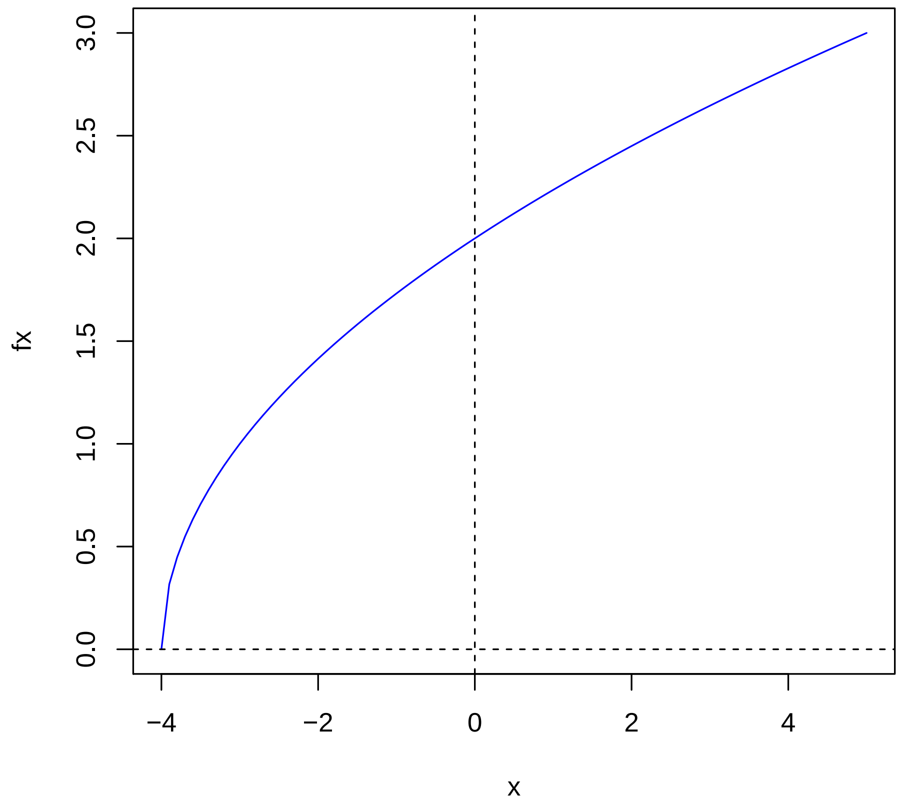
Marguerite A. Butler

DUE: Saturday, March 3th by 5pm, please turn in to GitHub

This week you will write your own numerical methods functions for differentiation and integration of arbitrary functions and use them to try different methods for finding roots and area under the curve. These are "bread and butter" tools used for many many data analysis applications. This is because although exact solutions to some problems are difficult, it is often easy to find a numerical approximation using an iterative algorithm like what you will code below. Ideally, you want an algorithm that always converges (or at least gets very close to) the solution, even if you start off with a dumb guess.

