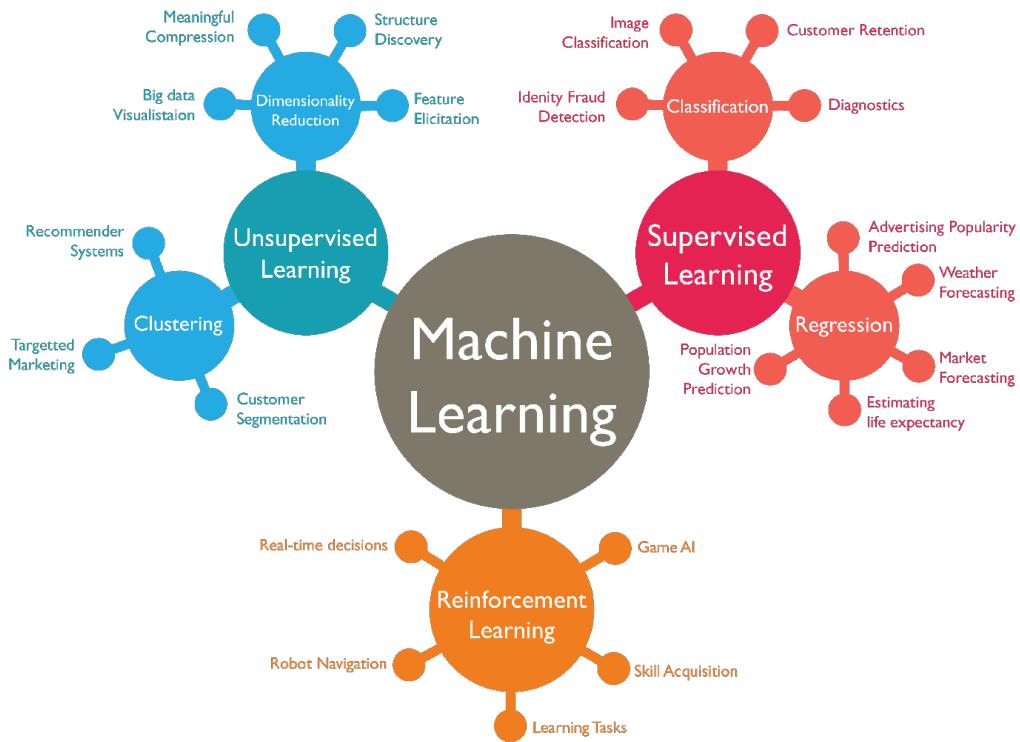


МЕТОДЫ ОПТИМИЗАЦИИ

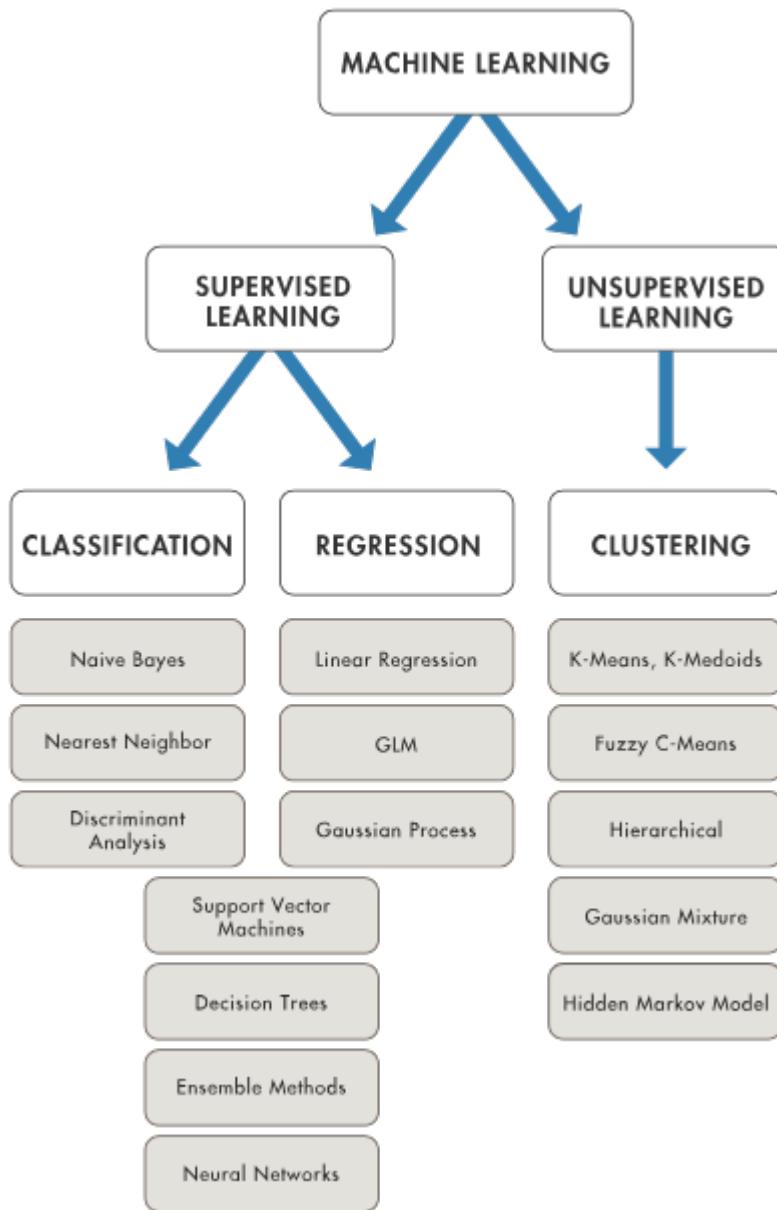
Семинар 9. Линейная регрессия

В терминах машинного обучения, регрессия - это "supervised learning"



(<https://github.com/shanmukh05/Machine-Learning-Roadmap>)

youtube: AI Warehouse - reinforcement learning)



(<https://www.mathworks.com/solutions/machine-learning.html>)

Задача линейной регрессии в обозначениях статистики

В предыдущем семинаре был вектор \vec{e} - вектор, который лежит в нуль-пространстве матрицы A^T , это та часть вектора \vec{b} , которая не может быть описана в рамках модели $A \vec{x} = \vec{b}$.

$\vec{e} = \vec{b} - \vec{p}$, где $\vec{p} = AA^\dagger \vec{b}$ - наиболее близкий (в терминах квадратичного отклонения) к \vec{b} вектор, лежащий в пространстве столбцов матрицы A . Если \vec{b} изначально лежит в пространстве столбцов $C(A)$ (и все столбцы линейнонезависимы), то $AA^\dagger \vec{b} = \vec{b}$, наша модель описывает данные полностью, без ошибки.

Теперь появляется случайная составляющая, вектор \vec{e} будет иметь случайную природу.

Будем пользоваться принятыми в статистике обозначениями.

Теперь: $\vec{b} \rightarrow \vec{y}, \vec{a} \rightarrow \vec{\beta}, \vec{a}^* \rightarrow \vec{b}, \vec{t} \rightarrow \vec{x}, A \rightarrow X$

\vec{Y} - зависимая переменная,

