多元回归分析C语言实现

```
#include <stdio.h>
#include <stdlib.h>
#include <malloc.h>

void free_data(double **dat,double *d,int count){
     int i;
     int j;
     free(d);
     for(i=0;i<count;i++){
         free(dat[i]);
     }
     free(dat);
}

//解线性方程,data[count*(count+1)]矩阵数组;count 方程元数
//answer[count] 求解数组. 返回:0 求解成功,-1 无解或者无穷解

int linear_equations(double *data,int count,double *answer){
    int j,m,n;
    double tmp,**dat,*d=data;
    dat=(double**)malloc(count*sizeof(double*));
    for(m=0;m<count;m++,d+=(count+1)){
        dat[m]=(double*)malloc((count+1)*sizeof(double));
        memcpy(dat[m],d,(count+1)*sizeof(double));
    }

    d=(double*)malloc((count+1)*sizeof(double));
    for(m=0;m<count-1;m++){
        for(n=m+1;n<count&&(dat[m][m]==0.0);n++){
            if(dat[n][m]!=0.0){
                memcpy(d,dat[m],(count+1)*sizeof(double));
                memcpy(dat[m],dat[n],(count+1)*sizeof(double));
                memcpy(dat[n],d,(count+1)*sizeof(double));
            }
        }

        //wujie
        if(dat[m][m]==0.0){
            printf("No result!\n");
            free_data(dat,d,count);
            return -1;
        }

        for(n=m+1;n<count;n++){
            tmp=dat[n][m]/dat[m][m];
            for(j=m;j<=count;j++){
                dat[n][j]-=tmp*dat[m][j];
            }
        }
    }


    for(j=0;j<count;j++){
        d[j]=0.0;
    }


    answer[count-1]=dat[count-1][count]/dat[count-1][count-1];

    for(m=count-2;m>=0;m--){
        for(j=count-1;j>m;j--)d[m]+=answer[j]*dat[m][j];

        answer[m]=(dat[m][count]-d[m])/dat[m][m];
    }

    free_data(dat,d,count);
    return 0;
}

//y=b0+b1x1+b2x2+...+bnxn
//data[rows*cols]: x1i,x2i...xni,yi
//answer[cols] 返回回归系数(b0,b1..bn);

int multiple_regression(double *data,int rows,int cols,double *answer,double *square_poor){
    int m,n,i,count=cols-1;
    double *dat,*p,a,b;
    if(data==0 || answer==0 || rows<2 || cols<2){
        return -1;
    }

    dat=(double*)malloc(cols*(cols+1)*sizeof(double));
    dat[0]=(double)rows;

    for(n=0;n<count;n++){
        a=b=0.0;
        for(p=data+n,m=0;m<rows;m++,p+=cols){
            a+=*p;
            b+=((*p) * (*p));
        }

        dat[n+1]=a;
        dat[(n+1)*(cols+1)]=a;
        dat[(n+1)*(cols+1)+n+1]=b;

        for(i=n+1;i<count;i++){
            for(a=0.0,p=data,m=0;m<rows;m++,p+=cols) a+=(p[n]*p[i]);

            dat[(n+1)*(cols+1)+i+1]=a;
            dat[(i+1)*(cols+1)+n+1]=a;
        }

    }

    for(b=0.0,m=0,p=data+n;m<rows;m++,p+=cols) b+=*p;

    dat[cols]=b;

    for(n=0;n<count;n++){
        for(a=0.0,p=data,m=0;m<rows;m++,p+=cols) a+=(p[n]*p[count]);

        dat[(n+1)*(cols+1)+cols]=a;
    }

    n=linear_equations(dat, cols, answer);

    if(n==0 && square_poor){
        b=b/rows;
        square_poor[0]=square_poor[1]=0.0;
        p=data;

