Final Project
STAT 280 B: Statistical Programming
Winter 2024

Due date and time: Monday, April 15 at 23:59 EST

Instructions

- Students should upload their solutions on Moodle before due date. **Late submissions will not be accepted.**
- Solutions should be uploaded as **a single PDF**.
- Make sure to clearly state the problem number in your solutions. Always show the code used to solve an exercise (not the output only, unless explicitly stated) and comment on your code by using # where approperiate.
- Readability of code and clarity of presentation will be taken into account when marking.
- **Important note about graphics:** When Producing figures, use approperiate additional features such as titles, axis labales, legends, suitable markers, colors, etc. to make your plots clearly understandable. Clarity of visualization will be taken into account during marking.

Problem 1 - Olympic games [20 marks]

Consider the data set in the file `speed.txt`, available on Moodle as an attachment to the Final Project. It contains the times in seconds recorded by the winners in the finals of various men's running events (200, 400, 800 and 1500 metres) at each of the 21 Olympic Games from 1900 to 2000, along with the heights above sea level of the different venues.

1. Read the data and store it in a data frame called `speed.data`. Print the first 5 rows of `speed.data`.
2. Calculate the average speed (in meters per second) of the winner of each race, and add this data as a new column of the data frame `speed.data`. Name this new column `Speed`. Print the first 5 rows of the modified data frame.
3. Sort the data by increasing value of year. Print the first 10 rows of the sorted data frame.
4. Create a new data frame called `speed.year` containing two columns: the year (called `Year`) and the average speed among all winners of that year (called `Speed`). Plot the speed as a function of the year for the new data frame `speed.year`. What do you obeserve?
5. Using the command `lm()`, find the line that best fits the data visualized in part 4. Create a plot that shows botoh the data and the best fitting line.
6. Assume the best fitting line computed in part 5 has the form $y = mx + q$. What are m and q?
7. According to the average speed predicted by the best fitting linear model, in what year will the 100 meters race be likely to be run in less than 7 seconds?
8. For each year, compute the residuals, i.e. the differences between the actual average speed and the speed predicted by the best fitting line. Plot a histogram of the residuals.
9. How close are the residuals to being normally distributed? Support your answer with an appropriate visualization strategy.
10. Compute m and q using the command `qr.solve()` instead of `lm()`.
11. Considering the data frame `speed.data`, investigate if there is any relation between the altitude of the venue and the average speed. Support your answer by appropriate statistical considerations and visualizations.

Problem 2 - Modified Newton's method [20 marks]

The objective of this problem is to implement an efficient root finding technique for polynomials. In this problem, a polynomial with coefficients $c_1, \ldots, c_{n+1}$ is defined as

$$P(x) = c_{n+1}x^n + c_n x^{n-1} + \cdots + c_2 x + c_1.$$

1. Create a function with header

```
EvalPoly <- function(c, x)
```

that evaluates a polynomial $P(x)$ with coefficients $c_1, \ldots, c_{n+1}$ (stored in the vector `c`) at the point x.

2. Evaluate the polynomial $P(x) = 3.5x^3 - 1.7x + 1$ at $x = 13.4$ using `EvalPoly`.
3. Create a function

```
PolyDerEval <- function(c, x)
```

that evaluates the derivative of a polynomial $P(x)$ with coefficients $c_1, \ldots, c_{n+1}$ at the point x.

4. Test the function `PolyDerEval` by evaluating $P'(x)$, where $P(x)$ and x are as in part 2.

5. Create a function

```
NewtonPoly <- function(c, x0, TOL)
```

that implements Newton's method applied to a polynomial $P(x)$ defined by coefficients in `c`. This function should use the `PolyEval` and `PolyDerEval` to evaluate $P(x)$ and its derivative $P'(x)$. The function should return the vector of computed approximations $x_0, \dots, x_k$ and stop as soon as $|P(x_k)| \leq \text{TOL}$ or $k > 1000$.

6. Use the function `NewtonPoly` to approximate one of the roots of $P(x) = x^3 - 7.1x + 2.3$ with $x_0 = -1$ and $\text{TOL} = 10^{-10}$. Print the sequence of approximations computed by Netwon's method.
7. Now, consider the polynomials $P(x) = x^k$, for $k = 2, \dots, 10$. Apply Newton's method with $x_0 = 1$ and $\text{TOL} = 10^{-12}$ to these polynomials. Plot the number of iterations needed by Newton's method to reach the desired accuracy, as a function of the exponent k.

