Математическое моделирование управления процентными рисками в банке

Процентный риск - возможность убытков из-за неблагоприятных изменений процентных ставок. Как известно, процентные ставки по кредитам меняются раз в год, а по депозитам - раз в 3 месяца. Это несовпадение и приводит к процентному риску.

Сейчас банки обеспечивают свою платежеспособность по обязательствам собственным капиталом. Регулятор (Банк России) регламентирует, что капитал должен быть не меньше определенной доли от активов с учетом процентного риска (RWA, risk-weighted assets), и регулирует расчет RWA. Банки заинтересованы в минимальном капитале, и потому стараются снизить RWA.
Банк России устанавливает математическую модель для оценки RWA для имеющегося портфеля активов и пассивов, опираясь на два принципа: если актив и пассив близки между собой по сроку (разница не более 30 дней) и процентной ставке (разница не более 0,15% годовых), то в RWA входит только разница их сумм. Для оставшихся после «неттирования» остаточных активов и пассивов RWA считаются по формуле:

$$RWA = (\sum Актив + \sum Пассив) \cdot w_{gross} + |\sum Актив - \sum Пассив| \cdot w_{net}.$$

Это означает, что для минимизации RWA необходим алгоритм группировки сделок с учетом следующих условий:

1. ∑Актив и ∑Пассив в каждой «группе неттирования» должны быть примерно одинаковы, чтобы их разность была минимальна.
2. Сделки, которые не попали в «группы неттирования», должны быть такими активами и пассивами, что сумма и разность сумм всех оставшихся активов и пассивов были минимальны, то есть чтобы каждое слагаемое в формуле RWA было минимальным.

С учетом этих условий мы сразу приступили к рассмотрению Случая 2. Мы пользуемся данными файла (таблицы) “Данные_Мамонт_3мес” , размещенного на сайте турнира.

Оценка RWA сверху

Для начала рассмотрим оценим RWA сверху: сделки не группируются и не неттируются, а сразу подставляются в формулу RWA. В таком случае ∑Актив = 40.342.052.582 и ∑Пассив = 38.309.579.595, коэффициенты берутся из условия задачи ($w_{gross} = 0{,}1$, $w_{net} = 0{,}4$). Тогда RWA = 8.678.152.413 (≈**8,678 млрд**).

Оценка RWA снизу

Для оценки RWA снизу рассмотрим случай, когда все активы и пассивы в портфеле могут неттироваться между собой (при таком допущении мы получим недостижимый RWA из-за того, что в нашем портфеле не все активы и пассивы могут неттироваться).

Результат неттирования будет численно равен ∑Актив - ∑Пассив = 2.032.472.987 (значения ∑Актив и ∑Пассив рассчитаны ранее). Результат неттирования > 0, значит записываем его в ∑Актив и по формуле получаем

$RWA = (2.032.472.987 + 0) \cdot 0{,}1 + |2.032.472.987 - 0| \cdot 0{,}4 = 1.016.236.494 \approx$ **1,016 млрд**.

Модель 1

Для удобства работы с данными мы преобразовали их из такого вида:

Кредит (актив)	Вклад (пассив)	Дата погашения	Срок погашения, дней	Ставка, %
0	33000000	6/4/2018	74	2.0246
100000000	0	6/11/2018	81	2.0714
550000000	0	5/7/2018	46	1.7935
0	5000000	6/11/2018	81	2.0714
0	5000000	6/11/2018	81	2.0714
6100000	0	6/14/2018	84	2.1069
0	8200000	6/14/2018	84	2.1069
14400000	0	6/18/2018	88	2.1775

в такой:

0	33000000	74	2.0246
100000000	0	81	2.0714
550000000	0	46	1.7935
0	5000000	81	2.0714
0	5000000	81	2.0714
6100000	0	84	2.1069
0	8200000	84	2.1069
14400000	0	88	2.1775

Будем рассматривать постепенно усложняющиеся модели.

Первый шаг Модели 1 представляет из себя суммирование активов и пассивов, имеющих одинаковые ставки ***r*** (rate) и сроки погашения ***t*** (time). Таким образом, исходный портфель из 2175 сделок сводится к набору из 116, которым присваиваем значение ***s*** - разность соответствующих активов и пассивов. Следовательно, если s положительна, то считаем её активом, отрицательна - пассивом.

По формуле рассчитаем RWA для полученного набора, не неттируя входящие в него сделки.

Рассчитанное таким образом RWA равно $\approx$ **2 млрд**, поэтому мы считаем, что эти преобразования данных эффективны.

Вторым шагом задается плоскость с ортогональными осями **t** (ось сроков погашения, дней) и **r** (ось ставок, %). На этой плоскости размещаются точки, являющиеся группами сделок с одинаковыми t и r, и каждой точке присваивается соответствующее s из первого шага. Таким образом, получается заполненная точками из полученного набора плоскость. (см. Рис.1)

Заметим, что все точки располагаются в определенном прямоугольнике(назовем его *главным*), границы которого зависят от

минимальных и максимальных значений сроков погашений и ставок (в нашем случае это 32 и 92 дней и 0,8495 и 9,1250%).

