|  |
| --- |
| //t test  ##include <stdio.h>  #include <math.h>  int main()  {  int i, n;  float x[20], sum = 0, sum1 = 0, xbar, ttab, tcal, mu, sd, se;  printf("Enter the population mean: ");  scanf("%f", &mu);  printf("Enter the sample size : ");  scanf("%d", &n);  // Check if sample size is within valid range  if (n <= 0 || n > 20) {  printf("Sample size must be between 1 and 20.\n");  return 1;  }  printf("Enter the sample values: ");  for (i = 0; i < n; i++) {  scanf("%f", &x[i]);  sum = sum + x[i];  }  xbar = sum / n;  printf("Sample mean (xbar): %f\n", xbar);  for (i = 0; i < n; i++) {  sum1 = sum1 + (x[i] - xbar) \* (x[i] - xbar);  }  sd = sqrt(sum1 / (n - 1)); // Sample standard deviation  se = sd / sqrt(n); // Standard error  tcal = fabs((xbar - mu) / se); // Correct formula for t-statistic  printf("Enter the t-table value: ");  scanf("%f", &ttab);  printf("Results: \n");  printf("tcal = %f\n", tcal);  printf("ttab = %f\n", ttab);  printf("Degrees of freedom (df) = %d\n", n - 1);  printf("Level of significance (los) = 0.05\n");  if (tcal < ttab) {  printf("\nAccept Null hypothesis H0, The given samples are drawn from the assumed population with mean %f\n", mu);  } else {  printf("\nReject the Null hypothesis H0, The given samples are not drawn from the assumed population with mean value %f\n", mu);  }  return 0;  } |
| //regression equation  #include <stdio.h>  #include <stdlib.h>  int main() {  int n;  printf("Enter number of x values: ");  scanf("%d", &n);  double x\_values[n];  double y\_values[n];  printf("Enter values for sample x: \n");  for(int i = 0; i < n; i++){  printf("x[%d]: ", i);  scanf("%lf", &x\_values[i]);  }  printf("Enter values for sample y: \n");  for(int i = 0; i < n; i++){  printf("y[%d]: ", i);  scanf("%lf", &y\_values[i]);  }  double sumx2 = 0, sumx = 0, sumy = 0, sumxy = 0;  for(int i = 0; i < n; i++){  sumx += x\_values[i];  sumx2 += x\_values[i] \* x\_values[i];  sumy += y\_values[i];  sumxy += x\_values[i] \* y\_values[i];  }  double sq\_sumx = sumx \* sumx;  double a = ((n \* sumxy) - (sumx \* sumy)) / ((n \* sumx2) - sq\_sumx);  double b = (sumy - a \* sumx) / n;  double c = 1 / a;  double d = -b / a;  printf("\nEquation 1: y = %8.5fx + %8.5f", a, b);  printf("\nEquation 2: x = %8.5fy + %8.5f", c, d);  return 0;  } |
| //f-test  #include <stdio.h>  #include <math.h>  // Function to calculate mean  double calculate\_mean(double data[], int n) {  double sum = 0.0;  for (int i = 0; i < n; i++) {  sum += data[i];  }  return sum / n;  }  // Function to calculate standard deviation  double calculate\_standard\_deviation(double data[], int n, double mean) {  double sum = 0.0;  for (int i = 0; i < n; i++) {  sum += pow(data[i] - mean, 2);  }  return sqrt(sum / (n - 1));  }  int main() {  int n1, n2;  double ftable;  // Read the number of items in the sample1  printf("Enter the number of items in the sample 1: ");  scanf("%d", &n1);    // Read the number of items in the sample2  printf("Enter the number of items in the sample 2: ");  scanf("%d", &n2);  // Read the items in the sample 1  double data1[n1];  printf("Enter the items in the sample 1:\n");  for (int i = 0; i < n1; i++) {  scanf("%lf", &data1[i]);  }    // Read