Statistic Programming

STATS 202A: Homework #3

Zachary Lacey

November 11, 2021

Problem 1

Problem 1 Prompt

Approximation of an infinite series in C.

It is well known that the series:

$$1 - \frac{1}{2} + \frac{1}{3} - \frac{1}{4} + \frac{1}{5} - \frac{1}{6} \pm \cdots = ln(2) \tag{1}$$

Write a C function called $alt2(n)$ that computes the first n terms in this series, as a function of n. Call your C function from R to evaluation $alt2(n)$ for various n. Using R, plot $alt2(n)$ vs. n, for n ranging from some small number up to 1 million. You may set up your range of the y-axis in a way that you feel is appropriate. You don't need to show $alt2(n)$ for all values of n and should not plot $alt2$ for very small values of n if they are off the plot.

Problem 1 Output

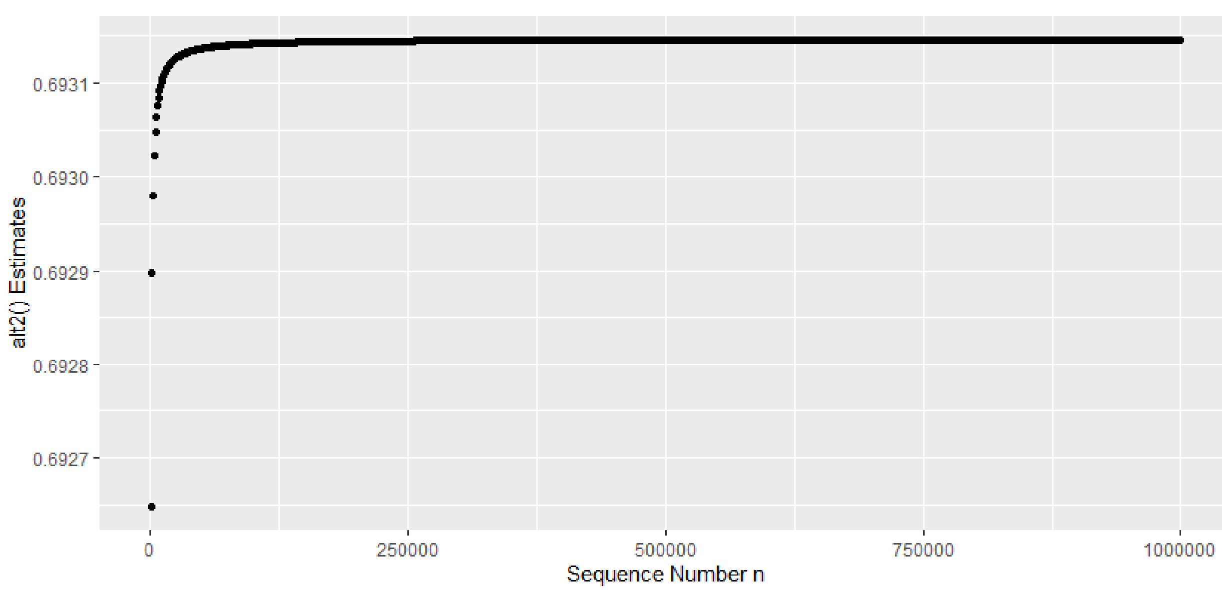

Figure 1: Estimated Values of $ln(2)$, Using an Infinite Series.

Figure 1 represents the plot of $alt2(n)$ versus n, for $n = 1000, 2000, \ldots, 1000000$. This decreases the computational cost of plotting (as opposed to plotting all 1 million values).

Problem 2

Problem 2 Prompt

Kernel density estimation in C and plotted in R.

Write a C function to compute a Gaussian kernel density estimate for uni-variate data. The inputs to the function should be two integers, m and n, a vector g of m gridpoints at which to calculate the estimates, a vector x consisting of the n observed data points, and a vector y of length m which will contain the resulting density estimates.

