МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ
УНИВЕРСИТЕТ им. А. И. ГЕРЦЕНА»
Институт информационных технологий и технологического образования
Кафедра информационных технологий и электронного обучения

КУРСОВАЯ РАБОТА

РАЗРАБОТКА ПРОГРАММЫ ДЛЯ РАСЧЁТА УРАВНЕНИЯ ПАРНОЙ РЕГРЕССИИ МЕТОДОМ НАИМЕНЬШИХ КВАДРАТОВ

Направление подготовки: «Информатика и вычислительная техника»

Руководитель:
доктор педагогических наук, профессор
_____ Власова Е. З.
« » _____ 2021 г.

Автор работы:
студент группы 2ИВТ(2)/19
_____ Мельников Ф. В.
« » _____ 2021 г.

Санкт-Петербург
2021

Оглавление

Введение

Регрессионный анализ используется при решении большого количества практических задач, которые предполагают определение связи между результативным значением и некоторыми факторами и определение прогнозных значений результативного признака при влиянии определённых факторов.

При проведении анализа используется, как правило, большой объём статистических данных. В связи с этим целесообразно проводить вычисления с использованием ЭВМ.

Цель работы – разработать программу для вычисления параметров парной регрессии.

Задачи:

- составить математическую модель;
- реализовать алгоритм вычисления параметров регрессии с использованием языка программирования;
- разработать графический интерфейс программы.

Глава 1. Определение объекта моделирования

Глава 1.1. Основные определения

Регрессионный анализ – набор статистических методов исследования влияния одной или нескольких независимых переменных X_1, X_2, …, X_n на зависимую переменную Y. Терминология зависимых и независимых переменных отражает корреляцию между величинами.

Целями регрессионного анализа являются определение степени детерминированности вариации зависимой переменной независимыми переменными, вычисление прогнозного значения зависимой переменной с помощью независимых, определение вклада отдельных независимых переменных в вариацию зависимой.

Корреляция (корреляционная зависимость) – статистическая взаимосвязь двух или более случайных величин (либо величин, которые можно с некоторой допустимой степенью точности считать таковыми). При этом изменения значений одной ил нескольких из этих величин сопутствуют систематическому изменению значений другой или других величин.

Регрессия – односторонняя зависимость, устанавливающая соответствие между случайными переменными, т.е. математическое выражение, отражающее связь между зависимой переменной Y и независимыми переменными X_1, X_2, …, X_n при условии, что это выражение будет иметь статистическую значимость.

Одним из базовых методов регрессионного анализа для оценки неизвестных параметров регрессионных моделей по выборочным данным является метод наименьших квадратов – математический метод, применяемый для решения различных задач, основанный на минимизации суммы квадратов отклонений некоторых функций от исходных переменных.

Линия регрессии может быть найдена в виде линейной или полиномиальной функции, наилучшим образом приближающей искомую кривую. С помощью метода наименьших квадратов минимизируется сумма квадратов отклонений реально наблюдаемых Y от их оценок $\hat{Y}$.

Глава 1.2. Математическая модель

Модель множественной линейной регрессии предназначена для проверки и изучения связи между одной зависимой переменной и несколькими независимыми переменными. Предполагается, что такая связь теоретически может быть описана линейной функцией вида

$$Y = \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_k x_k + U, \quad (1.1)$$

где Y – зависимая (объясняемая) переменная – регрессанд, U – случайная составляющая модели, x_j – независимые (объясняющие) переменные – регрессоры.

Модель множественной регрессии является обобщением модели парной линейной регрессии на многомерный случай.

Параметры уравнения регрессии могут быть найдены при решении системы нормальных уравнений:

$$X^T X b = X^T y, \quad (1.2)$$

которая содержит k линейных уравнений относительно k неизвестных b_j, $i = 1, 2, \ldots, k$, где

y – вектор-столбец наблюдений зависимой переменной (регрессанда)

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix},$$

X – матрица наблюдений независимых переменных регрессоров

$$X = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1k} \\ x_{21} & x_{22} & \cdots & x_{2k} \\ \vdots & \vdots & \cdots & \vdots \\ x_{i1} & x_{i2} & \cdots & x_{ik} \\ \cdots & \cdots & \cdots & \cdots \\ x_{n1} & x_{n2} & \cdots & x_{nk} \end{bmatrix},$$

b – вектор-столбец коэффициентов (параметров)

$$b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_k \end{bmatrix}.$$

Таким образом,

y – вектор-столбец размерности n, $y_1, y_2, \dots, y_n$ – наблюдения зависимой переменной (регрессанда);

X – матрица размерности $n \times k$, i-ый столбец матрицы X (кроме первого) содержит n наблюдений i-го регрессора, первый столбец состоит из единиц - $x_{i1} = 1, i = 1, 2, \dots, n$;

b – вектор-столбец размерности k, $b_1, b_2, \dots, b_k$ – коэффициенты уравнения регрессии.

Решение системы нормальных уравнений в виде расчётной формулы можно получить только в матричной форме. Матрица X имеет полный ранг и квадратная матрица X^TX размерности $k \times k$ также имеет полный ранг при выполнении следующих предпосылок классической многомерной линейной регрессионной модели:

- Решение задачи оценивания параметров существует и является единственным.
- Регрессоры не коллинеарны. В этом случае матрица наблюдений регрессоров должна быть полного ранга, т.е. её столбцы должны быть линейно независимы.
- Количество наблюдений больше количества оцениваемых параметров, т.е. $n > k$.

Следовательно, существует обратная матрица $(X^TX)^{-1}$. При умножении обеих частей уравнения (1.1) на эту матрицу получим

$$(X^TX)^{-1}(X^TX)b = (X^TX)^{-1}X^Ty. \quad (1.3)$$

Далее, учитывая, что $(X^T X)^{-1}(X^T X) = I_k$, где I_k – единичная матрица размерности k, получаем выражение для оценок коэффициентов в виде:

$$b = (X^T X)^{-1} X^T y \tag{1.4}$$

Формула (1.4) определяет оценку коэффициентов многомерной линейной регрессии по методу наименьших квадратов.