the items in the sample 1  double data2[n2];  printf("Enter the items in the sample 2:\n");  for (int i = 0; i < n2; i++) {  scanf("%lf", &data2[i]);  }    // Calculate the sample means  double mean1 = calculate\_mean(data1, n1);  double mean2 = calculate\_mean(data2, n2);  // Calculate the sample standard deviations  double sd1 = calculate\_standard\_deviation(data1, n1, mean1);  double sd2 = calculate\_standard\_deviation(data2, n2, mean2);    // Calculate the f-value  double f\_value = sd1 > sd2 ? (sd1/sd2) : (sd2/sd1);  // Display the results  printf("Calculated f-value: %.2f\n", f\_value);  printf("Enter the table value of f: ");  scanf("%lf", &ftable);    printf("\nDegrees of freedom: %d, %d\n", (n1-1), (n2-1));  if(f\_value > ftable){  printf("\nNull hypothesis is rejected");  }else{  printf("\nNull hypothesis is accepted");  }  return 0;  } |
| //chisquare  import java.util.Scanner;  public class Chi\_square {   public static void main(String[] args){   *//Read number of values  //define holders for values, observed frequencies & expected frequencies  //collect observed values  //calculate expected values  //compute chi square  //compare with table value  //print hypothesis* int n;  double exp\_f, chi\_cal = 0, sum\_o = 0;  Scanner myScanner = new Scanner(System.*in*);   System.*out*.println("Enter number of observations: ");  n = myScanner.nextInt();   int[] observed = new int[n];  double[] expected = new double[n];   for(int i = 0; i < n; i++){  System.*out*.printf("Enter observed value observer[%d]: ", i);  observed[i] = myScanner.nextInt();  sum\_o += observed[i];  }  exp\_f = sum\_o / n;   for(int i = 0; i < n; i++){  expected[i] = exp\_f;  }   for(int i = 0; i < n; i++){  chi\_cal += (Math.*pow*((observed[i] - expected[i]),2)) / expected[i];  }   System.*out*.println("Enter the table value of chi-square:");  double chi\_tab = myScanner.nextDouble();   System.*out*.printf("Calculated Value of chi-square: %10.5f%n", chi\_cal);  System.*out*.printf("Table Value of chi-square: %10.5f%n", chi\_tab);   if(chi\_cal < chi\_tab){  System.*out*.println("Calculated value is less than table value, accept null hypothesis...");  }else{  System.*out*.println("Calculated value is greater than table value, reject null hypothesis...");  }  } } |
| //correlation coefficient  import java.util.Scanner;  public class Corr\_coeff {   *//read x, y, x values, y values  //calculate sumx, sumy, sumx\*sumy, sum(x square), (sumx square), sum(x\*y)  //a = ([n \* (sum(x\*y))] - [sumx \* sumy]) / ([n \* sum(x square)] - (sumx square)  //b = (sumy - a\*sumx) / n;* static double corr\_Coeff1(double[] x\_values, double[] y\_values){   double sumx2 = 0, sumx = 0, sumy = 0, sumxy = 0, sumy2 = 0;  int n = x\_values.length;  for(int i = 0; i < n; i++){  sumx += x\_values[i];  sumx2 += x\_values[i] \* x\_values[i];   sumy += y\_values[i];  sumy2 += y\_values[i] \* y\_values[i];   sumxy += x\_values[i] \* y\_values[i];  }   double denom = Math.*sqrt*((n \* sumx2 - sumx\*sumx)) \* Math.