X - матрица независимых переменных (предикторов), в ней первый столбец - единичный, он учитывает постоянное смещение, это, так называемая **dummy variable**

$\vec{\beta}$ - это истинные значения параметров модели,

$$\vec{Y} = X \vec{\beta} + \vec{\epsilon}$$

$\vec{\epsilon}$ - случайный вектор ошибки.

Основное предположение, что $\mathbb{E}[\vec{\epsilon}] = \vec{0}$, то есть ошибка является несмещенной.

Например, $\vec{Y} = [I, X][\begin{matrix} \vec{\beta}_0 \\ \vec{\beta}_1 \end{matrix}] + \vec{\epsilon}$ - для линейной регрессии первого порядка, I - вектор, состоящий из единиц (в отличие от единичной матрицы I)

В выражении для выходное переменной, случайной является только вектор ошибки

Производя регрессионный анализ (обучение регрессионной модели) мы получаем вектор \vec{b}

$$\vec{b} = (X^T X)^{-1} X^T \vec{Y} = X^\dagger \vec{Y}$$

\vec{b} - вектор коэффициентов регрессионной модели, которые отличаются от истинных коэффициентов

$\vec{\beta}$ из-за наличия погрешности в исходных данных, вектор \vec{b} является оценкой истинного набора параметров нашей модели.

$$\vec{b} = X^\dagger \vec{Y} = X^\dagger X \vec{\beta} + X^\dagger \vec{\epsilon} = \vec{\beta} + X^\dagger \vec{\epsilon} \quad (\text{так как } X^\dagger X = (X^T X)^{-1} X^T X = I)$$

Так как $\mathbb{E}[\vec{\epsilon}] = \vec{0}$, то $\mathbb{E}[\vec{b}] = \vec{\beta}$, то есть, проведя, достаточно количество регрессий мы в среднем получим истинные параметры.

