Description of the problem

There are many cases in science and engineering where there are noisy sets of data and we need to estimate the straight line that “best fits” the data. This problem is called the linear regression problem. Given a noisy set of measurements (x, y) that appear to fall along a straight line, how can we find the equation of the line y = mx + b example problem

Calculate the slope m and intercept b of a least squares line that best fits an input data set consisting of an arbitrary number of (x, y) pairs. The input (x, y) data resides in a user-specified input file or a user input value.

factor that influence linear regration model

two-dimensional sample points with one independent variable and one dependent variable (conventionally, the x and y coordinates in a Cartesian coordinate system) and finds a linear function (a non-vertical straight lie) that, as accurately as possible, predicts the dependent variable values as a function of the independent variable

Matimatical Model

A standard method for finding the regression coefficients m and b is the method of least squares. This method is named “least squares” because it produces the line y = mx + b The slope of the least squares line is given by m =((Σxy) − (Σx)y)/((Σx2) − (Σx)x) b = y − mx where Σx is the sum of the x values Σx2 is the sum of the squares of the x values Σxy is the sum of the products of the corresponding x and y values x is the mean (average) of the x values y is the mean (average) of the y values

writing the matimatical Model in to fortran code

```
!*******************************************************************************************
*************************************
!
!                                    project assinment
!
!  Program:     LINEAR REGRATION
!
!  Programmer:   TADELE TATEK GEBREWOLD
!            Department of COMPUTATIONAL DATA SCIENCE
!            ADDIS ABABA UNIVERSITY
!
!
!  Date:       January 24, 2021
!
!  Language:    Fortran-9
!
!  Description:  This program performs a linear regression analysis for a set of data given as (x,y) pairs.  The output f
rom
!            the program is the slope and y-intercept of the least-squares best fit straight line through the data points.
!
!*******************************************************************************************
*************************************


!*******************************************************************************************
*************************************
!  Main program
!*******************************************************************************************
*************************************

program linreg

  implicit none                                        ! no default data types
  integer, parameter  :: dbl = kind (0.0d0)                    ! define kind for double precision

  real(dbl)          ::  b                                     ! y-intercept of least-squares best fit line
  real(dbl)          ::  m                                     ! slope of least-squares best fit line
  real(dbl)          ::  n = 0.0d0                             ! number of data points
  real(dbl)          ::  r                                     ! squared correlation coefficient
  character (len=80)  ::  str                                  ! input string
  real(dbl)          ::  sumx  = 0.0d0                         ! sum of x
  real(dbl)          ::  sumx2 = 0.0d0                         ! sum of x**2
  real(dbl)          ::  sumxy = 0.0d0                         ! sum of x * y
  real(dbl)          ::  sumy  = 0.0d0                         ! sum of y
  real(dbl)          ::  sumy2 = 0.0d0                         ! sum of y**2
  real(dbl)          ::  x                                     ! input x data
  real(dbl)          ::  y                                     ! input y data

  write (unit=*, fmt="(a)") " LINREG - Perform linear regression"         ! print introductory message
  write (unit=*, fmt="(a/)") "   (Enter END to stop data entry and compute"//  &
                    " linear regression.)"

  do                                                           ! loop for all data points
    write (unit=*, fmt="(a)", advance="no") " Enter x y:  "    ! prompt to enter data
    read (unit=*, fmt="(a)") str                               ! read x and y into string
    if (str == "end" .or. str == "END") exit                   ! if no more data, then exit loop
    read (unit=str, fmt=*) x, y                                ! else read x and y from string

    n = n + 1.0d0                                              ! increment number of data points by 1
    sumx  = sumx + x                                           ! compute sum of x
    sumx2 = sumx2 + x * x                                      ! compute sum of x**2
    sumxy = sumxy + x * y                                      ! compute sum of x * y
    sumy  = sumy + y                                           ! compute sum of y
    sumy2 = sumy2 + y * y                                      ! compute sum of y**2
  end do

  m = (n * sumxy  -  sumx * sumy) / (n * sumx2 - sumx**2)                  ! compute slope
  b = (sumy * sumx2  -  sumx * sumxy) / (n * sumx2  -  sumx**2)            ! compute y-intercept



  write (unit=*, fmt="(/a,es15.6)") " Slope        m = ", m                ! print results
  write (unit=*, fmt="(a, es15.6)") " y-intercept  b = ", b


end program linreg
```