(A) Gather data on all earthquakes of magnitude at least 3.0 in the longitude range -122.0 to -118.0 and latitude range 34.0 to 38.0, from Jan 1, 1960 to Oct 1, 2021, from https://service.scedc.caltech.edu/eq-catalogs/date_mag_loc.php. Input the data into R. (Use minimum magnitude = 3.0, maximum magnitude = 9.0, min depth = -5km, max depth = 100km, event type = earthquake, geographic type = local). Take this vector of earthquake magnitudes, and use your C function to make a kernel density estimate of the earthquake magnitudes, using a Gaussian kernel with bandwidth selected using the rule of thumb suggested by Scott (1992). You may calculate this bandwidth in R. Let $m_1, m_2, \ldots, m_{100}$ = a vector of 100 equally spaced magnitudes spanning the observed range of magnitudes in your dataset, and plot your kernel density estimate $\hat{f}(m_1), \hat{f}(m_1), \ldots, \hat{f}(m_{100})$.

Problem 2 Output

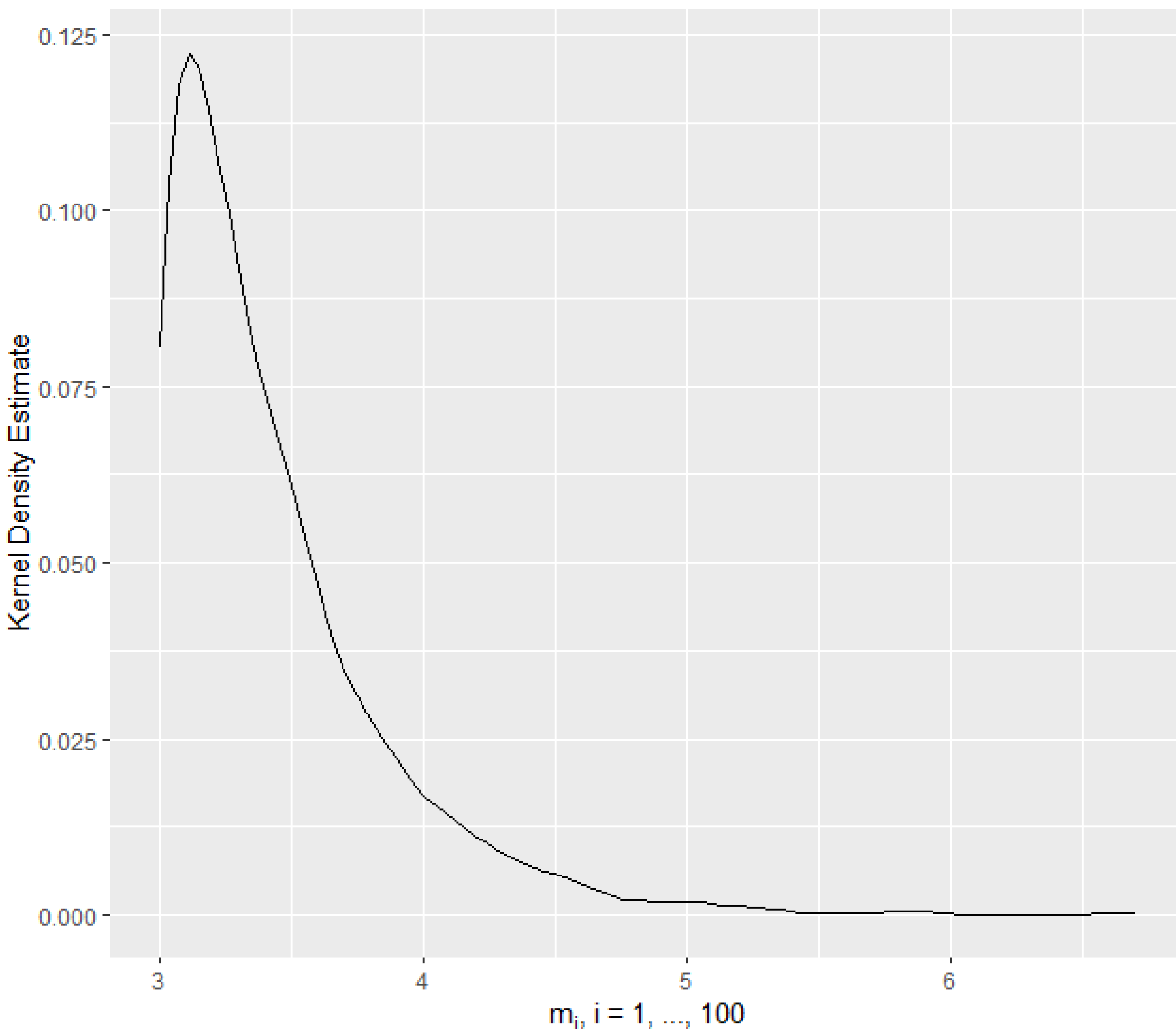

Figure 2: Kernel Density Estimates.