Матрица $H = X(X^T X)^{-1} X^T$ - это "hat-matrix", потому что она надевает шапку на вектор \vec{Y}

$\hat{Y} = HY = X(X^T X)^{-1} X^T = X \hat{b}$, соответственно \hat{Y} - это значения зависимой переменной, рассчитанные в рамках нашей модели. После обучения, мы можем использовать полученную модель для предсказаний, то есть, мы можем использовать полученные параметры регрессий для расчета вектора \hat{Y}_1 который будет соответствовать новым значениям предикторов X_1

Попробуем это в эксперименте на простом примере линейной регрессии с двумя параметрами (фитинг прямой линией)

```

clearvars
N=10; % размер выборки
tests_number = 76; % количество тестов
beta = [2; 5]; % истинные значения параметров
X1 = linspace(-1,1,N)';
X = [ones(N,1),X1]; % матрица предикторов
distrib = 'Normal'; % выбираем тип генератора случайных чисел
delta = 0.973; % параметр дисперсии ошибки
b_mat = zeros(tests_number,2); % матрица, в которой мы будем сохранять результаты
% тестов
Yo = X*beta;
Y_points = repmat(Yo,[1 tests_number]);
for ii=1:tests_number
    pd = make_dist(distrib,mu=0,sig=delta^2); %генерит генератор случайных чисел
    Y = Yo + delta*pd(N); % генерим данные как истинное значение модели + случайная
    % ошибка
    %std(Y) % стандартное отклонение примерно равно заданной амплитуде ошибки
    b_mat(ii,:) = transpose(X\Y);
    Y_points(:,ii) = Y;
end
% табличка для отображения результатов
tb = table(beta(:,1),mean(b_mat)',std(b_mat)', ... % вторая колонка таблицы - среднее
            % значения коэффициентов по всем испытаниям
            'VariableNames', ...
            ["Истинные значения","Средние ", ...
            "Стандартное отклонение"], ...
            'RowNames', ...
            ["b_0" "b_1"])
```

`tb = 2x3 table`

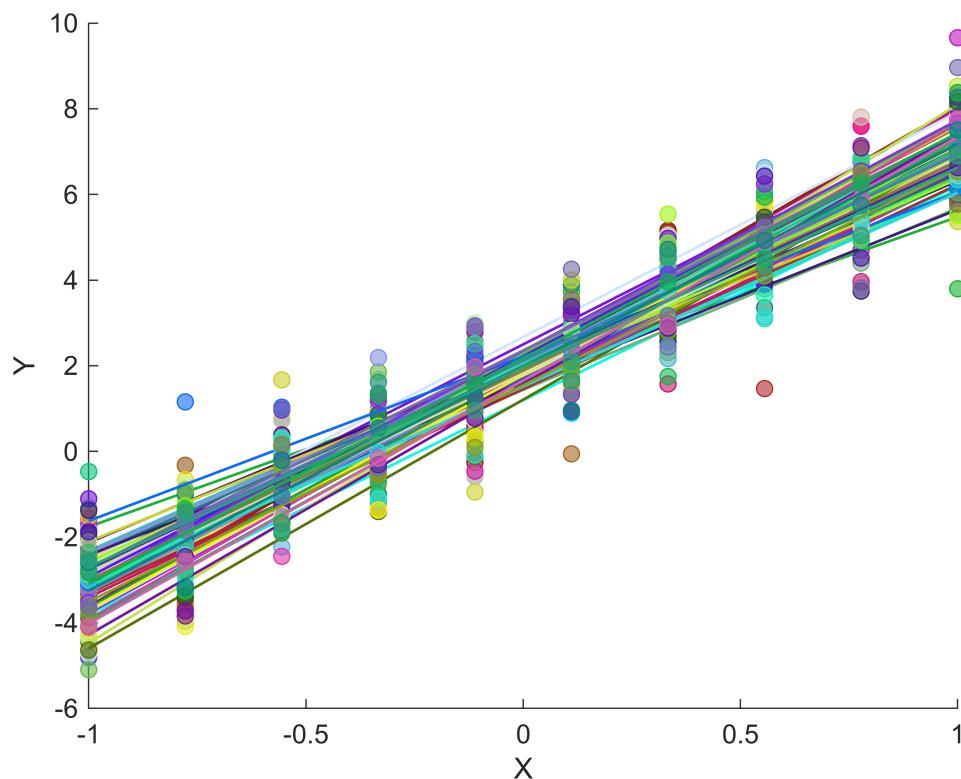
	Истинные значения	Средние	Стандартное отклонение
1 b_0	2	1.9401	0.2858
2 b_1	5	5.0113	0.4916