We now consider modified Newton's method for the approximation of roots of $P(x)$ that have multiplicity greater than 1 (for example, 0 is a root of multiplicity k of x^k). The idea is to apply Newton's method to the function $P(x)/P'(x)$ instead of $P(x)$. This corresponds to an update of the form

$$x_{k+1} = x_k - \frac{P(x_k)P'(x_k)}{[P'(x_k)]^2 - P(x_k)P''(x_k)}$$

Note that this method requires the second derivative of $P(x)$.

8. Implement a function

```
PolyDer2Eval <- function(c, x)
```

that evaluates the second derivative of a polynomial $P(x)$ with coefficients defined by `c`. Test your function on the polynomial $P(x)$ and point x considered in part 2.

9. Create a function

```
ModifiedNewtonPoly <- function(c, x0, TOL)
```

that implements the modified Newton's method. The stopping criterion should be the same as in `NewtonPoly`.

10. Repeat the experiment in part 7 using modified Newton's method. What do you observe?

Problem 3 - Ada's walk [20 marks]

The goal of this problem is to simulate a random walk of an agent called Ada (in honor of Ada Lovelace) over an infinite two-dimensional grid, i.e. the set of all pairs (i, j), where i, j are integers. The random walk is defined as follows:

- The random walk is composed of a sequence of positions $A_0, A_1, A_2, \dots$, where each A_k is a 2D point with integer coordinates.
- At time $t = 0$, Ada is in position $A_0 = (0, 0)$.
- Assume Ada to be in position A_t at time t. At time $t + 1$, she will move up, down, left or right with equal probability. For example, if $A_3 = (3, -1)$, then A_4 can be either $(4, -1)$, $(3, 0)$, $(2, -1)$, or $(3, -2)$ with equal probability.
- Ada always moves ($A_t \neq A_{t+1}$ for all t)
- The random walk stops when Ada is back in position $(0, 0)$ or if she has already done more than 100 steps (in that case, her final position will be A_{100}, which might or might not be $(0, 0)$).

1. Write a function with header

```
AdaWalk <- function()
```

The function should return the trajectory of Ada's random walk. The positions A_k should be stored as columns of a matrix with two rows.

2. Plot 4 examples of random walks generated by your code, choosing an appropriate visualization strategy.
3. Estimate the probability that Ada comes back to the orgin $(0,0)$ in at most 100 steps. Use a Monte Carlo simulation with at least 100 repeated experiments (or more, if your computer can)
4. Estimate the average number of steps needed by Ada to return at the origin, conditional to the event that she is able to do so in at most 100 steps. Use again at least 100 repetitions of the random walk.

Problem 4 - Gradient descent [20 marks]

Let A be an $n \times n$ matrix, b an n-dimensional vector and c a scalar. We would like to find an n-dimensional vector $x_{\min}$ that minimizes the following function:

$$f(x) = \frac{1}{2} x^T A x + x^T b + c.$$

(Note that the constant c has no impact on the minimizer $x_{\min}$, but only on the minimal value $f(x_{\min})$. Also, note that this function could have multiple minimizers). An algorithm used for this purpose is the so-called **gradient descent**. Using multivariate calculus, it is possible to show that the gradient of $f(x)$ is given by the formula

$$\nabla f(x) = Ax + b.$$

The idea of the gradient descent algorithm is the following. Assume to have an initial approximation x_0 of $x_{\min}$. Let $h > 0$ be a certain step size (usually, a very small positive number). Define the gradient descent iterations as

$$x_{k+1} = x_k - h \nabla f(x_k).$$

We keep computing new iterations x_k until a certain stopping criterion is met. For example, given a certain tolerance $\texttt{TOL} > 0$, we can stop when

$$\|x_{k+1} - x_k\|_2 \leq \texttt{TOL},$$

where $\|x\|_2$ is the Euclidean norm of x.

1. Implement the gradient descent algorithm in a function with header

```
GradientDescent <- function(A, b, h, x0, TOL, N.max)
```

The function should return all the iterations x_k produced by the gradient descent method until the stopping critetion given above is met or if the maximum number of iterations `N.max` has been reached.

2. Test your function with

$$A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \quad b = \begin{bmatrix} 5 \\ 6 \end{bmatrix}$$

and with $x_0 = (0,0)$, $\texttt{TOL} = 10^{-7}$, $h = 0.1$, and $\texttt{N.max} = 100$.