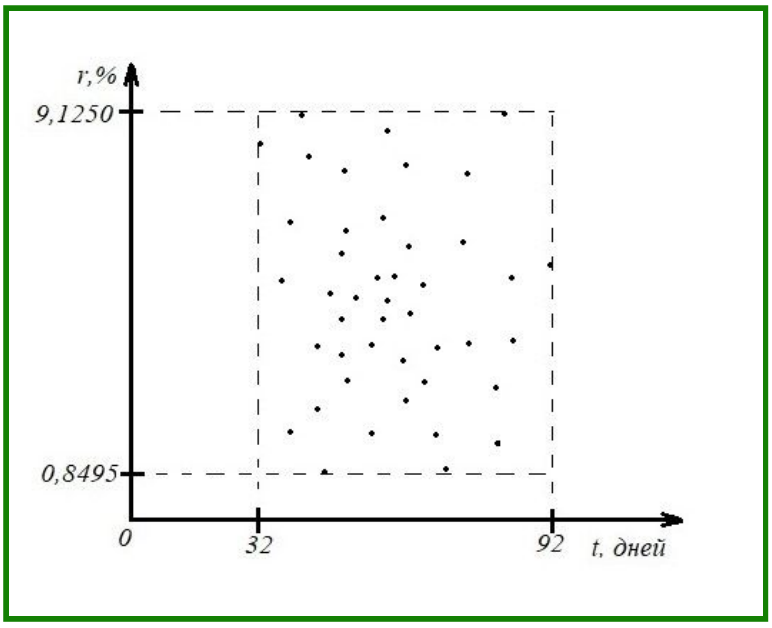

Рис.1

Далее задаем прямоугольник с максимально допустимыми сторонами t_0 = 30 дней и r_0 = 0,15%.

Располагаем такой прямоугольник в I четверти так, чтобы он пересекал главный прямоугольник (см. Рис.2).

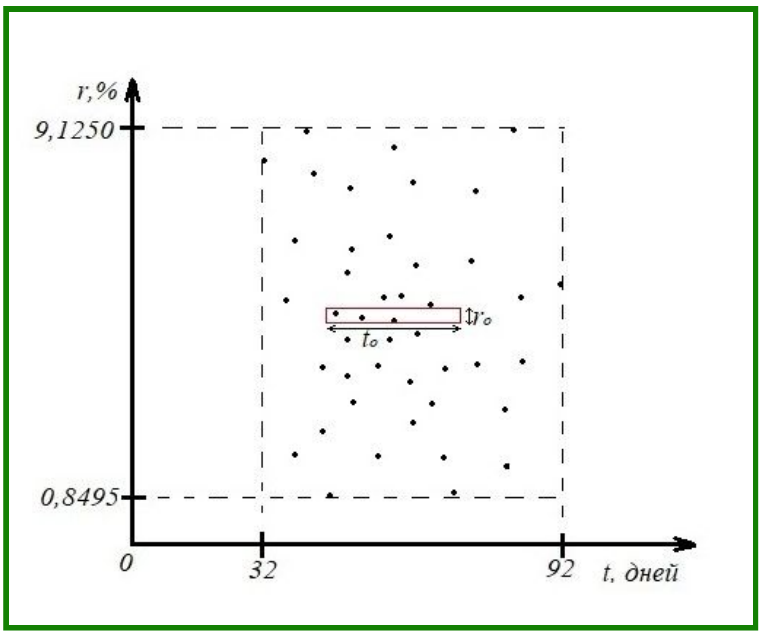

Рис.2

Все точки лежащие в прямоугольнике неттируются и результат записывается в

- ∑Актив, если результат неттирования положительный (> 0).

- ∑Пассив, если результат неттирования отрицательный (< 0). В этом случае берём модуль от этого результата и только тогда записываем в ∑Пассив.
- Нигде не учитывается, если равен нулю (= 0).

Неттированные точки, находившиеся в этом прямоугольнике, стираются. (см. Рис.3)

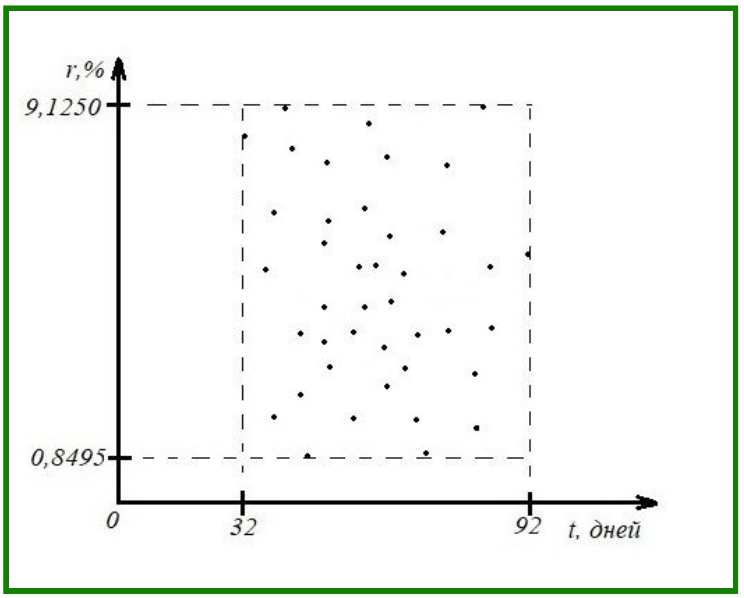

Рис.3

Далее плоскость аналогичным образом заполняется такими же прямоугольниками со сторонами t_0 и r_0 (см. Рис.4), и значения неттируются описанным ранее способом.