`%% графики`

```
disp("Точки - данные по которым производилась регрессия, каждый цвет - свой тест,
прямые - результат фитинга")
```

Точки - данные по которым производилась регрессия, каждый цвет - свой тест, прямые - результат фитинга

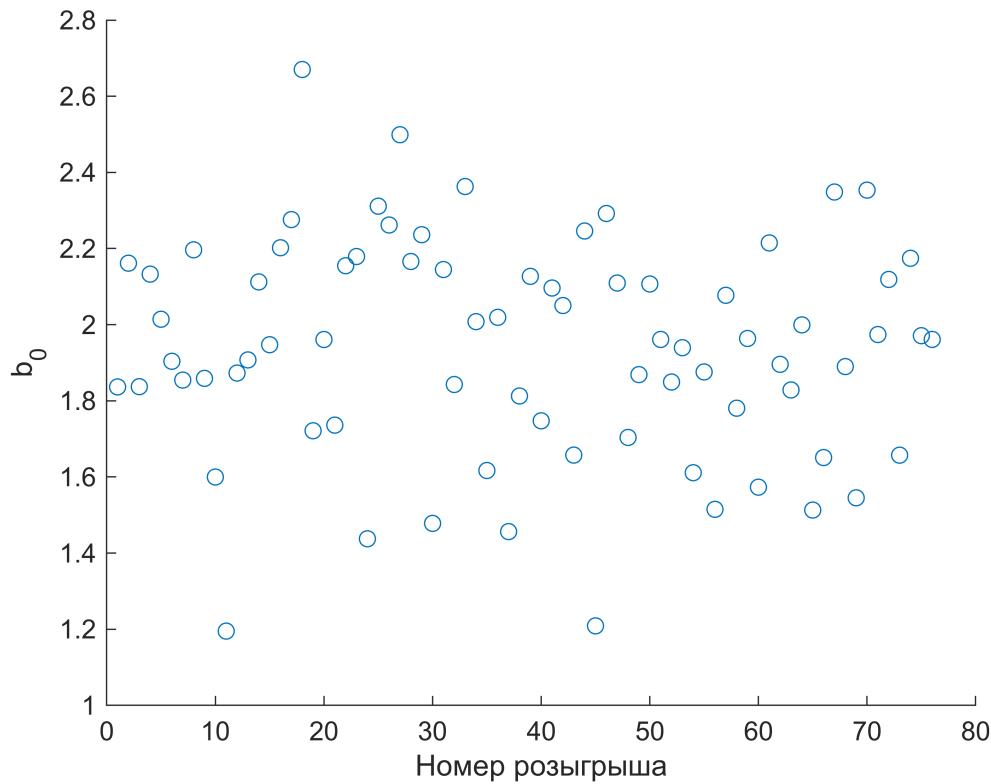
```
fitted_distr_array = arrayfun(@(jj)fitdist(b_mat(:,jj),"Normal"),1:2);
ax = get_next_ax();
iteration_step = 1;% строим каждый
inds = 1:iteration_step:tests_number;% будем шагать по номерам тестов
Cmat = rand([numel(inds),3]);% матрица цветов
X1 = X(:,2);
hold(ax,"on")
for ii=inds
    Y = Y_points(:,ii); % генерим данные как истинное значение модели + случайная
    %ошибка
    %std(Y) % стандартное отклонение примерно равно заданной амплитуде ошибки
    p1 = scatter(ax,X1,Y);
    p1.MarkerEdgeColor = Cmat(1+(ii-1)/iteration_step,:);
    p1.MarkerFaceColor = Cmat(1+(ii-1)/iteration_step,:);
    p1.MarkerFaceAlpha = 0.6;
    Ycalc = X*b_mat(ii,:)';
    p1 = plot(ax,X1,Ycalc,"LineWidth",1);
    p1.Color = Cmat(1+(ii-1)/iteration_step,:);
end
xlabel("X")
ylabel("Y")
hold(ax,"off")
```