*sqrt*((n \* sumy2 - sumy\*sumy));  double corr\_coeff = ((n \* sumxy) - (sumx \* sumy)) / denom;  return corr\_coeff;   }   static double corr\_Coeff2(double[] x\_values, double[] y\_values){   double x\_mean = 0, y\_mean = 0, x\_mean\_sq = 0, y\_mean\_sq = 0, x\_y\_diff = 0;  int n = x\_values.length;  for(int i = 0; i < n; i++){  x\_mean += x\_values[i];  y\_mean += y\_values[i];  }  x\_mean /= n; y\_mean /= n;   for(int i = 0; i < x\_values.length; i++){  x\_y\_diff += ((x\_values[i] - x\_mean) \* (y\_values[i] - y\_mean));  x\_mean\_sq += Math.*pow*((x\_values[i] - x\_mean),2);  y\_mean\_sq += Math.*pow*((y\_values[i] - y\_mean),2);  }   double corr\_coeff = x\_y\_diff / Math.*sqrt*(x\_mean\_sq \* y\_mean\_sq);  return corr\_coeff;  }   public static void main(String[] args) {  int n;  Scanner myScanner = new Scanner(System.*in*);  System.*out*.println("Enter number of x values: ");  n = myScanner.nextInt();   double[] x\_values = new double[n];  double[] y\_values = new double[n];   System.*out*.println("Enter value for x: ");  for (int i = 0; i < n; i++) {  System.*out*.printf("x[%d]: ", i);  x\_values[i] = myScanner.nextDouble();  }   System.*out*.println("Enter value for y: ");  for (int i = 0; i < n; i++) {  System.*out*.printf("y[%d]: ", i);  y\_values[i] = myScanner.nextDouble();  }   double corr\_coeff = *corr\_Coeff1*(x\_values, y\_values);  System.*out*.printf("Correlation Coeff method 1 is %8.5f%n", corr\_coeff);   corr\_coeff = *corr\_Coeff2*(x\_values, y\_values);  System.*out*.printf("Correlation Coeff method 2 is %8.5f%n", corr\_coeff);   if(corr\_coeff > 0){  System.*out*.println("Positive correlation exists...");  } else if (corr\_coeff < 0) {  System.*out*.println("Negative correlation exists...");  }else{  System.*out*.println("No correlation exists...");  }   myScanner.close();  } } |
| //poisson dist  import java.util.Scanner;  public class Poisson1 {   *// Function to calculate the factorial of a number* public static int factorial(int x) {  if (x == 0) {  return 1;  } else {  return x \* *factorial*(x - 1);  }  }   public static void main(String[] args) {  Scanner scanner = new Scanner(System.*in*);  int n;  int[] x = new int[10]; *// Array to store the event values* int[] y = new int[10]; *// Array to store the observed frequencies of the events* double[] y1 = new double[10]; *// Array to store the expected frequencies based on Poisson distribution* double sum = 0, prod\_x\_y = 0;  double mean1, p0, p1, f1, f0;   *// Prompt the user to enter the number of values* System.*out*.print("Enter the number of values: ");  n = scanner.nextInt();   *// Read 'n' values into the array 'x' (event values)* for (int i = 0; i < n; i++) {  System.*out*.print("x[" + i + "]: ");  x[i] = scanner.nextInt();  }   *// Read 'n' observed frequencies into the array 'y' (observed frequencies)* System.*out*.println("Enter the observed frequencies: ");  for (int i = 0; i < n; i++) {  System.*out*.print("y[" + i + "]: ");  y[i] = scanner.nextInt();  }   *// Calculate the sum of the observed frequencies and the product of event values and their frequencies* for (int i = 0; i < n; i++) {  sum += y[i];  prod\_x\_y += x[i] \* y[i];  }   *// Print the sum of observed frequencies and the product of event values and their frequencies* System.*out*.println("sum = " + sum);  System.*out*.println("product = " + prod\_x\_y);   *// Calculate the mean (mean1) of the Poisson distribution* mean1 = prod\_x\_y / sum;  System.