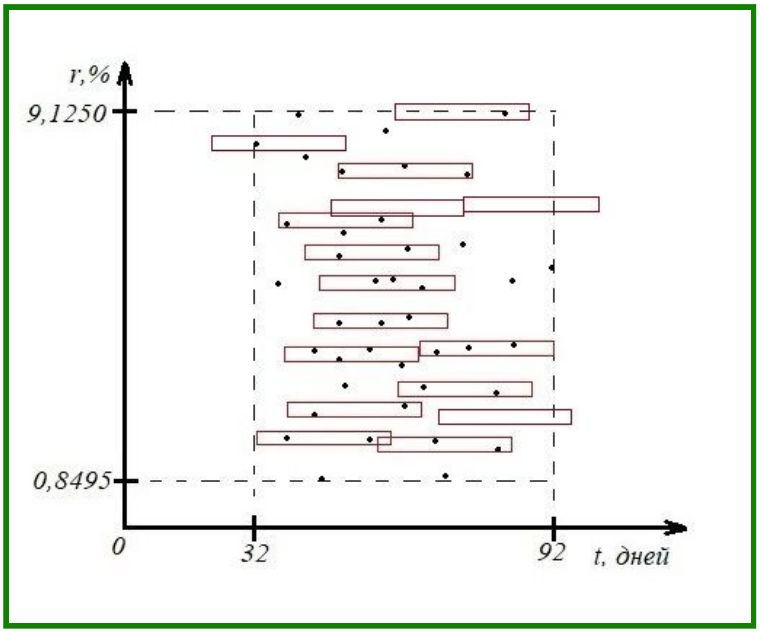

Рис.4

Для не попавших ни в один прямоугольник точек значения сразу записываются в ∑Актив и ∑Пассив.

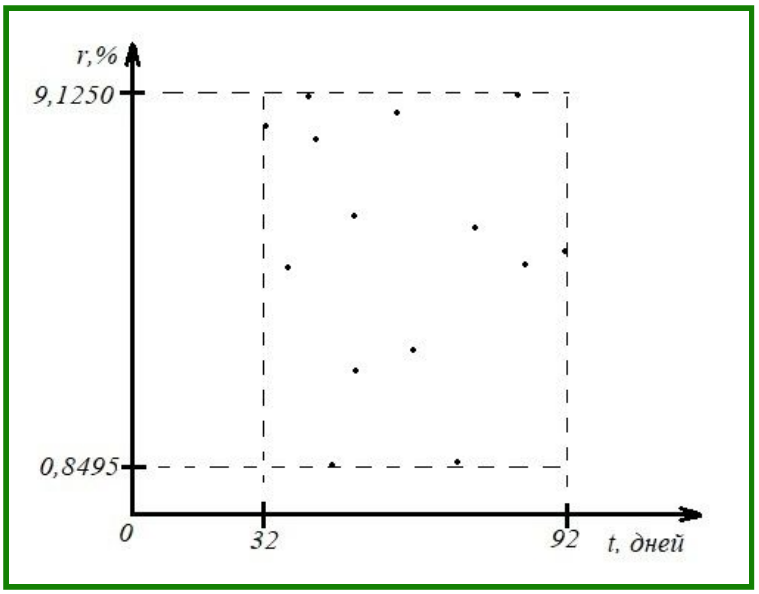

Рис.5

Далее рассчитывается RWA по формуле Банка России, в которую подставляются полученные ∑Актив и ∑Пассив и коэффициенты w_{gross} = 0,1, w_{net} = 0,4 . Мы много раз вычисляем RWA при случайном расположении 300 прямоугольников по этому алгоритму и берём минимум, который в нашем случае составил **1,091 млрд**.

Расчёт номеров производится следующим образом. В случае, если группа сделок попала в прямоугольник с номером N, мы присваиваем группе номер N. В случае, если точка не попала ни в один прямоугольник, мы присваиваем группе из этой точки минимальный незанятый номер. Таким образом, произошла группировка всех сделок. Всего мы разбили все сделки на 37 групп, нумерация групп начинается с 0. К статье приложена таблица с группировкой согласно этой модели.

В Приложении 1 представлена осуществляющая этот алгоритм программа. Эта программа также измеряет потраченное на расчет наименьшего полученного RWA время, для 2175 сделки оно составило 7 минут.

Для проверки правильности работы нашего алгоритма написана программа (см. Приложение 2), производящая неттирование находящихся в одной группе сделок. Запуск программы показал, что наш алгоритм работает верно.