3. Knowing that the true solution to the problem in part 3 is $x_{\min} = (-4/3, -7/3)$, create a convergence plot for the gradient descent method applied in part 2, by showing the decay of $\|x_k - x_{\min}\|_2$ as a function of the iteration k. Use a logarithmic scale for the y-axis.
4. Define the matrix M_n to be of dimension $n \times n$ having 2's on the main diagonal, -1's on the first upper and lower diagonal, and zeros elsewhwere. For example,

$$M_4 = \begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 2 \end{bmatrix}.$$

Moreover, define v_n to be an n-dimensional vector of ones. Compute M_{10} and v_{10} and store them in `M` and `v`. (If you can, use vectorized code).

5. Use the gradient descent algorithm with appropriate input parameters to find a minimizer $x_{\min}$ and the minimum value $f(x_{\min})$ for the function

$$f(x) = \frac{1}{2}x^T M_{10} x + v_{10}^T x + 1.$$

Problem 5: Monte Carlo Simulation [20 Points]

Part I [5 Points]

i. To estimate $I = \int_{-2}^{2} e^{x^2+x}dx$, generate 1000 random numbers and use the substitution $y = \frac{x-(-2)}{2-(-2)}$. Note that y varies in the interval $(0, 1)$.

ii. Now generate 1000 random samples from $X \sim U(-2, 2)$ to estimate $I = \int_{-2}^{2} e^{x^2+x}dx$.

iii. Furthermore, create an R function `fun()` that implements the mathematical function

$$x \to f(x) = e^{x^2+x}.$$

Considering `fun()` and applying the syntax `integrate` in R, calculate the definite integral; $I = \int_{-2}^{2} e^{x^2+x}dx$ and compare the result with those obtained in parts a. and b. Comment on your comparisons.

Part II [15 Points]

i. Create an R function `fun()` that implements the mathematical function

$$x \to f(x) = cos^3(ex) + log_3(5x) - arctan(x)$$

ii. Using `seq`, create a numeric vector called `grid` containing `N+1` equispaced points between 1 and 2 (inclusive) where `N` = 10^6. (Do not print the result).

iii. Create a vector `m = (10, 100, 1000, 10000)` and 4 vectors `subgrid.1`, `subgrid.2`, `subgrid.3` and `subgrid.4` defined as follows. For every `i`, `subgrid.i` contains `mi` points randomly chosen from `grid` (without repitItions) using the built-in function `sample()`.

iv. Create vectors `eval.1`, `eval.2`, `eval.3` and `eval.4` containing the evaluations of `f` at points in `subgrid.1`, `subgrid.2`, `subgrid.3` and `subgrid.4`. Determine averages of `eval.1`, `eval2.`, `eval.3` and `eval.4` in a four dimensional vector space called `monte.carlo`.

v. Assume that the exact value of the integral of the above function is

$$I = \int_1^2 f(x)dx = \int_1^2 [cos^3(ex) + log_3(5x) - arctan(x)]dx = 0.479199$$

.

Compare it with the entries of the vector `monte.carlo`. What do you observe?

Problem 6: Simulation [20 Points]

Part I [10 Points]

An eyeglass shop has n eyeglasses to sell and makes \$1.00 on each sale. Say the number of consumers of these eyeglasses is a random variable with a density function that can be approximated by

$$f(x) = 1/200, \quad 0 < x < 200,$$

a pdf of the continuous type. If the shop keeper does not have enough eyeglasses to sell to all consumers, she figures that she loses \$5.00 in goodwill from each unhappy customer. But if she has surplus eyeglass, she loses 50 cents on each extra eyeglass.

i. What should n, the number of eyglasses, should be to maximize profit?
(Hint: Note that the expected profit is:

$$E(Profit) = \int_0^n (x - \frac{1}{2}(n-x))\frac{1}{200}dx + \int_n^{200} (n - 5(x-n))\frac{1}{200}dx)$$

ii. Simulate 100 sales of the shop.

Part II [10 Points]

For uniform $(0,1)$ random variables $U_1, U_2, U_3, \ldots$ define

$$N = minimum\{n : \Sigma_{i=1}^{n} U_i > 1\}$$

That is, N is equal to the number of random numbers that must be summed to exceed 1.

i. Estimate $E(N)$ by generating 100 values of N.

ii. Calculate the exact value of $E(N)$.