Оценённую с помощью метода наименьших квадратов эмпирическую линейную регрессионную функцию можно записать в виде

$$\hat{y}_i = b^T x_i = b_1 x_{i1} + b_2 x_{i2} + \cdots + b_k x_{ik}, \tag{1.5}$$

где b – оптимальная по методу наименьших квадратов оценка вектора коэффициентов регрессии, которая определяется выражением (1.4), $x_i = (x_{i1}, x_{i2}, \dots, x_{ik})^T$ – вектор-столбец размерности k.

Оценённый (эмпирический) регрессионный коэффициент b_j, $j = 1, 2, \dots, k$ является частной производной эмпирической регрессионной функции по j-му регрессору (независимой переменной). Он показывает, на сколько изменится оценённое значение $\hat{y}_i$ при изменении j-ого регрессора на единицу при фиксированных значениях остальных регрессоров.

Модель парной полиномиальной регрессии степени m описывается аналогично.

В случае полиномиальной зависимости при парной регрессии матрица X будет иметь следующий вид:

$$X = \begin{bmatrix} 1 & x_{12} & x_{12}^2 & \cdots & x_{1k}^m \\ 1 & x_{22} & x_{22}^2 & \cdots & x_{2k}^m \\ \vdots & \vdots & \vdots & \cdots & \vdots \\ 1 & x_{i2} & x_{i2}^2 & \cdots & x_{ik}^m \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ 1 & x_{n2} & x_{n2}^2 & \cdots & x_{nk}^m \end{bmatrix},$$

где m – степень уравнения регрессии.

Глава 1.3. Операции над матрицами, применяемые для оценки параметров регрессии

Оценка коэффициентов парной полиномиальной регрессии включает следующие операции с матрицами:

- транспонирование матрицы;
- умножение матриц;
- нахождение обратной матрицы.

Транспонирование матрицы

Транспонированная матрица – матрица A^T, полученная из исходной матрицы A заменой строк на столбцы.

Транспонированной матрицей для матрицы A размером $m \times n$ является матрица A^T размером $n \times m$, определённая следующим образом:

$$A_{ij}^T = A_{ji}. \quad (2.1)$$

Умножение матриц

Пусть даны две прямоугольные матрицы A и B размерности $l \times m$ и $m \times n$ соответственно. Тогда матрица C размерностью $l \times n$, в которой

$$c_{ij} = \sum_{r=1}^{m} a_{ir} b_{rj} \quad (i = 1, 2, \dots, l; j = 1, 2, \dots, n), \quad (2.2)$$

называется их произведением.

Операция умножения двух матриц выполнима только в том случае, если число столбцов в первом сомножителе равно числу строк во втором. Умножение всегда выполнимо, если оба сомножителя – квадратные матрицы одного и того же порядка.

Вычисление обратной матрицы

Обратная матрица – такая матрица A^{-1}, при умножении на которую исходная матрица A даёт в результате единичную матрицу E:

$$AA^{-1} = A^{-1}A = E. \quad (2.3)$$

Квадратная матрица обратима тогда и только тогда, когда она не является вырожденной, т.е. её определитель не равен нулю.

Если матрица обратима, то для нахождения обратной матрицы можно использовать следующие методы:

- метод Гаусса—Жордана,
- с помощью матрицы алгебраических дополнений,
- метод треугольной факторизации (LU/LUP-разложение).

Алгоритм метода Гаусса—Жордана:

- Выбирают первый слева столбец матрицы, в котором есть хоть одно отличное от нуля значение.
- Если самое верхнее число в этом столбце ноль, то меняют всю первую строку матрицы с другой строкой матрицы, где в этой колонке нет нуля.
- Все элементы первой строки делят на верхний элемент выбранного столбца.
- Из оставшихся строк вычитают первую строку, умноженную на первый элемент соответствующей строки, с целью получить первым элементом каждой строки (кроме первой) ноль.
- Далее проводят такую же процедуру с матрицей, получающейся из исходной матрицы после вычёркивания первой строки и первого столбца.
- После повторения этой процедуры n-1 раз получают верхнюю треугольную матрицу.
- Вычитают из предпоследней строки последнюю строку, умноженную на соответствующий коэффициент, с тем, чтобы в предпоследней строке осталась только 1 на главной диагонали.
- Повторяют предыдущий шаг для последующих строк. В итоге получают единичную матрицу и решение на месте свободного вектора (с ним необходимо проводить все те же преобразования).

Глава 2. Разработка программы для вычисления параметров уравнения регрессии

Глава 2.1. Реализация алгоритма вычисления параметров регрессии с использованием языка программирования

В соответствии с формулой вычисления коэффициентов парной регрессии необходимо реализовать следующие операции:

- транспонирование матрицы;
- умножение матриц;
- нахождение обратной матрицы.

Транспонирование матрицы реализуется в соответствии с определением (2.1).

Умножение матриц реализуется в соответствии с определением (2.2).

Метод Гаусса—Жордана реализуется в соответствии с алгоритмом, приведённым в гл. 1.3.

Алгоритм вычисления параметров регрессии реализован на языке C++. Исходный код программы размещён в приложении (функции *transpose_matrix*, *multiply_matrix*, *inverse_matrix*).

Алгоритм заключается в следующем:

- составление матрицы X для полинома степени m;
- транспонирование матрицы X;
- умножение матрицы XT на матрицу X;
- вычисление обратной матрицы $(X^TX)^{-1}$;
- вычисление вектор-столбца $b = (X^TX)^{-1}y$.

Глава 2.2. Разработка программы

Программа написана на языке C++. Для построения графического интерфейса пользователя используются библиотеки IUP, CD.

IUP – кроссплатформенная библиотека базовых элементов графического пользовательского интерфейса с использованием языков C и Lua.

CD – кроссплатформенная библиотека для работы с векторной графикой.

Данные библиотеки разработаны Tecgraf/PUC-Rio в сотрудничестве с PETROBRAS/CENPES и имеют свободную лицензию MIT.

Программа реализует следующие функции:

- вычисление коэффициентов уравнения парной линейной или полиномиальной регрессии;
- вычисление прогнозного значения зависимой переменной с помощью независимых;
- ввод зависимых и независимых величин в виде таблицы;
- загрузка таблицы из файла;
- сохранение таблицы в файл;
- выбор типа уравнения регрессии;
- вывод коэффициентов уравнения;
- вывод уравнения регрессии;
- построение линии регрессии;
- настройка параметров осей графика.

Общий вид программы представлен на рисунке 1.