Модель 2

В следующей модели мы задаем прямоугольник с нефиксированными сторонами t и r, где $\Delta t \in (0;30]$ и $\Delta r \in (0;0,15]$ (см. Рис.6).

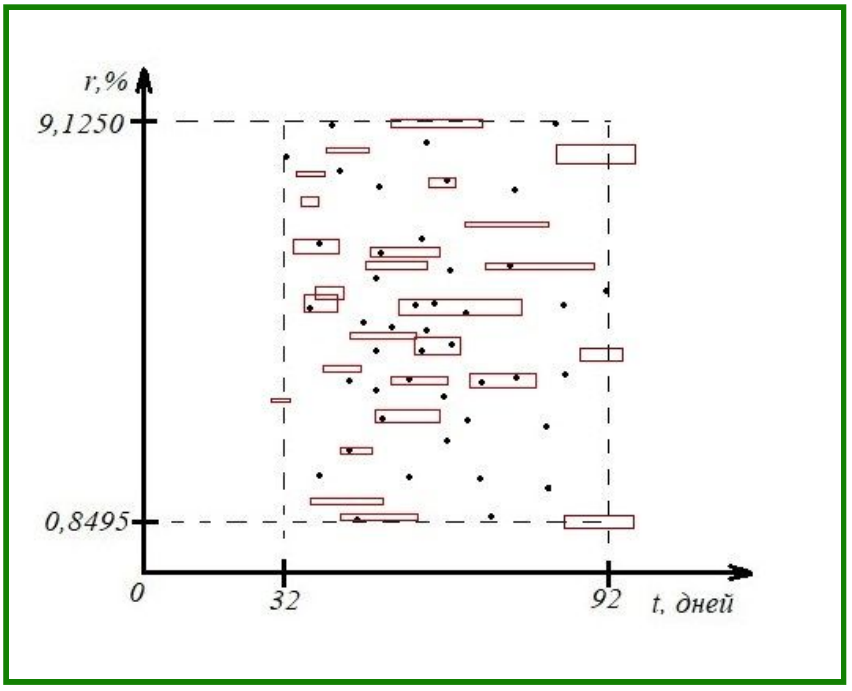

Рис.6

Для этой модели написана программа (см. Приложение 3). Однако, наилучший результат модели 2 (1,22 млрд) уступает наилучшему результату модели 1. Исходя из этого, мы посчитали целесообразным вернуться к алгоритму 1-ой модели.

Модель 3

Мы решили улучшить алгоритм Модели 1, чтобы уменьшить RWA. Для этого мы оптимизировали неттирование значений точек в прямоугольнике со сторонами t_0 и r_0.

Рассмотрим такой прямоугольник с точками внутри него (см. Рис.7). При неттировании значений этих точек мы получаем некоторое число ***d***.

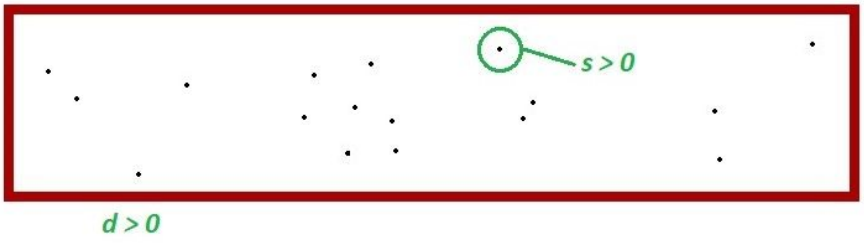

Рис.7

1. a) Если d > 0, то находим в прямоугольнике точку, значения которой $s > 0$, и вычитаем это значение из d. Если $d_1 = (d-s) > 0$, то снова находим точку, значение который $s_1 > 0$, и вычитаем это значения из $d_1 = (d-s)$.
   Продолжаем так делать пока $d_i > 0$ и затем неттируем все точки кроме s_1, s_2 … s_n.
   b) Если d < 0, то делаем все тоже самое, за исключением того, что $d_i < 0$ и $s_i < 0$.
2. После этого точки, значения которых вычитались, остаются в прямоугольнике, а остальные неттируются (см. Рис.8). и это значение записывается либо в ∑Актив, либо в ∑Пассив.

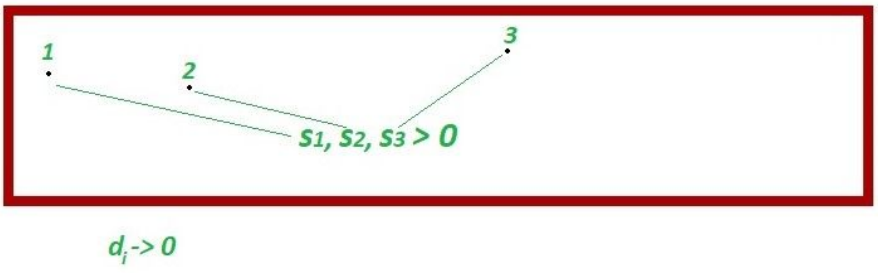

Рис.8

Для данного алгоритма написана программа (см. Приложение 4). Полученное по этому алгоритму RWA равно 1,090 млрд и отличается от результата первой модели на 0,09%, что не является существенным выигрышем. Учитывая этот факт и недостаток времени, мы решили не развивать эту модель и не писать код для разбиения сделок по группам в таблице.