*out*.println("mean = " + mean1);   *// Calculate the probability of zero occurrences (p0) and its expected frequency (f0)* p0 = Math.*exp*(-mean1);   *// Calculate the expected frequency of zero occurrences (f0)* f0 = sum \* p0;  System.*out*.println("Expected frequency for 0 events (f0) = " + f0);  f0 = 0;   *// Print expected frequencies for each value based on the Poisson distribution* System.*out*.println("Expected frequency y");  for (int i = 0; i < n; i++) {  *// Calculate the probability of observing 'i' events (p1)* p1 = (p0 \* Math.*pow*(mean1, i)) / *factorial*(i);   *// Calculate the expected frequency for 'i' events (f1)* f1 = sum \* p1;  y1[i] = f1;  f0 += f1;   *// Output the expected frequency* System.*out*.printf("y1[%d]: %.4f%n", i, y1[i]);  }  System.*out*.printf("Total expected frequency: %.4f%n", f0);   *// Close the scanner to prevent resource leaks* scanner.close();  } } |
| //numerical integration  #include <stdio.h>  #include <math.h>  double f(double x) {  return log(x);  }  // Trapezoidal Rule  double trapezoidal(double a, double b, int n) {  double h = (b - a) / n;  double sum = 0.5 \* (f(a) + f(b));  for (int i = 1; i < n; i++) {  sum += f(a + i \* h);  }  return h \* sum;  }  // Simpson's 1/3 Rule  double simpsons\_one\_third(double a, double b, int n) {  if (n % 2 != 0) {  n++; // n must be even for Simpson's 1/3 rule  }  double h = (b - a) / n;  double sum = f(a) + f(b);  for (int i = 1; i < n; i++) {  if (i % 2 == 0) {  sum += 2 \* f(a + i \* h);  } else {  sum += 4 \* f(a + i \* h);  }  }  return (h / 3) \* sum;  }  // Simpson's 3/8 Rule  double simpsons\_three\_eighth(double a, double b, int n) {  if (n % 3 != 0) {  n += 3 - (n % 3); // n must be a multiple of 3 for Simpson's 3/8 rule  }  double h = (b - a) / n;  double sum = f(a) + f(b);  for (int i = 1; i < n; i++) {  if (i % 3 == 0) {  sum += 2 \* f(a + i \* h);  } else {  sum += 3 \* f(a + i \* h);  }  }  return (3 \* h / 8) \* sum;  }  int main() {  double a = 4.0;  double b = 5.2;  int n = 6; // Number of subintervals  double result\_trapezoidal = trapezoidal(a, b, n);  double result\_simpsons\_one\_third = simpsons\_one\_third(a, b, n);  double result\_simpsons\_three\_eighth = simpsons\_three\_eighth(a, b, n);  printf("Trapezoidal Rule: %.6f\n", result\_trapezoidal);  printf("Simpson's 1/3 Rule: %.6f\n", result\_simpsons\_one\_third);  printf("Simpson's 3/8 Rule: %.6f\n", result\_simpsons\_three\_eighth);  return 0;  } |
| //guass  import java.util.Arrays; import java.util.Scanner;  import static java.lang.System.*exit*;  public class gauss1 {   static void printM(double[][] matrix) {  for (double[] row : matrix) {  System.*out*.println(Arrays.*toString*(row));  }  }   public static void main(String[] args) {   int n\_variables;  Scanner myScanner = new Scanner(System.*in*);  System.*out*.println("Enter the number of variables:");  n\_variables = myScanner.nextInt();  System.*out*.println("Size of augmented matrix is :" + n\_variables + " rows and " + (n\_variables + 1) + " columns");   double[][] aug\_mat = new double[n\_variables][n\_variables + 1];  System.