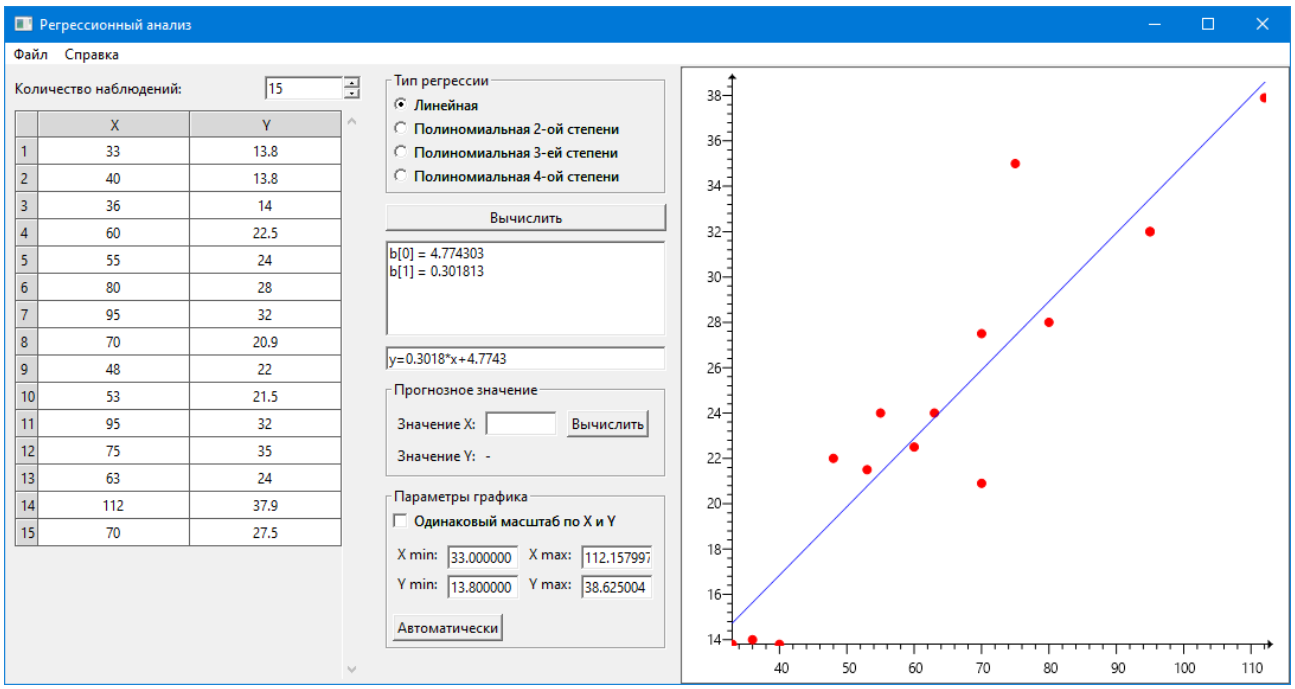

Рисунок 1 – Общий вид программы

В поле «Количество наблюдений» вводится количество наблюдений. Данные наблюдений вводятся в таблицу. Затем необходимо выбрать тип регрессии. После нажатия кнопки «Вычислить» производится вычисление, и затем в текстовых полях выводятся коэффициенты уравнения регрессии и уравнение регрессии. Программа строит график зависимости Y(X) и отображает выводит линию регрессии в области построения.

Для определения прогнозного значения необходимо ввести значение зависимой переменной X. Вычисленное значение независимой переменной Y отображается в метке «Значение Y».

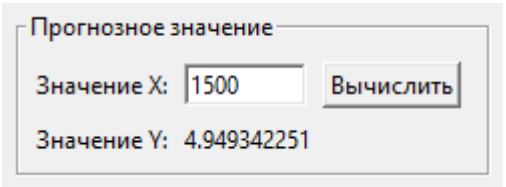

Рисунок 2 – Расчёт прогнозного значения

Для приближения графика необходимо с зажатой клавишей Ctrl использовать колёсико мыши. Возможно настроить параметры графика – установить одинаковый масштаб по осям X и Y, задать координаты границ осей или подобрать их автоматически. При нажатии правой кнопкой мыши в области построения открывается контекстное меню. Для отображения/скрытия сетки

необходимо нажать «Show/Hide Grid». Для копирования изображения графика необходимо выбрать пункт «Copy», для сохранения изображения в файл – «Export», для печати – «Print».

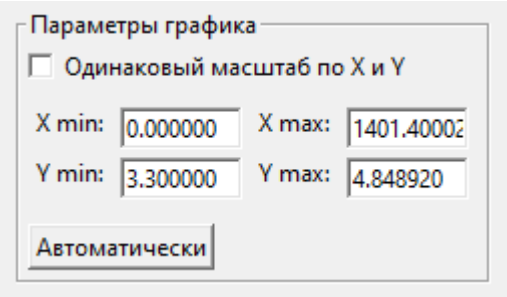

Рисунок 3 – Настройка параметров области построения

Информация для сборки программы

Интегрированная среда разработки: Code::Blocks версии 20.03.

Программа компилируется с помощью компилятора *gcc* (набор инструментов MinGW).

Для сборки программы требуются следующие модули: *iup_plot, iupcontrols, cdcontextplus, iupcd, cd, iupgl, cdgl, ftgl, freetype6, z, iup, gdi32, user32, comdlg32, kernel32, comctl32, kernel32, uuid, ole32, opengl32, gdiplus, winspool, glu32*.

Глава 2.3. Решение задачи с использованием программы

Постановка задачи:

Было проведено измерение выходного напряжения датчика газа при различных содержаниях некоторого газа в воздухе. Имеются следующие данные:

X	0	150	270	390	400	700	710	1020	1120	1400
Y	3,3	3,45	3,65	3,92	4	4,2	4,21	4,5	4,56	4,7

где X – концентрация газа в воздухе (ppm),

Y – выходное напряжение датчика (В).

Требуется построить график зависимости между переменными, рассчитать уравнение регрессии, определить прогнозное значение напряжения при концентрации газа в 1500 ppm.

Общий вид программы с заданными исходными данными представлен на рисунке 4.