Альтернативная модель на 2 временных промежутках

Рассмотрим альтернативную модель для решения задачи. В этой модели сначала разбиваем сделки на группы А (срок погашения 32-62 дня) и Б (срок погашения 63-92 дня). Как показывают численные эксперименты на модели 1, разбиение на два промежутка по времени наиболее оптимально, а это наиболее естественное такое разбиение.

Далее сортируем обе группы сделок по ставкам. Для этого портфеля сделок имеем, что в группе А минимальная ставка - 0,8495%, а максимальная - 9,125%.

В группе Б минимальная ставка - 0,934%, а максимальная - 7,75%. Мы рассматриваем 15 вариантов разбиения сделок в каждой группе по времени на интервалы по ставкам вида:
$[0{,}75+0{,}15 \cdot n+0{,}01 \cdot (i-1); 0{,}75+0{,}15 \cdot (n+1)+0{,}01 \cdot (i-1))$. Здесь n - номер интервала, i - номер разбиения, 0,75 - это стартовая точка (для группы Б она тоже равна 0,75, т.к. начиная с 5 разбиения, ставка 0,934% попадает в интервал [0,79; 0,94)).
Таким образом, группа А разбивается на 54 интервала для 15 разбиения и 55 интервалов для всех остальных разбиений, а группа Б - на 45 интервалов для 12-15 разбиений и 46 интервалов для прочих разбиений. Таким образом, всего у нас 99 интервалов для 15 разбиения, 100 интервалов для 12-14 разбиений, 101 интервал для остальных разбиений. Но из-за того, что у нас очень мало контрактов со ставками, большими, чем 3%, у нас в большинство интервалов в диапазоне (3%, 9%) не попадает ни один контракт. Поэтому для каждого разбиения у нас 31 непустой интервал.
По каждому интервалу проводится неттинг, затем по формуле рассчитывается RWA. В качестве итогового разбиения берется разбиение с минимальным RWA.
Для модели на двух промежутках проводилось два численных эксперимента - на обработанных данных (RWA_1) и на необработанных (RWA_2). Обработка данных проводилась в соответствии с шагом 1 Модели 1.
Все расчеты и операции проводились в программе Excel.
Рассчитанное RWA_1 - **1,17 млрд**.
Рассчитанное RWA_2 - **1,16 млрд**.
Изменения не более 1%.

Альтернативная модель на 3 временных промежутках

Мы полагали, что увеличение числа временных промежутков может оптимизировать нашу модель, поэтому решили поставить два численных эксперимента на обработанных и необработанных данных, разбитых по срокам погашения на группы А (32-50 дней), Б (53-71 день) и В (74-92 дня).
Разбиения по ставкам проводятся аналогично модели на двух промежутках.
Рассчитанное RWA - **1,32 млрд**, изменения для 2 экспериментов не более 0,5%.

Кроме того, эти эксперименты показывают, что разбиение на большее число промежутков по времени не приводит к оптимизации модели.
Поскольку Модель 1 показывает лучший результат, чем эта, в ответе указана группировка для Модели 1, но группировку для альтернативной модели легко восстановить, как и проверить её корректность.

Анализ моделей

Рассмотренные нами модели имеют как достоинства, так недостатки.

Опишем плюсы и минусы Моделей 1, 2, 3 и Альтернативной модели.

+	-
Модели 1, 2, 3	
Наглядна и проста для понимания. Работает для неограниченного количества сделок. Поддается оптимизации. Имеется готовая программа. Легко пересчитать результат для новых данных. Полученное значение RWA близко к оценке снизу	Используется перебор. Для 2175 сделок минимальный полученный RWA считается около 7 минут. Минимальность RWA не гарантирована.
Альтернативная модель	
Простая для понимания и реализации. Минимальный RWA считается за 10 секунд.	Решение менее оптимально, чем в моделях 1-3. Не поддается дальнейшему улучшению. Используется перебор из 15 разбиений. Программная реализация в Excel требует некоторой “подгонки” под данные.

Вывод

В результате мы получили наименьший RWA = **1,091 млрд**, полученный в Модели 1, разбив все сделки на **37 групп** неттирования.

Также стоит отметить нашу альтернативную модель, где наименьший RWA равен **1,17 млрд**. Этот результат был получен в модели на 2 временных промежутках, путём разбиения всех сделок на **31 группу** неттирования.

Подводя итоги, отметим, что:

- Оценка RWA сверху равна 8,678 млрд, а значит нам удалось **уменьшить RWA в 7,95 раза** (**на 7,587 млрд**).
- Оценка RWA снизу равна 1,016 млрд, а значит наш RWA на 7,38% больше недостижимого минимума (больше на 0,075 млрд).