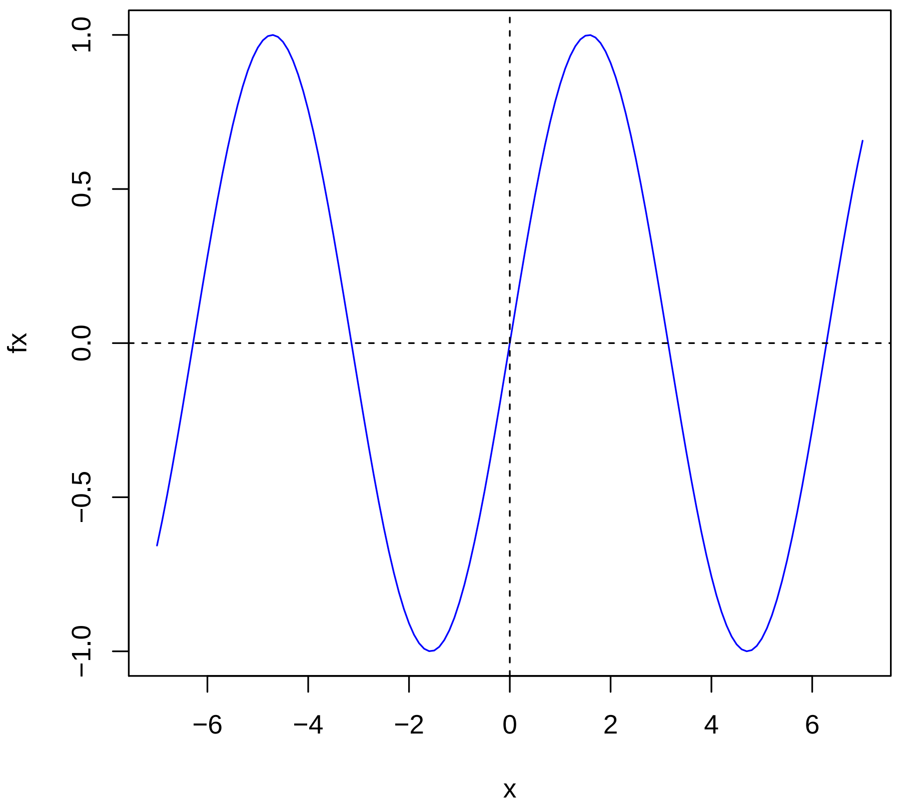
For all exercises where you are working on an equation or a function, first generate x and $f(x)$ (otherwise know as the y values) as inputs (examples given below). Design your functions to work on the x and $f(x)$ objects. So for example, if you wanted to use $f(x) = \sqrt{x+4}$ over the domain $-4 <= x <= 5$ you could generate the input this way:

```
> x <- seq( from=-4, to=5, by=0.1)
> fx <- sqrt( x+4 )    ## for generating input it is OK to use the built-in functions
> plot(x,fx, type="l", col="blue")
> abline(h=0, v=0, lty=2)
```



Or a sine wave over the domain -7 to 7, etc.:

```
> x <- seq( from=-7, to=7, by=0.1)
> fx <- sin(x)
> plot(x,fx, type="l", col="blue")
> abline(h=0, v=0, lty=2)
```



1. Roots

   (a) Write function that can find the square root of an arbitrary real number using Mechanic‘s rule (algorithm from Thursday‘s class ”Homemade Pickles, Preserves, and Square Roots”). Allow the user to supply an initial guess and tolerance if they wish. Test it on at least 10 numbers.

   (b) Write a script that performs Newton‘s method for solving nonlinear equations following the guidance given in Problem 7-1 from the FORTRAN Coloring Book, but of course write the code in R, not FORTRAN, lol. The instructions include testing on certain functions as well as comparing results between Mechanic‘s rule and Simpson‘s rule so you can include your Mechanic‘s rule function in this same script.

2. Area under the curve

   (a) Write a function to calculate the derivative of an arbitrary input function $F(x)$ at an arbitrary x, in an arbitrary neighborhood around the point of interest. The easiest method is to use simple difference formulas as described here:

http://oregonstate.edu/instruct/ch490/lessons/lesson11.htm. You can use whichever you like, but the central difference formula is often the best.

(b) Write a function that performs numerical integration using the trapezoidal rule, with arbitrary limits of integration and at arbitrary interval widths. You can use follow the guidance in Problem 8-1 from the FORTRAN Coloring Book.

(c) Write another integration function but using Simpson‘s rule (Problem 8-1).

(d) Yup, you guessed it - complete the full assignment in 8-1 which involves comparing the two methods for find the area under the curve at different intervals and for different functions.

Again, follow the logic but translate to R. You may work with your partner on these components, but be sure to state your partner‘s name in the header of your scripts in the comments. Feel free to ask if any questions come up after discussing with your partner.

Turn in: A script or set of scripts. For demonstration that your code works, please make sure your script generates a pdf plot of your input data (where this makes sense), and to a text file write the input function you used, the input data, and the output.

Grading: The most important factor is that your programs work, and that you know how to use appropriate R programming elements. Again, use the building blocks that we learned in class, do NOT search for a ready-made solutions on the internet. Other factors include “elegance” and neatness of your code; basically the code should be readable and commented, and should not contain a bunch of extra parameters and functions that are not used in the main program. Keep it simple!