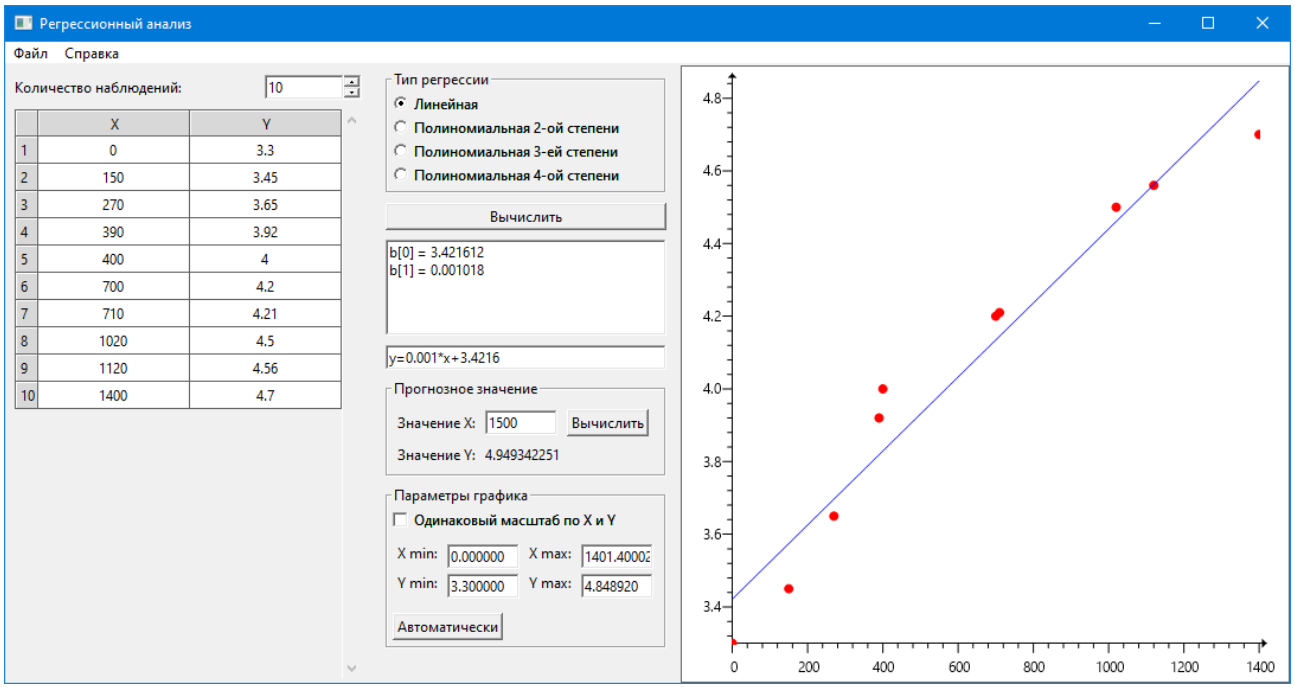

Рисунок 4 – общий вид программы

Характер расположения точек на графике показывает, что связь между переменными может выражаться линейным уравнением регрессии.

График зависимости между переменными с линией регрессии представлен на рисунке 5.

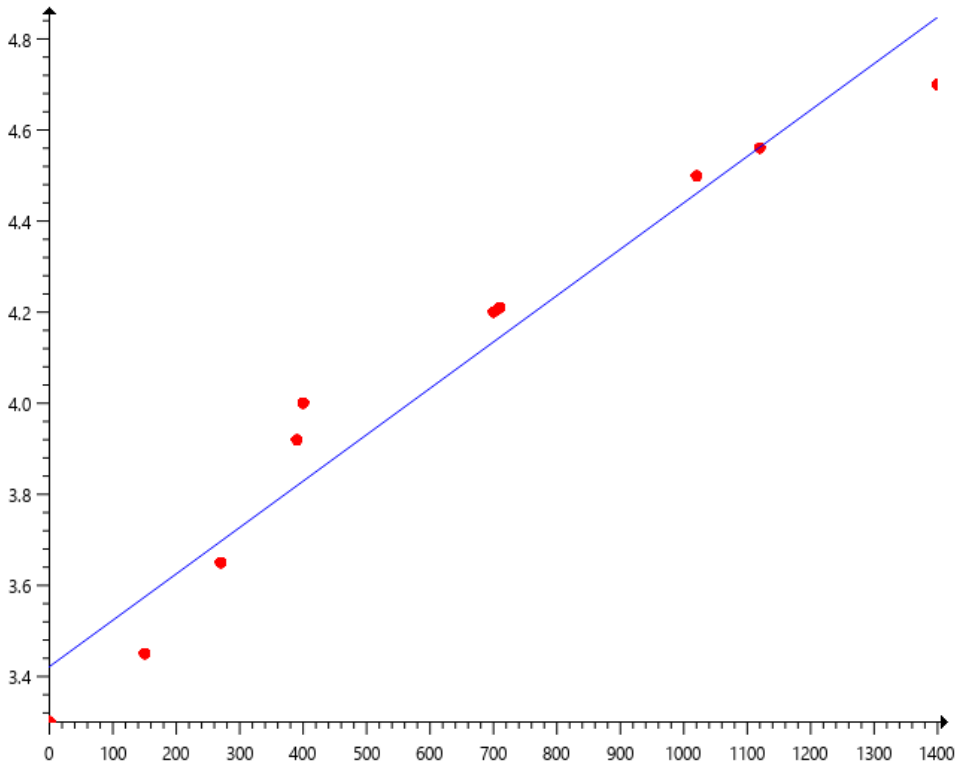

Рисунок 5 – график зависимости между переменными с линией регрессии

Вычисленное уравнение регрессии: $y = 0.001x + 3.4216$.

Прогнозное значение при концентрации газа в 1500 ppm при линейной регрессии примерно равно 4,949.

Заключение

В ходе работы была разработана программа для вычисления коэффициентов уравнения парной регрессии.

Программа позволяет определить зависимость между зависимыми и независимыми переменными. Реализована возможность вычисления параметров для линейной и полиномиальной парной регрессии, вычисления прогнозного значения зависимой переменной при заданном значении независимой переменной.

Разработанная программа отображает результаты вычисления в графическом виде.

Программа может применяться при решении задач регрессионного анализа в случае влияния одной зависимой переменной на зависимую переменную.

Список источников

1. Магнус Я. Р., Катышев П. К., Пересецкий А. А. Эконометрика. Начальный курс: Учеб. – 6-е изд., перераб. и доп. – М.: Дело, 2004. – 576 с.