Приложение 1 (C++)

```
#include <string>
#include <iostream>
#include <fstream>
#include <vector>
#include <cmath>
#include <ctime>
#include <cstdlib>
#include <chrono>
using namespace std;
using namespace std::chrono;
int delta = 0;

class Act{
  public:
  double moneyup;
  double moneydown;
  int days;
  double stav;
  int group = -1;
  Act(double x, double y, int e, double r){
    moneyup = x;
    moneydown = y;
    days = e;
    stav = r;
  }
};

void clear_a(vector<Act> &deals){
  int res = deals.size();
  for(int i = 0; i < res; i++){
    for(int j = i+1; j < res; j++){
      if((deals[i].days == deals[j].days) && (deals[i].stav == deals[j].stav)){
        deals[i].moneyup = deals[i].moneyup + deals[j].moneyup;
        deals[i].moneydown = deals[i].moneydown + deals[j].moneydown;
        deals.erase(deals.begin()+j);
        j--;
        res--;
      }
    }
  }
```

```
}

double updates(vector<Act> &deals, int x, double y, int dx, double dy){
   double plus = 0;
   double minus = 0;
   vector<pair<int,double>> ti;
   for(int g = 0; g < deals.size(); g++){
      if(((deals[g].days >= x) && (deals[g].days <= x + dx)) && ((deals[g].stav >= y ) && (deals[g].stav <= y + dy))){
         delta = 1;
         plus = plus + deals[g].moneyup;
         minus = minus + deals[g].moneydown;
         deals.erase(deals.begin()+g);
         g--;
      }
   }
   return plus - minus;
}

void add_points(int x, double y, vector<Act> &deals, int now){
   for(int g = 0; g < deals.size(); g++){
      if(((deals[g].days >= x) && (deals[g].days <= x + 30)) && ((deals[g].stav >= y ) && (deals[g].stav <= y + 0.15)) &&
(deals[g].group == -1)){
         deals[g].group = now;
      }
   }
}

 int main(){
   high_resolution_clock::time_point t1 = high_resolution_clock::now();
    srand(unsigned(time(0)));
    vector<vector<pair<int, double>>> groupes;
    ifstream in;
    vector<string> q;
    vector<Act> deals;
    vector<Act> dealss;
    vector<pair<int,double>> now;
    vector<pair<int,double>> nowmin;
    in.open("date.csv");
    //В файле date.csv находятся данные в преобразованном ввиде согласно началу модели 1
    int n = 2175;
    string k;
    int d;
```

```
double a,b,c;
for(int i = 0; i < n; i++){
    in >> k;
    int s = 0;
    for(int j = 0; j < k.size(); j++){
        if((k[j] == ',')){
            q.push_back(k.substr(s,j-s));
            s = j+1;
        }
        if((j == k.size()-1)){
            q.push_back(k.substr(s,j-s+1));
        }
    }
    deals.push_back(Act(stod(q[0]), stod(q[1]), stoi(q[2]), stod(q[3])));
    q.clear();
}
in.close();
ofstream out;
out.open("test.csv");

dealss = deals;
clear_a(deals);
for(int i = 0; i < deals.size(); i++){
   out << deals[i].moneyup << ","<< deals[i].moneydown << ","<< deals[i].days << ","<< deals[i].stav <<endl;
}
double ac = 0;
double pa = 0;
double min = 1e100;
vector<Act> save = deals;

for(int i = 0; i < 2000000; i++){
   ac = 0;
   pa = 0;
   for(int j = 0; j < 300; j++){
       double da = rand()%90;
       double sta = (double(rand()%10000)/1000);
       double ka = updates(deals,da, sta, 30, 0.15);
       if(delta != 0){
           now.push_back(make_pair(da,sta));
           delta = 0;
       }
       if(ka > 0){
```

```
            ac = ac + ka;
        }else{
            pa = pa + abs(ka);
        }
    }
    for(int i = 0; i < deals.size(); i++){
        double qw = deals[i].moneyup-deals[i].moneydown;
        if(qw > 0){
            ac = ac + qw;
        }else{
            if(qw < 0){
                pa = pa + abs(qw);
            }
        }
    }
    if(min > (( 0.1 * (ac + pa)) + ( 0.4 * (abs(ac-pa))))){
        min = (( 0.1 * (ac + pa)) + ( 0.4 * (abs(ac-pa))));
        nowmin = now;
        groupes.clear();
        for(int i = 0; i < deals.size(); i++){
            vector<pair<int,double>> ss;
            ss.push_back(make_pair(deals[i].days,deals[i].stav));
            groupes.push_back(ss);
            }
    }
    now.clear();
    deals = save;
}

for(int i = 0; i < nowmin.size(); i++){
    add_points(nowmin[i].first, nowmin[i].second, save,i);
}

int last = nowmin.size();
for(int i = 0; i < save.size(); i++){
    if(save[i].group == -1){
        save[i].group = last;
        last++;
    }
}

ofstream ban;
```

```
    ban.open("result.csv");
    //В файл result.csv заносятся разбиение на группы и файл преобразован как в модели 1
    for(int i = 0; i < save.size(); i++){
        for(int j = 0; j < dealss.size(); j++){
            if((save[i].days == dealss[j].days) && ( save[i].stav == dealss[j].stav)){
                dealss[j].group = save[i].group;
            }
        }
    }

    for(int i = 0; i < dealss.size(); i++){
        ban << dealss[i].moneyup << "," << dealss[i].moneydown << "," << dealss[i].days << "," << dealss[i].stav << ","
<< dealss[i].group << endl;
    }

    ban.close();
    high_resolution_clock::time_point t2 = high_resolution_clock::now();
        auto duration = duration_cast<milliseconds>(t2-t1).count();
    cout << min << " "<< duration;
    return 0;
}
```