Analysis and Interpretation

```
program shots
use aplot
implicit none

   integer::n
   real, dimension(:), allocatable::totalshots, goals

   ! Arrays for calling LAPACK's SGELS driver
   real, dimension(:,:), allocatable::bdata, adata
   real, dimension(:), allocatable::work
   integer::info

   ! A plot for when we're done
   type(aplot_t)::p
   integer::i
   real, dimension(:), allocatable::fitdata

   ! First, load all our data
   call loaddata()

   ! The 'n' variable now holds the number of data points,
   ! and our totalshots and goals arrays should be properly
   ! dimensioned and populated.

   ! Build arrays for LAPACK calls
   allocate(adata(n,2), bdata(n,1), work(2*n))
   adata(:,1) = 1.0            ! So an intercept is calculated
   adata(:,2) = totalshots      ! Total shots
   bdata(:,1) = goals          ! Goals

   ! Perform a least squares fit - documentation at:
   ! http://www.netlib.org/lapack/explore-html/d0/db8/group__real_g_esolve_gacd49b6b29636a826370633a8856bd
3bd.html
   call SGELS('N', &   ! TRANS
           n, &     ! M
           2, &     ! N
           1, &     ! NRHS
           adata, & ! A
           n, &     ! LDA
           bdata, & ! B
           n, &     ! LDB
           work, &  ! WORK
           2*n, &   ! LWORK
           info)

   Print *, "Result (0=success): ", info
   Print *, "Intercept: ", bdata(1,1)
   Print *, "Slope: ", bdata(2,1)

   ! Generate regression data for each totalshots point
   allocate(fitdata(n))
   do i = 1, n
      fitdata(i) = bdata(1,1) + totalshots(i)*bdata(2,1)
   end do

   ! Plot the results
   p = initialize_plot()

   call add_dataset(p, totalshots, goals)
   call set_seriestype(p, 0, APLOT_STYLE_DOT)
   call set_serieslabel(p, 0, "Player Data")

   call add_dataset(p, totalshots, fitdata)
```

```
    call set_seriestype(p, 1, APLOT_STYLE_LINE)
    call set_serieslabel(p, 1, "Linear Fit")

    call set_xlabel(p, "Total Shots")
    call set_ylabel(p, "Goals Scored")

    call set_title(p, "sample Data: Total Shots vs. Goals Scored")

    call display_plot(p)
    call destroy_plot(p)

contains

    ! Simply loads the NWSL data from a text file
    subroutine loaddata()
    implicit none

        integer::row_shots, row_goals
        integer::i

        open(unit=100, file='nwsl2016.txt', status='old')

        ! Two header rows
        read(100, *)
        read(100, *)

        ! First number is the number of data points
        read(100, *) n

        ! Allocate data arrays
        allocate(totalshots(n), goals(n))

        ! Load each data point as integers and re-store
        ! them in our arrays as REAL values
        do i = 1, n
            read(100, *) row_shots, row_goals
            totalshots(i) = real(row_shots)
            goals(i) = real(row_goals)
        end do

        close(100)

    end subroutine loaddata

end program shots
```