2. Домбровский В. В. Эконометрика. [Электронный ресурс] / Томский государственный университет. – Томск, 2016. URL: http://sun.tsu.ru/mminfo/2016/Dombrovski/start.htm (дата обращения: 28.04.2021).

3. Метод Гаусса — Жордана [Электронный ресурс] / Википедия. URL: https://ru.wikipedia.org/wiki/Метод_Гаусса_—_Жордана (дата обращения: 03.05.2021).

4. IUP - Portable User Interface [Электронный ресурс] / Tecgraph/PUC-Rio. URL: https://www.tecgraf.puc-rio.br/iup/ (дата обращения: 03.05.2021).

5. CD - Canvas Draw [Электронный ресурс] / Tecgraph/PUC-Rio. URL: https://www.tecgraf.puc-rio.br/cd/ (дата обращения: 04.05.2021).

Приложение

Программный код

```
#include <stdio.h>
#include <stdlib.h>
#include <vector>
#include <string>
#include <cmath>
#include <sstream>
#include <fstream>
#include <time.h>

#include "iup.h"
#include "iupcontrols.h"
#include "cd.h"
#include "cdiup.h"
#include "iup_plot.h"
#include "iupgl.h"

Ihandle *dlg, *spin, *t_matrix, *plotter, *cs_text, *r_t1, *r_t2, *r_t3, *r_t4, *
eq_text, *fcast_x, *fcast_y, *xmin, *xmax, *ymin, *ymax;

typedef std::vector<double> num_vector;
typedef std::vector<num_vector> arr;

arr coefficients;

int r_type = 1;

void calculate_coefficients(arr &matrix, arr &coefficients, int r_type);
int set_plot_bounds_auto(Ihandle *self);

void generate_array(arr &matrix, int n, int m)
{
    matrix.clear();
    matrix.resize(n);
    for (auto &row : matrix) {
        row.resize(m, rand() % 25/*1*/);
        for (int i = 0; i < row.size(); i++) {
            row[i] += i + rand() % 25;
        }
    }
}

void transpose_matrix(arr &matrix)
{
    arr t = matrix;
    size_t r = t.size();
    size_t c = t[0].size();
    generate_array(matrix, c, r);
```

```
    for (unsigned int i = 0; i < r; i++) {
        for (unsigned int j = 0; j < c; j++) {
            matrix[j][i] = t[i][j];
        }
    }
}

void multiply_matrix(arr &a, arr &b, arr &result)
{
    size_t r1 = a.size();
    size_t c1 = a[0].size();
    size_t r2 = b.size();
    size_t c2 = b[0].size();
    if (c1 != r2) {
        printf("err");
    }
    generate_array(result, r1, c2);
    for (unsigned int i = 0; i < r1; i++) {
        for (unsigned int j = 0; j < c2; j++) {
            result[i][j] = 0;
            for (unsigned int k = 0; k < c1; k++) {
                result[i][j] += a[i][k] * b[k][j];
            }
        }
    }
}

void inverse_matrix(arr &matrix, arr &result)
{
    arr inv = matrix;
    int size = matrix.size();
    for (auto &row : inv) {
        row.resize(size*2);
    }
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < 2 * size; j++) {
            if (j == i + size) {
                inv[i][j] = 1;
            }
        }
    }
    for (int i = size - 1; i > 0; i--) {
        if (inv[i-1][0] < inv[i][0]) {
            num_vector t = inv[i];
            inv[i] = inv[i-1];
            inv[i-1] = t;
        }
    }
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
```

```
            if (j != i) {
                if (inv[i][i] == 0) {
                    throw 0;
                }
                double t = inv[j][i] / inv[i][i];
                for (int k = 0; k < 2 * size; k++) {
                    inv[j][k] -= inv[i][k] * t;
                }
            }
        }
    }
    for (int i = 0; i < size; i++) {
        double t = inv[i][i];
        for (int j = 0; j < 2 * size; j++) {
            inv[i][j] = inv[i][j] / t;
        }
    }
    for (auto &row : inv) {
        row.erase(row.begin(), row.begin()+size);
    }
    result = inv;
}

void calculate_coefficients(arr &matrix, arr &coefficients, int r_type)
{
    try {
        arr matrix_x = matrix;
        for (auto &row : matrix_x) {
            row.pop_back();
            row.insert(row.begin(), 1);
            if (r_type >= 2) {
                row.push_back(row[row.size()-1] * row[row.size()-1]);
            }
            for (int i = 3; i <= r_type; i++) {
                row.push_back(row[1] * row[row.size()-1]);
            }
        }
        arr matrix_xt = matrix_x;
        transpose_matrix(matrix_xt);
        arr matrix_m;
        multiply_matrix(matrix_xt, matrix_x, matrix_m);
        arr inv_matrix;
        inverse_matrix(matrix_m, inv_matrix);
        multiply_matrix(inv_matrix, matrix_xt, matrix_m);
        arr matrix_y = matrix;
        for (auto &row : matrix_y) {
            row.erase(row.begin());
        }
        multiply_matrix(matrix_m, matrix_y, coefficients);
    }
```

```
    catch (int e) {
        throw 1;
    }
}

int calculate(Ihandle *self)
{
    try {
        arr matrix;
        int rows_count = IupGetInt(t_matrix, "NUMLIN");
        generate_array(matrix, rows_count, 2);
        for (int i = 1; i <= rows_count; i++) {
            for (int j = 1; j <= 2; j++) {
                matrix[i-1][j-1] = IupGetFloatId2(t_matrix, "", i, j);;
            }
        }
        calculate_coefficients(matrix, coefficients, r_type);
        IupSetAttribute(plotter, "CLEAR", NULL);
        IupPlotBegin(plotter, 0);
        for (int i = 1; i <= rows_count; i++) {
            float x = IupGetFloatId2(t_matrix, "", i, 1);
            float y = IupGetFloatId2(t_matrix, "", i, 2);