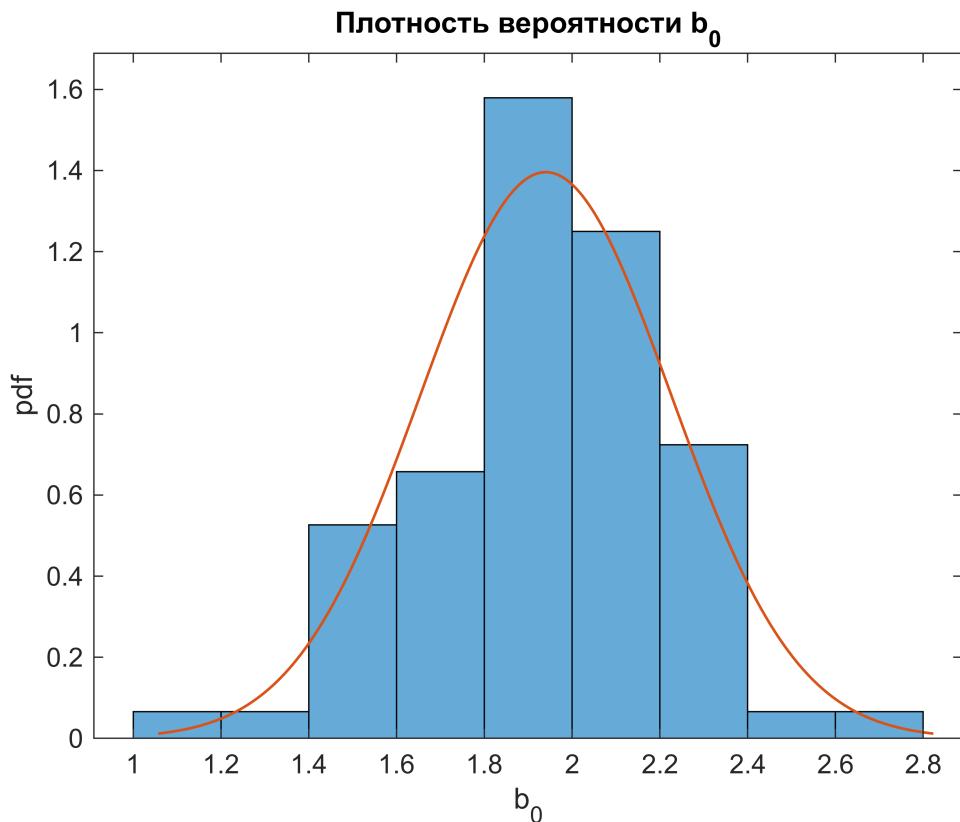
% выбираем параметр, для которого отображается распределение

```
% фитинг
```

```
selection = 1;
scatter(1:tests_number,b_mat(:,selection))
b_str = "b_" + string(selection-1);
ylabel(b_str); xlabel("Номер розыгрыша")
```



```
ax = get_next_ax();
fitted_distr = fitted_distr_array(selection); % фитим распределения методом из
statistical toolbox
plot(ax,fitted_distr)
title(ax,"Плотность вероятности "+ b_str)
xlabel(ax,b_str);
ylabel(ax,"pdf");
```