raw Data

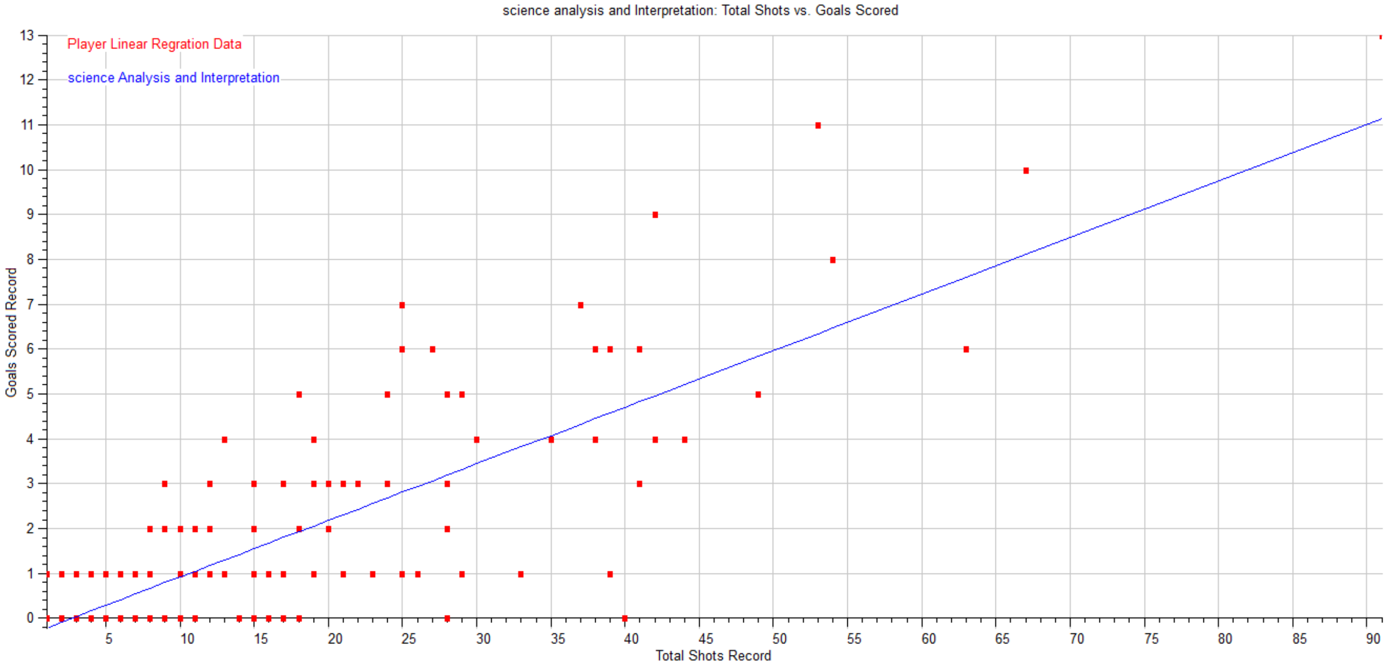
```
Column 1: All Shots
Column 2: Goals
184
91 13
67 10
63 6
54 8
```

53	11
49	5
44	4
42	9
42	4
41	6
41	3
40	0
39	6
39	1
39	1
38	6
38	4
37	7
35	4
33	1
30	4
29	5
29	1
29	1
28	5
28	3
28	2
28	0
28	0
27	6
26	1
25	7
25	6
25	1
24	5
24	3
23	1
22	3
21	3
21	3
21	1
21	1
20	3
20	2
19	4
19	3
19	1
18	5
18	2
18	2
18	0
17	3
17	1
17	1
17	0
17	0
17	0
16	1

16	0
15	3
15	2
15	1
15	0
14	0
14	0
14	0
14	0
14	0
13	4
13	1
12	3
12	2
12	1
12	1
12	1
11	2
11	1
11	0
10	2
10	2
10	1
10	1
10	1
10	1
10	0
10	0
10	0
10	0
9	3
9	2
9	0
9	0
8	2
8	1
8	0
8	0
7	1
7	1
7	1
7	0
7	0
7	0
7	0
7	0
6	1
6	1
6	1
6	0
6	0
6	0
6	0
6	0

```
6 0
6 0
6 0
6 0
6 0
6 0
5 1
5 1
5 1
5 0
5 0
5 0
5 0
4 1
4 0
4 0
4 0
4 0
4 0
4 0
4 0
4 0
4 0
3 1
3 0
3 0
3 0
3 0
3 0
3 0
3 0
3 0
3 0
2 1
2 1
2 1
2 0
2 0
2 0
2 0
2 0
2 0
2 0
2 0
2 0
2 0
2 0
2 0
2 0
2 0
2 0
1 1
1 1
1 1
```

```
1 0
1 0
1 0
1 0
1 0
1 0
1 0
1 0
1 0
1 0
1 0
1 0
1 0
1 0
1 0
1 0
1 0
1 0
```


science analysis and Interpretation: Total Shots vs. Goals Scored
Player Linear Regration Data
science Analysis and Interpretation
Goals Scored Record
Total Shots Record
0
1
2
3
4
5
6
7
8
9
10
11
12
13
5
10
15
20
25
30
35
40
45
50
55
60
65
70
75
80
85
90