Приложение 2 (C++)

```
#include <string>
#include <iostream>
#include <fstream>
#include <vector>
#include <cmath>
#include <ctime>
#include <cstdlib>
using namespace std;
int delta = 0;

class Act{
    public:
    double moneyup;
    double moneydown;
    int days;
    double stav;
    int group = -1;
    Act(double x, double y, int e, double r,int g){
```

```
        moneyup = x;
        moneydown = y;
        days = e;
        stav = r;
        group = g;
    }
};


 int main(){
    double A[100];
    vector<Act> deals;
    vector<string> q;
    for(int i = 0; i < 100; i++){
        A[i] = 0;
    }
     ifstream in;
     in.open("result.csv");
     //result.csv - файл созданный в ходе работы программы их приложения 1
     int n = 2175;
     string k;
     int d;
     double a,b,c;
     for(int i = 0; i < n; i++){
         in >> k;
         int s = 0;
         for(int j = 0; j < k.size(); j++){
             if((k[j] == ',')){
                 q.push_back(k.substr(s,j-s));
                 s = j+1;
             }
             if((j == k.size()-1)){
                 q.push_back(k.substr(s,j-s+1));
             }
         }
         deals.push_back(Act(stod(q[0]), stod(q[1]), stoi(q[2]), stod(q[3]), stoi(q[4])));
         q.clear();
     }
    in.close();
    ofstream out;
    out.open("test.csv");
```

```
    for(int i = 0; i < deals.size(); i++){
        int kd = deals[i].group;
        A[kd] = A[kd] + deals[i].moneyup;
        A[kd] = A[kd] - deals[i].moneydown;
    }

    double ac = 0;
    double pa = 0;
    for(int i = 0; i < 100; i++){
        if(A[i] > 0){
            ac = ac + A[i];
        }
        if(A[i] < 0){
            pa = pa + abs(A[i]);
        }
    }

    cout << (( 0.1 * (ac + pa)) + ( 0.4 * (abs(ac-pa))));
    return 0;
}
```

Приложение 3 (C++)

```
#include <string>
#include <iostream>
#include <fstream>
#include <vector>
#include <cmath>
#include <ctime>
#include <cstdlib>
using namespace std;

class Act{
    public:
    double moneyup;
    double moneydown;
    int days;
    double stav;
    int group = -1;
    Act(double x, double y, int e, double r){
        moneyup = x;
        moneydown = y;
        days = e;
```

```
        stav = r;
    }
};

void clear_a(vector<Act> &deals){
    int res = deals.size();
    for(int i = 0; i < res; i++){
        for(int j = i+1; j < res; j++){
            if((deals[i].days == deals[j].days) && (deals[i].stav == deals[j].stav)){
                deals[i].moneyup = deals[i].moneyup + deals[j].moneyup;
                deals[i].moneydown = deals[i].moneydown + deals[j].moneydown;
                deals.erase(deals.begin()+j);
                j--;
                res--;
            }
        }
    }
}

double updates(vector<Act> &deals, int x, double y, int dx, double dy){
    double plus = 0;
    double minus = 0;
    for(int g = 0; g < deals.size(); g++){
        if(((deals[g].days >= x) && (deals[g].days <= x + dx)) && ((deals[g].stav >= y ) && (deals[g].stav <= y + dy))){
            plus = plus + deals[g].moneyup;
            minus = minus + deals[g].moneydown;
            deals.erase(deals.begin()+g);
            g--;
        }
    }
    return plus - minus;
}

void add_points(int x, double y, vector<Act> &deals, int now){
    for(int g = 0; g < deals.size(); g++){
        if(((deals[g].days >= x) && (deals[g].days <= x + 30)) && ((deals[g].stav >= y ) && (deals[g].stav <= y + 0.15)) &&
(deals[g].group == -1)){
            deals[g].group = now;
        }
    }
}
```

```
int main(){
   srand(unsigned(time(0)));
   ifstream in;
   vector<string> q;
   vector<Act> deals;
   in.open("date.csv");
   int n = 2175;
   string k;
   int d;
   double a,b,c;
   for(int i = 0; i < n; i++){
      in >> k;
      int s = 0;
      for(int j = 0; j < k.size(); j++){
         if((k[j] == ',')){
            q.push_back(k.substr(s,j-s));
            s = j+1;
         }
         if((j == k.size()-1)){
            q.push_back(k.substr(s,j-s+1));
         }
      }
      deals.push_back(Act(stod(q[0]), stod(q[1]), stoi(q[2]), stod(q[3])));
      q.clear();
   }
   in.close();
   clear_a(deals);
   double ac = 0;
   double pa = 0;
   double min = 1e100;
   vector<Act> save = deals;
         for(int i = 0; i < 20000; i++){
            ac = 0;
            pa = 0;
            for(int j = 0; j < 300; j++){
               int da = rand()%90;
               double sta = (double(rand()%10000)/1000);
               double sizex = rand()%30 + 1;
               double sizey = (double(rand()%1200 + 300))/10000;
               double ka = updates(deals,da, sta, sizex, sizey);
               if(ka > 0){
                  ac = ac + ka;
```

```
            }else{
                pa = pa + abs(ka);
            }
        }
        for(int i = 0; i < deals.size(); i++){
            double qw = deals[i].moneyup-deals[i].moneydown;
            if(qw > 0){
                ac = ac + qw;
            }else{
                if(qw < 0){
                    pa = pa + abs(qw);
                }
            }
        }
        if(min > (( 0.1 * (ac + pa)) + ( 0.4 * (abs(ac-pa))))){
            min = (( 0.1 * (ac + pa)) + ( 0.4 * (abs(ac-pa))));
        }
        deals = save;
    }

  cout << min;
  return 0;
}
```