*out*.println("Enter the values of augmented matrix:");  for (int i = 0; i < n\_variables; i++) {  for (int j = 0; j < (n\_variables + 1); j++) {  System.*out*.print("Enter vale for aug mat: " + (i + 1) + " and " + (j + 1) + ": ");  aug\_mat[i][j] = myScanner.nextDouble();  }  }   System.*out*.println("Augmented matrix: ");  *printM*(aug\_mat);   *//elimination process* for (int i = 0; i < n\_variables; i++) {   if (aug\_mat[i][i] == 0) {  System.*out*.println("Mathematical Error");  *exit*(0);  }  for (int j = i + 1; j < n\_variables; j++) {  double x = aug\_mat[j][i];  for (int k = 0; k < (n\_variables + 1); k++) {  */\*  cross multiply with first co-efficients ->  r[i-1]\*r[1][0] - r\*r[0][0]  \*/  //System.out.println(aug\_mat[j][k] + " \* " + aug\_mat[i][i] + " - " + aug\_mat[i][k] + " \* " + aug\_mat[j][i]);* aug\_mat[j][k] = aug\_mat[j][k] \* aug\_mat[i][i] - aug\_mat[i][k] \* x;  }  }  }  System.*out*.println("Updated Augmented matrix: ");  *printM*(aug\_mat);  *//solution by back substitution* double[] soln = new double[n\_variables];  *//get z* soln[n\_variables - 1] = aug\_mat[n\_variables - 1][n\_variables] / aug\_mat[n\_variables - 1][n\_variables - 1];   *//get the remaining by back substitution iteratively* for (int i = (n\_variables - 2); i >= 0; i--) {  soln[i] = aug\_mat[i][n\_variables];  for (int j = i + 1; j < n\_variables; j++) {  soln[i] -= soln[j] \* aug\_mat[i][j];  }  soln[i] /= aug\_mat[i][i];  }  System.*out*.println("Solution: ");  System.*out*.println(Arrays.*toString*(soln));  } } |
| //runge-kutta #include <stdio.h>  // Define the differential equation y' = y - 2xy  double f(double x, double y) {  return y - 2 \* x \* y;  }  // Runge-Kutta 4th order method  void rungeKutta4(double x0, double y0, double h, double xn) {  int n = (int)((xn - x0) / h);  double k1, k2, k3, k4, k;  double x = x0;  double y = y0;  printf("x\t\ty\n");  printf("%.2f\t%.6f\n", x, y);  for (int i = 0; i < n; i++) {  k1 = h \* f(x, y);  k2 = h \* f(x + h / 2.0, y + k1 / 2.0);  k3 = h \* f(x + h / 2.0, y + k2 / 2.0);  k4 = h \* f(x + h, y + k3);  k = (k1 + 2 \* k2 + 2 \* k3 + k4) / 6.0;  x = x + h;  y = y + k;  printf("%.2f\t%.6f\n", x, y);  }  }  int main() {  double x0 = 0.0; // Initial x value  double y0 = 1.0; // Initial y value (y(0) = 1)  double h = 0.2; // Step size  double xn = 2.0; // Final x value  rungeKutta4(x0, y0, h, xn);  return 0;  } |
| // lagrange Interpolation  #include <stdio.h>  int main() {  // Write C code here  printf("Lagrange Interpolation\n");  printf("Enter the number of values: ");  int n;  scanf("%d", &n);    double x\_vals[n], y\_vals[n];  printf("\nEnter x values:\n");  for(int i = 0; i < n; i++){  printf("x[%d]: ", i);  scanf("%lf", &x\_vals[i]);  }  printf("\nEnter y values:\n");  for(int i = 0; i < n; i++){  printf("y[%d]: ", i);  scanf("%lf", &y\_vals[i]);  }    double x\_p;  printf("\nEnter the interpolation point (x\_p): ");  scanf("%lf", &x\_p);    double y\_p = 0;  for(int i = 0; i < n; i++){  double prod = 1;  for(int j = 0; j < n; j++){  if(i != j){  prod = prod \* ((x\_p - x\_vals[j]) / (x\_vals[i] - x\_vals[j]));  }  }  y\_p = y\_p + (prod \* y\_vals[i]);  }  printf("Interpolated value at %.5lf is %.5lf\n", x\_p, y\_p);  return 0;  } |
|  |