            IupPlotAdd(plotter, x, y);
        }
        IupPlotEnd(plotter);
        IupSetAttribute(plotter, "DS_MODE", "MARK");
        IupSetAttribute(plotter, "DS_MARKSTYLE", "CIRCLE");
        IupSetAttribute(plotter, "REDRAW", NULL);
        std::string cs, r_eq;
        for (unsigned int i = 0; i < coefficients.size(); i++) {
            for (unsigned int j = 0; j < coefficients[i].size(); j++) {
                cs += "b[" + std::to_string(i) + "] = " + std::to_string(coeffici
ents[i][j]);
                if (i < coefficients.size() - 1) cs += "\n";
                std::string var;
                if (i == 1)
                    var = "*x";
                else if (i > 1)
                    var = "*x^" + std::to_string(i);
                std::string sign;
                if (coefficients[i][j] >= 0 && i < coefficients.size() - 1)
                    sign = "+";
                std::ostringstream out;
                out.precision(4 + floor(log10(fabs(coefficients[i][j])) + 1));
                out << coefficients[i][j];
                r_eq = sign + out.str() + var + r_eq;
            }
        }
        r_eq = "y=" + r_eq;
        IupSetAttribute(eq_text, "VALUE", r_eq.c_str());
```

```
        IupSetAttribute(cs_text, "VALUE", cs.c_str());
        IupPlotBegin(plotter, 0);
        double min_value = INT_MAX;
        double max_value = INT_MIN;
        for (auto &row : matrix) {
            if (row[0] < min_value) min_value = row[0];
            if (row[0] > max_value) max_value = row[0];
        }
        double d_x = fabs(max_value - min_value) / 1000;
        for (double x = min_value; x <= max_value+2*d_x; x+=d_x) {
            double x_p = 1, y = 0;
            for (unsigned int i = 0; i < coefficients.size(); i++) {
                double c = coefficients[i][0];
                y += c * x_p;
                x_p *= x;
            }
            IupPlotAdd(plotter, x, y);
        }
        IupPlotEnd(plotter);
        IupSetAttribute(plotter, "DS_COLOR", "0 0 255");
        IupSetAttribute(plotter, "REDRAW", NULL);
        set_plot_bounds_auto(NULL);
    }
    catch (int e)
    {
        IupMessageError(dlg, "Невозможно вычислить обратную матрицу.");
    }
    return IUP_DEFAULT;
}

int change_count(Ihandle *self)
{
    int row_count_num = IupGetInt(self, "VALUE");
    if (row_count_num > 100) {
        IupSetAttribute(self, "VALUE", "100");
    } else if (row_count_num < 2) {
        IupSetAttribute(self, "VALUE", "2");
    }
    char *row_count = IupGetAttribute(self, "VALUE");
    IupSetAttribute(t_matrix, "NUMLIN", row_count);
    int r_count = IupGetInt(self, "VALUE");
    for (int i = 1; i <= r_count; i++) {
        std::string num = std::to_string(i);
        std::string pos = num + ":0";
        IupSetAttribute(t_matrix, pos.c_str(), num.c_str());
    }
    IupSetAttribute(t_matrix, "REDRAW", NULL);
    return IUP_DEFAULT;
}
```

```
int set_rtype(Ihandle *self, int state)
{
    if (state != 1) return IUP_DEFAULT;
    if (self == r_t1)
        r_type = 1;
    else if (self == r_t2)
        r_type = 2;
    else if (self == r_t3)
        r_type = 3;
    else if (self == r_t4)
        r_type = 4;
    return IUP_DEFAULT;
}

int calc_forecast_value(Ihandle *self)
{
    double x = IupGetFloat(fcast_x, "VALUE");
    double x_pow = 1;
    double f_val = 0;
    for (int i = 0; i < coefficients.size(); i++) {
        if (i > 0) x_pow *= x;
        f_val += coefficients[i][0] * x_pow;
    }
    IupSetFloat(fcast_y, "TITLE", f_val);
    return IUP_DEFAULT;
}

int set_scaleequal(Ihandle *self, int state)
{
    if (state == 1) {
        IupSetAttribute(plotter, "AXS_SCALEEQUAL", "YES");
    } else {
        IupSetAttribute(plotter, "AXS_SCALEEQUAL", "NO");
    }
    set_plot_bounds_auto(NULL);
    IupSetAttribute(plotter, "REDRAW", NULL);
    return IUP_DEFAULT;
}

int set_xy_min(Ihandle *self, int state)
{
    if (state == 1) {
        IupSetAttribute(plotter, "AXS_XAUTOMIN", "NO");
        IupSetAttribute(plotter, "AXS_YAUTOMIN", "NO");
        IupSetAttribute(plotter, "AXS_XMIN", "0");
        IupSetAttribute(plotter, "AXS_YMIN", "0");
    } else {
        IupSetAttribute(plotter, "AXS_XAUTOMIN", "YES");
        IupSetAttribute(plotter, "AXS_YAUTOMIN", "YES");
    }
```

```
    IupSetAttribute(plotter, "REDRAW", NULL);
    return IUP_DEFAULT;
}

int set_plot_bounds(Ihandle *self)
{
    IupSetAttribute(plotter, "AXS_XAUTOMIN", "NO");
    IupSetAttribute(plotter, "AXS_YAUTOMIN", "NO");
    IupSetAttribute(plotter, "AXS_XAUTOMAX", "NO");
    IupSetAttribute(plotter, "AXS_YAUTOMAX", "NO");
    IupSetAttribute(plotter, "AXS_XMIN", IupGetAttribute(xmin, "VALUE"));
    IupSetAttribute(plotter, "AXS_XMAX", IupGetAttribute(xmax, "VALUE"));
    IupSetAttribute(plotter, "AXS_YMIN", IupGetAttribute(ymin, "VALUE"));
    IupSetAttribute(plotter, "AXS_YMAX", IupGetAttribute(ymax, "VALUE"));
    IupSetAttribute(plotter, "REDRAW", NULL);
    return IUP_DEFAULT;
}

int set_plot_bounds_auto(Ihandle *self)
{
    IupSetAttribute(plotter, "AXS_XAUTOMIN", "YES");
    IupSetAttribute(plotter, "AXS_YAUTOMIN", "YES");
    IupSetAttribute(plotter, "AXS_XAUTOMAX", "YES");