        for(m=0;m<rows;m++,p++){

            for(i=1,a=answer[0];i<cols;i++,p++) a+=(*p * answer[i]);

            square_poor[0]+=((a-b)*(a-b));
            square_poor[1]+=((*p-a)*(*p-a));
        }

        square_poor[2]=square_poor[0]/count;
        if(rows-cols>0.0){
            square_poor[3]=square_poor[1]/(rows-cols);
        }else{
            square_poor[3]=0.0;
        }
    }

    free(dat);
    return n;

}



void display(double *dat,double *answer,double *square_poor,int rows,int cols){
    double v,*p;
    int i,j;
    printf("回归方程式: Y=%.5lf\n",answer[0]);

    for(i=1;i<cols;i++) printf("+%.5lf*X%d",answer[i],i);

    printf("\n");

    printf("回归显著性检验:\n");
    printf("回归平方和:%12.4lf 回归方差: %12.4lf\n",square_poor[0],square_poor[2]);
    printf("剩余平方和:%12.4lf 剩余方差: %12.4lf\n",square_poor[1],square_poor[3]);
    printf("离差平方和:%12.4lf 标准误差: %12.4lf\n",square_poor[0]+square_poor[1],sqrt(square_poor[3]));
    printf("F 检验: %12.4lf 相关系数:
%12.4lf\n",square_poor[2]/square_poor[3],sqrt(square_poor[0]/(square_poor[0]+square_poor[1])));
    printf("....\n");

    printf("剩余分析:\n");

    printf("    观察值      估计值    剩余值      剩余平方\n");
    for(i=0,p=dat;i<rows;i++,p++){
        v=answer[0];
        for(j=1;j<cols;j++,p++) v+=*p * answer[j];

        printf("%12.2lf%12.2lf%12.2lf%12.2lf\n",*p,v,*p-v,(*p-v)*(*p-v));

    }
    getchar();

}



double data[15][5]={
  { 316, 1536, 874, 981, 3894 },
  { 385, 1771, 777, 1386, 4628 },
  { 299, 1565, 678, 1672, 4569 },
  { 326, 1970, 785, 1864, 5340 },
  { 441, 1890, 785, 2143, 5449 },
  { 460, 2050, 709, 2176, 5599 },
  { 470, 1873, 673, 1769, 5010 },
  { 504, 1955, 793, 2207, 5694 },
  { 348, 2016, 968, 2251, 5792 },
  { 400, 2199, 944, 2390, 6126 },
  { 496, 1328, 749, 2287, 5025 },
  { 497, 1920, 952, 2388, 5924 },
  { 533, 1400, 1452, 2093, 5657 },
  { 506, 1612, 1587, 2083, 6019 },
  { 458, 1613, 1485, 2390, 6141 },

};
```

```
double data1[6][6]={
  { 3.7361, 13.1993, 0.2616, 179.1341, 672.1474,132},
  { 7.34,   19.5001, 0.1769, 261.0760, 389.3585,121},
  { 13.038, 26.3476, 0.1329, 356.0133, 280.0121,110},
  { 2.9974, 5.5954,  0.2916, 76.0206 , 1469.4660,107},
  { 1.6850, 4.9143,  0.3580, 61.1992 , 1515.0510,113},
  { 1.4004, 4.8400,  0.4047, 63.8586 , 1750.1650,120},

};
double data2[6][3]={
  { 213,1,2 },
  { 2131,2,3},
  { 5,3,231},
  {7666,4,5.5},
  {33,5,55},
  {78,6,23},

};

int main(int argc,char *argv[]) {
    printf("hello world\n");
    //double answer[5],square_poor[4];
    //if(multiple_regression((double* )data, 15, 5, answer, square_poor)==0){
        //display((double *) data, answer, square_poor, 15, 5);
    //}
    double answer[6],square_poor[4];
    if(multiple_regression((double* )data1, 6, 6, answer, square_poor)==0){
        display((double *) data1, answer, square_poor, 6, 6);
    }
    getchar();
    return 0;
}
```