Figure 2 represents the kernel density estimates of the earthquake magnitudes from the gathered Los Angeles County earthquake dataset.

Code

C++ Code

alt2.cpp

```cpp
#include <Rcpp.h>
#include <math.h>
using namespace Rcpp;

// [[Rcpp::export]]
/* Function: alt2
 * Input:
 *    n: Number of iterations in the infinite series to estimate ln(n)
 *
 * Output:
 *    alt2Estimates: Vector that contains the first 'n' alt2 estimates of
 *                   ln(n)
 *
 */
DoubleVector alt2(int n) {

  DoubleVector alt2Estimates(n);
  double series_sum = 0.0;

  for (int k = 1; k <= n; k++) {
    series_sum += (pow(-1.0, k + 1)) / k;
    alt2Estimates[k - 1] = series_sum;
  }

  return alt2Estimates;
}
```

Listing 1: alt2.cpp

gaussianKernelDensity.cpp

```cpp
#include <Rcpp.h>
#include <Rmath.h>
using namespace Rcpp;

// [[Rcpp::export]]
/* Function: gaussianKernelDensity
 * Input:
 *    m: Number of elements in vector m
 *    n: Number of observations in earthquake magnitude data
 *    gridpoints: vector m, vector of values where we compute the
 *                kernel estimate.
 *    x: earthquake magnitude data
 *    y: initialized Gaussian kernel density estimate results vector.
 *    bw: bandwidth
 *
 * Output:
 *    y: Gaussian kernel density estimate results vector.
 *
 */
NumericVector gaussianKernelDensity(
    int m, int n, NumericVector gridpoints,
    NumericVector x, NumericVector y, double bw) {

  for (int i = 0; i <= m; i++) {
    y[i] = sum(dnorm((x - gridpoints[i]) / bw, 0, 1, 0)) / n;
  }

  return y;
}
```

Listing 2: gaussianKernelDensity.cpp

R Code

HW3_Question1.R

```
## Homework 3 Question 1
##
## Author: Zachary G. Lacey
## Email: zlacey@g.ucla.edu
## Date: October 27th, 2021
##
## Course: STAT 202A - Statistics Programming
## Assignment: Homework 3
## Due Date: November 11th, 2021
##
## Question 1
##

library(Rcpp)
library(tidyverse)

## 1. Approximation of an infinite series in C.
##
##    It is well known that
##
##    1 - 1/2 + 1/3 - 1/4 + 1/5 - 1/6 +/- ... = ln(2).
##
##    Write a C function called alt2(n) that computes the first 'n' terms
##    in this series, as a function of 'n'. Call your C function from R
##    to evaluate alt2(n) for various 'n'. Using R, plot alt2(n) vs. 'n',
##    for 'n' ranging from some small number up to 1 million. You may set
##    up your range of the y-axis in a way that you feel is appropriate.
##    You do not need to show alt2(n) for all values of n and should not
##    plot alt2 for very small values of 'n' if they are off the plot.

```

```r
# Source the C function
sourceCpp("alt2.cpp")

# Number of Iterations in the Series
nIterations = 1000000

# Initialize a vector to store the resulting alt2 estimates in.
alt2Estimates = vector("numeric", nIterations)

# Determine the first 1,000,000 alt2 estimates of ln(n)
alt2Estimates = alt2(1000000)

# Sequence from [(1,000), (1,000,000)] with a 1,000 increment spacing.
# This is used to summarize the results in a less computationally
# expensive way (as opposed to including all 1,000,000 values)
nSeq = seq(1000, nIterations, 1000)

# Summarized alt2 estimate results
alt2EstimatesSeq = alt2Estimates[nSeq]

# Place the summarized alt2 estimate results into a Tibble
alt2EstimatesTibble = tibble(n = nSeq,
                             alt2Estimates = alt2EstimatesSeq)

# Plot the results
alt2EstimatesTibble %>% ggplot() +
  geom_point(aes(x = n, y = alt2Estimates)) +
  labs(x = "Sequence Number n",
       y = "alt2() Estimates")

print("Finished")
```