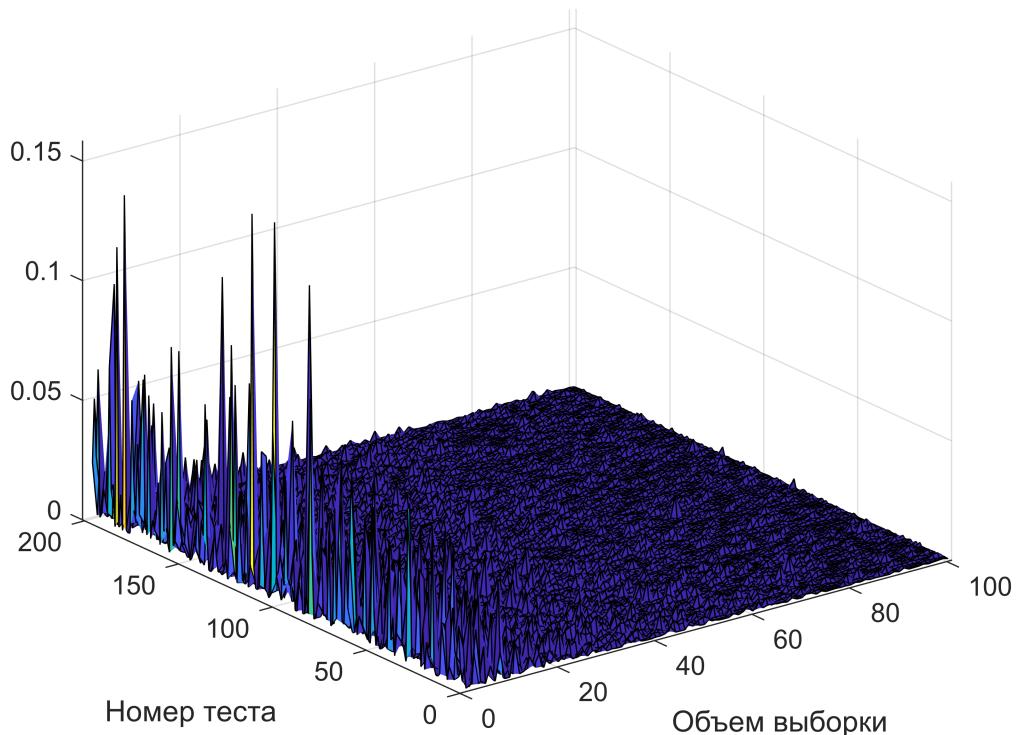


Тут важный момент, что для ошибки мы сделали только одно предположение, что она несмещенная, мы не полагали конкретной формы распределения.

Среднее по тестам значение нашей оценки параметров модели равно истинному значению

```
[sampling_volume,tests_number,beta_mat] = sampling_surf_plot(beta,101,201,0.2);
```

```
% изучим влияние объема выборки на вариацию параметров
b_mat_selected = beta_mat(:,:,selection);
selection_var= (b_mat_selected - beta(selection)).^2;
bmin = min(b_mat_selected,[],'all');
bmax = max(b_mat_selected,[],'all');
ax = get_next_ax();
[Xsurf,Ysurf]= meshgrid(sampling_volume(:),tests_number);
surf(ax,Xsurf,Ysurf,selection_var');%"LevelList",linspace(bmin,bmax,5))
xlabel("Объем выборки")
ylabel("Номер теста")
```