    IupSetAttribute(plotter, "AXS_YAUTOMAX", "YES");
    IupSetAttribute(plotter, "REDRAW", NULL);
    IupSetAttribute(xmin, "VALUE", std::to_string(IupGetFloat(plotter, "AXS_XMIN"
)).c_str());
    IupSetAttribute(xmax, "VALUE", std::to_string(IupGetFloat(plotter, "AXS_XMAX"
)).c_str());
    IupSetAttribute(ymin, "VALUE", std::to_string(IupGetFloat(plotter, "AXS_YMIN"
)).c_str());
    IupSetAttribute(ymax, "VALUE", std::to_string(IupGetFloat(plotter, "AXS_YMAX"
)).c_str());
    return IUP_DEFAULT;
}

int load_file(void)
{
    Ihandle *filedlg = IupFileDlg();
    IupSetAttribute(filedlg, "TITLE", "Загрузить данные из файла");
    IupSetAttribute(filedlg, "EXTFILTER", "Текстовые файлы (*.txt)|*.txt|Все файл
ы (*.*)|*.*");
    IupPopup(filedlg, IUP_CURRENT, IUP_CURRENT);
    if (IupGetInt(filedlg, "STATUS") != -1) {
        std::ifstream datafile (IupGetAttribute(filedlg, "VALUE"));
        if (datafile.is_open()) {
             std::vector<std::vector <std::string>> data;
             std::vector <std::string> row_t;
             std::string row, num;
             std::stringstream rs;
```

```
            while (getline(datafile, row)) {
                rs.str(row);
                while (rs >> num) {
                    row_t.push_back(num);
                }
                if (row_t.size() == 1) {
                    row_t.push_back(std::string());
                }
                if (row_t.size() != 0) {
                    data.push_back(row_t);
                }
                row_t.clear();
                rs.clear();
            }
            int rows_count = data.size();
            IupSetInt(spin, "VALUE", rows_count);
            IupSetInt(t_matrix, "NUMLIN", rows_count);
            for (int i = 0; i < rows_count; i++) {
                for (int j = 0; j < 2; j++) {
                    IupSetAttributeId2(t_matrix, "", i+1, j+1, data[i][j].c_str()
);
                }
            }
            IupSetAttribute(t_matrix, "REDRAW", NULL);
            change_count(spin);
            datafile.close();
        } else {
            IupMessageError(dlg, "Произошла ошибка при чтении файла.");
        }
    }
    return IUP_DEFAULT;
}

int save_file(void)
{
    Ihandle *filedlg = IupFileDlg();
    IupSetAttribute(filedlg, "DIALOGTYPE", "SAVE");
    IupSetAttribute(filedlg, "TITLE", "Сохранить данные в файл");
    IupSetAttribute(filedlg, "EXTFILTER", "Текстовые файлы (*.txt)|*.txt|Все файл
ы (*.*)|*.*");
    IupPopup(filedlg, IUP_CURRENT, IUP_CURRENT);
    if (IupGetInt(filedlg, "STATUS") != -1) {
        std::ofstream datafile (IupGetAttribute(filedlg, "VALUE"));
        if (datafile.is_open()) {
            int rows_count = IupGetInt(t_matrix, "NUMLIN");
            for (int i = 1; i <= rows_count; i++) {
                datafile << IupGetAttributeId2(t_matrix, "", i, 1) << " ";
                datafile << IupGetAttributeId2(t_matrix, "", i, 2) << "\n";
            }
            datafile.close();
```

```
        } else {
            IupMessageError(dlg, "Произошла ошибка при записи файла.");
        }
    }
    return IUP_DEFAULT;
}

int app_exit(void)
{
    return IUP_CLOSE;
}

int show_about(void)
{
    Ihandle *msg;
    msg = IupMessageDlg();
    IupSetAttribute(msg, "DIALOGTYPE", "INFORMATION");
    IupSetAttribute(msg, "TITLE", "О программе");
    IupSetAttribute(msg, "VALUE", "Программа для расчёта уравнения регрессии\n\n"
                                  "Данная программа использует библиотеки IUP и C
D\n"
                                  "Лицензия: MIT\n"
                                  "© 1994-2020 Tecgrap/PUC-Rio\n\n"
                                  "© 2021 Мельников Ф. В.");
    IupPopup(msg, IUP_CURRENT, IUP_CURRENT);
    return IUP_DEFAULT;
}

int main(int argv, char **argc)
{
    srand(time(NULL));
    Ihandle *menu, *item_open, *item_save, *item_exit, *file_menu, *file_submenu,
            *item_about, *help_menu, *help_submenu,
            *main_container, *button, *lc_label, *spin_container, *matrix_contain
er,
            *l_container, *m_container, *rt_select, *fcast_b1, *fcast_b2, *fcast,
 *fcastx_label, *fcasty_label, *fcast_button,
            *plot_params, *plot_pxy, *pscale_eq,
            *xmin_label, *xmax_label, *ymin_label, *ymax_label, *auto_bounds;
    IupOpen(&argv, &argc);
    IupControlsOpen();
    IupPlotOpen();
    lc_label = IupLabel("Количество наблюдений:");
    spin = IupText(NULL);
    IupSetAttribute(spin, "SPIN", "YES");
    IupSetAttribute(spin, "SPINMIN", "2");
    IupSetAttribute(spin, "VALUE", "15");
    IupSetCallback(spin, "VALUECHANGED_CB", (Icallback) change_count);
    spin_container = IupHbox(lc_label, spin, NULL);
    IupSetAttribute(spin_container, "ALIGNMENT", "ACENTER");
```

```
    IupSetAttribute(spin_container, "GAP", "70");
    IupSetAttribute(spin_container, "MARGIN", "4x4");
    t_matrix = IupMatrix(NULL);
    IupSetAttribute(t_matrix, "NUMLIN", "15");
    IupSetAttribute(t_matrix, "NUMCOL", "2");
    IupSetAttribute(t_matrix, "NUMCOL_VISIBLE", "2");
    IupSetAttribute(t_matrix, "0:1", "X");
    IupSetAttribute(t_matrix, "0:2", "Y");
    IupSetAttribute(t_matrix, "1:0", "1");