Приложение 4 (C++)

```
#include <string>
#include <iostream>
#include <fstream>
#include <vector>
#include <cmath>
#include <ctime>
#include <cstdlib>
#include <algorithm>
using namespace std;
class Act{
  public:
  double moneyup;
  double moneydown;
  int days;
  double stav;
  int group = -1;
```

```
    Act(double x, double y, int e, double r){
        moneyup = x;
        moneydown = y;
        days = e;
        stav = r;
    }
};

void clear_a(vector<Act> &deals){
    int res = deals.size();
    for(int i = 0; i < res; i++){
        for(int j = i+1; j < res; j++){
            if((deals[i].days == deals[j].days) && (deals[i].stav == deals[j].stav)){
                deals[i].moneyup = deals[i].moneyup + deals[j].moneyup;
                deals[i].moneydown = deals[i].moneydown + deals[j].moneydown;
                deals.erase(deals.begin()+j);
                j--;
                res--;
            }
        }
    }
}

double updates(vector<Act> &deals, int x, double y, int dx, double dy){
    double plus = 0;
    double minus = 0;
    vector <int> del;
    for(int g = 0; g < deals.size(); g++){
        if(((deals[g].days >= x) && (deals[g].days <= x + dx)) && ((deals[g].stav >= y ) && (deals[g].stav <= y + dy))){
            plus = plus + deals[g].moneyup;
            minus = minus + deals[g].moneydown;
            del.push_back(g);
        }
    }
    double d = plus - minus;
    double z;
    if(d > 0){
        for(int i = 0; i < del.size(); i++){
            z = deals[del[i]].moneyup - deals[del[i]].moneydown;
            if((z > 0) && (d - z > 0)){
                d = d- z;
                del.erase(del.begin() + i);
```

```
            i--;
          }
        }
      }

      if(d < 0){
        for(int i = 0; i < del.size(); i++){
          z = deals[del[i]].moneyup - deals[del[i]].moneydown;
          if((z < 0) && (d - z < 0)){
            d = d- z;
            del.erase(del.begin() + i);
            i--;
          }
        }
      }

      sort(del.begin(), del.end());
      reverse(begin(del),end(del));
      for(int i = 0; i < del.size(); i++){
        deals.erase(deals.begin() + del[i]);
      }
      return d;
}

 int main(){
     srand(unsigned(time(0)));
     ifstream in;
     vector<string> q;
     vector<Act> deals;
     in.open("date.csv");
     int n = 2175;
     string k;
     int d;
     double a,b,c;
     for(int i = 0; i < n; i++){
        in >> k;
        int s = 0;
        for(int j = 0; j < k.size(); j++){
          if((k[j] == ',')){
            q.push_back(k.substr(s,j-s));
            s = j+1;
          }
```

```
        if((j == k.size()-1)){
            q.push_back(k.substr(s,j-s+1));
        }
    }
    deals.push_back(Act(stod(q[0]), stod(q[1]), stoi(q[2]), stod(q[3])));
    q.clear();
}
in.close();

clear_a(deals);
double ac = 0;
double pa = 0;
double min = 1e100;
vector<Act> save = deals;

for(int i = 0; i < 20000; i++){
   ac = 0;
   pa = 0;
   for(int j = 0; j < 300; j++){
      int da = rand()%90;
      double sta = (double(rand()%10000)/1000);
      double ka = updates(deals,da, sta, 30, 0.15);
      if(ka > 0){
         ac = ac + ka;
      }else{
         pa = pa + abs(ka);
      }
   }
   for(int i = 0; i < deals.size(); i++){
      double qw = deals[i].moneyup-deals[i].moneydown;
      if(qw > 0){
         ac = ac + qw;
      }else{
         if(qw < 0){
            pa = pa + abs(qw);
         }
      }
   }
   if(min > (( 0.1 * (ac + pa)) + ( 0.4 * (abs(ac-pa))))){
      min = (( 0.1 * (ac + pa)) + ( 0.4 * (abs(ac-pa))));
   }
   deals = save;
```

```
    }

    cout << min;

    return 0;
}
```