Listing 3: HW3_Question1.R

HW3_Question2.R

```r
## Homework 3 Question 2
##
## Author: Zachary G. Lacey
## Email: zlacey@g.ucla.edu
## Date: October 27th, 2021
##
## Course: STAT 202A - Statistics Programming
## Assignment: Homework 3
## Due Date: November 11th, 2021
##
## Question 2
##

library(Rcpp)
library(tidyverse)

## 2. Kernel density estimation in C and plotted in R.
##
##    Write a C function to compute a Gaussian kernel density estimate for
##    uni-variate data. The inputs to the function should be two integers, $m$
##    and $n$, a vector $g$ of $m$ gridpoints at which to calculate the
##    estimates, a vector $x$ consisting of the $n$ observed data points, and
##    a vector $y$ of length $m$ which will contain the resulting density
##    estimates.
##
##    Gather data on all earthquakes of magnitude at least 3.0 in the longitude
##    range -122.0 to -118.0 and latitude range 34.0 to 38.0, from Jan 1, 1960
```

```r
##    to Oct 1, 2021, from
##
##    https://service.scedc.caltech.edu/eq-catalogs/date\_mag\_loc.php.
##
##    Input the data into R. (Use minimum magnitude = 3.0, maximum magnitude
##    = 9.0, min depth = -5km, max depth = 100km, event type = earthquake,
##    geographic type = local). Take this vector of earthquake magnitudes,
##    and use your C function to make a kernel density estimate of the
##    earthquake magnitudes, using a Gaussian kernel with bandwidth selected
##    using the rule of thumb suggested by Scott (1992). You may calculate
##    this bandwidth in R. Let $m_{1}, m_{2}, \dots, m_{100}$ = a vector of
##    100 equally spaced magnitudes spanning the observed range of magnitudes
##    in your dataset, and plot your kernel density estimate $\hat{f}(m_{1}),
##    \hat{f}(m_{1}), \dots, \hat{f}(m_{100})$.

# Loading the Earthquake Data
earthquakeData = read.table("SearchResults.txt")

# Extracting the Earthquake Magnitude Data
earthquakeMag = as.numeric(as.vector(earthquakeData[, 5]))

# Bandwidth of the Earthquake Magnitude data using Scott's Rule of Thumb
b2 = bw.nrd(earthquakeMag)

# Range of the Earthquake Magnitude data
earthquakeMagRange = range(earthquakeMag)

# Vector 'm': vector of equally spaced earthquake magnitudes spanning the
```

```r
# range of the Earthquake Magnitude data (vector of length 100).
m = seq(earthquakeMagRange[1], earthquakeMagRange[2], length = 100)

# Sourcing the C function
sourceCpp("gaussianKernelDensity.cpp")

# Number of elements in vector m
numVectorM = length(m)

# Number of observations in the data (earthquake magnitude data)
numEarthquakeMag = length(earthquakeMag)

# Kernel density estimates results vector
kernelDensityEstimate = vector("numeric", numVectorM)

# Estimation of the Gaussian kernel density
kernelDensityEstimate =
  gaussianKernelDensity(numVectorM, numEarthquakeMag,
                        m, earthquakeMag, kernelDensityEstimate, b2)

# Place the resulting Gaussian kernel density estimates into a Tibble.
kernelDensityEstimateTibble = tibble(m = m,
                                     kernelDensityEstimate =
                                       kernelDensityEstimate)

# Plot the results
kernelDensityEstimateTibble %>% ggplot() +
  geom_line(aes(x = m, y = kernelDensityEstimate)) +
  labs(x = expression(m[i]*", i = 1, ..., 100"),
       y = "Kernel Density Estimate")

print("Finished")
```

Listing 4: HW3_Question2.R