    IupSetAttribute(t_matrix, "YAUTOHIDE", "NO");
    button = IupButton("Вычислить", NULL);
    IupSetCallback(button, "ACTION", (Icallback) calculate);
    r_t1 = IupToggle("Линейная", NULL);
    r_t2 = IupToggle("Полиномиальная 2-ой степени", NULL);
    r_t3 = IupToggle("Полиномиальная 3-ей степени", NULL);
    r_t4 = IupToggle("Полиномиальная 4-ой степени", NULL);
    IupSetCallback(r_t1, "ACTION", (Icallback) set_rtype);
    IupSetCallback(r_t2, "ACTION", (Icallback) set_rtype);
    IupSetCallback(r_t3, "ACTION", (Icallback) set_rtype);
    IupSetCallback(r_t4, "ACTION", (Icallback) set_rtype);
    rt_select = IupFrame(IupRadio(IupVbox(r_t1, r_t2, r_t3, r_t4, NULL)));
    IupSetAttribute(rt_select, "TITLE", "Тип регрессии");
    IupSetAttribute(rt_select, "GAP", "4");
    IupSetAttribute(rt_select, "EXPAND", "YES");
    cs_text = IupText(NULL);
    IupSetAttribute(cs_text, "MULTILINE", "YES");
    IupSetAttribute(cs_text, "READONLY", "YES");
    IupSetAttribute(cs_text, "VISIBLELINES", "5");
    IupSetAttribute(cs_text, "VISIBLECOLUMNS", "10");
    IupSetAttribute(cs_text, "SCROLLBAR", "NO");
    eq_text = IupText(NULL);
    IupSetAttribute(eq_text, "READONLY", "YES");
    IupSetAttribute(eq_text, "VISIBLECOLUMNS", "20");
    fcastx_label = IupLabel("Значение X:");
    fcast_x = IupText(NULL);
    fcasty_label = IupLabel("Значение Y:");
    fcast_y = IupLabel("-");
    IupSetAttribute(fcast_y, "MINSIZE", "80x");
    fcast_button = IupButton("Вычислить", NULL);
    IupSetCallback(fcast_button, "ACTION", (Icallback) calc_forecast_value);
    fcast_b1 = IupHbox(fcastx_label, fcast_x, fcast_button, NULL);
    IupSetAttribute(fcast_b1, "GAP", "8");
    IupSetAttribute(fcast_b1, "ALIGNMENT", "ACENTER");
    fcast_b2 = IupHbox(fcasty_label, fcast_y, NULL);
    IupSetAttribute(fcast_b2, "GAP", "8");
    fcast = IupFrame(IupVbox(fcast_b1, fcast_b2, NULL));
    IupSetAttribute(fcast, "TITLE", "Прогнозное значение");
    pscale_eq = IupToggle("Одинаковый масштаб по X и Y", NULL);
    IupSetCallback(pscale_eq, "ACTION", (Icallback) set_scaleequal);
    xmin_label = IupLabel("X min:");
```

```
    xmin = IupText(NULL);
    xmax_label = IupLabel("X max:");
    xmax = IupText(NULL);
    ymin_label = IupLabel("Y min:");
    ymin = IupText(NULL);
    ymax_label = IupLabel("Y max:");
    ymax = IupText(NULL);
    IupSetCallback(xmin, "VALUECHANGED_CB", (Icallback) set_plot_bounds);
    IupSetCallback(xmax, "VALUECHANGED_CB", (Icallback) set_plot_bounds);
    IupSetCallback(ymin, "VALUECHANGED_CB", (Icallback) set_plot_bounds);
    IupSetCallback(ymax, "VALUECHANGED_CB", (Icallback) set_plot_bounds);
    auto_bounds = IupButton("Автоматически", NULL);
    IupSetCallback(auto_bounds, "ACTION", (Icallback) set_plot_bounds_auto);
    plot_pxy = IupGridBox(xmin_label, xmin, xmax_label, xmax, ymin_label, ymin, y
max_label, ymax, NULL);
    IupSetAttribute(plot_pxy, "SIZECOL", "1");
    IupSetAttribute(plot_pxy, "NUMDIV", "4");
    IupSetAttribute(plot_pxy, "GAPCOL", "8");
    IupSetAttribute(plot_pxy, "GAPLIN", "4");
    plot_params = IupFrame(IupVbox(pscale_eq, plot_pxy, auto_bounds, NULL));
    IupSetAttribute(plot_params, "GAP", "8");
    IupSetAttribute(plot_params, "TITLE", "Параметры графика");
    plotter = IupPlot();
    matrix_container = IupHbox(t_matrix, NULL);
    IupSetAttribute(matrix_container, "EXPAND", "YES");
    IupSetAttribute(matrix_container, "MARGIN", "4x");
    l_container = IupVbox(spin_container, matrix_container, NULL);
    IupSetAttribute(l_container, "GAP", "4");
    IupSetAttribute(l_container, "MARGIN", "4x4");
    IupSetAttribute(l_container, "EXPAND", "VERTICAL");
    m_container = IupVbox(rt_select, button, cs_text, eq_text, fcast, plot_params
, NULL);
    IupSetAttribute(m_container, "NGAP", "8");
    IupSetAttribute(m_container, "MARGIN", "4x4");
    IupSetAttribute(m_container, "NORMALIZESIZE", "HORIZONTAL");
    main_container = IupHbox(l_container, m_container, IupVbox(plotter, NULL), NU
LL);
    IupSetAttribute(main_container, "NGAP", "8");
    dlg = IupDialog(main_container);
    item_open = IupItem("Загрузить данные из файла", NULL);
    IupSetCallback(item_open, "ACTION", (Icallback) load_file);
    item_save = IupItem("Сохранить данные в файл", NULL);
    IupSetCallback(item_save, "ACTION", (Icallback) save_file);
    item_exit = IupItem("Выход", NULL);
    IupSetCallback(item_exit, "ACTION", (Icallback) app_exit);
    file_menu = IupMenu(item_open, item_save, item_exit, NULL);
    file_submenu = IupSubmenu("Файл", file_menu);
    item_about = IupItem("О программе", NULL);
    IupSetCallback(item_about, "ACTION", (Icallback) show_about);
    help_menu = IupMenu(item_about, NULL);
```

```
    help_submenu = IupSubmenu("Справка", help_menu);
    menu = IupMenu(file_submenu, help_submenu, NULL);
    IupSetHandle("menu", menu);
    IupSetAttribute(dlg, "MENU", "menu");
    IupSetAttribute(dlg, "MINSIZE", "1100x580");
    IupSetAttribute(dlg, "TITLE", "Регрессионный анализ");
    IupMap(dlg);
    IupShowXY(dlg, IUP_CENTER, IUP_CENTER);
    change_count(spin);
    IupUpdate(dlg);
    IupMainLoop();
    IupClose();
    return EXIT_SUCCESS;
}
```