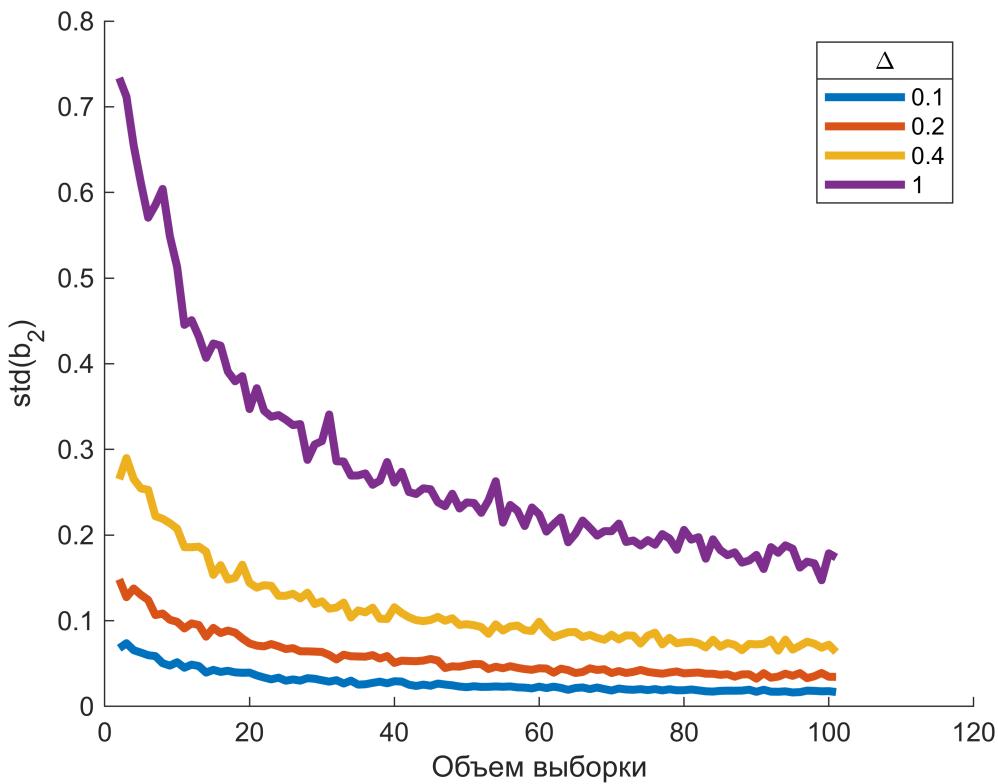


```
% строим зависимость вариации параметров модели от амплитуды ошибки
selection = 2;
b_str = "b_" + string(selection);
ax = get_next_ax();
delta = [0.1,0.2,0.4,1]

delta = 1×4
    0.1000    0.2000    0.4000    1.0000

b_std = cell(numel(delta),1);

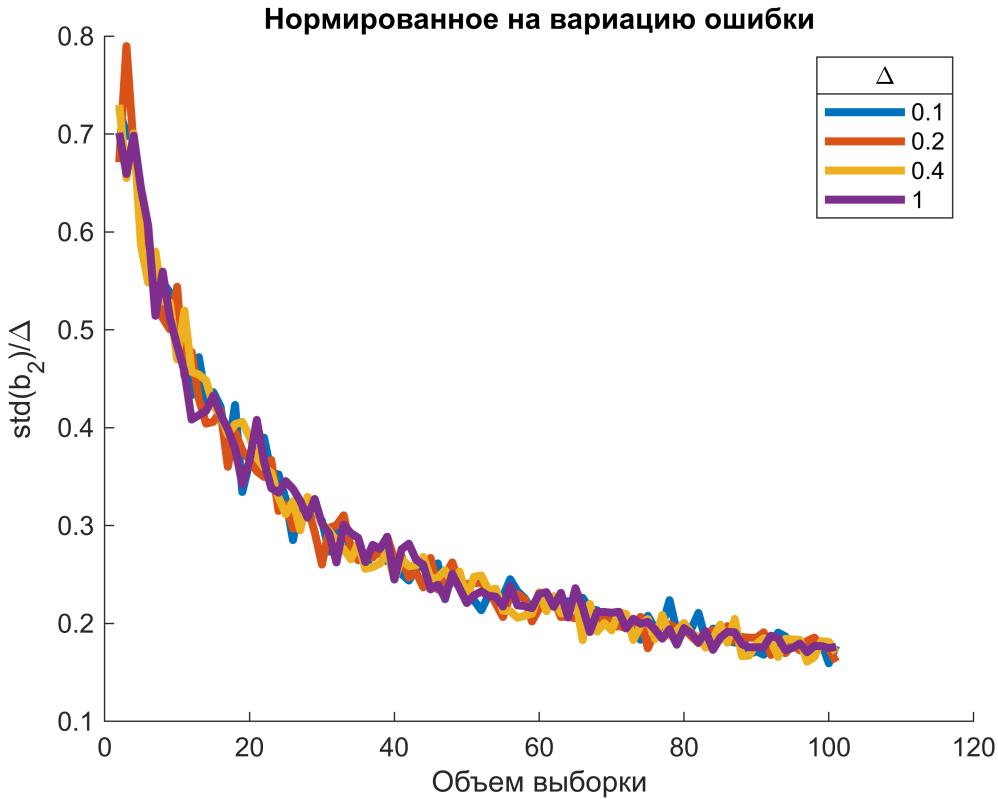
hold(ax, "on")
for ii = 1:numel(delta)
    [~,~,beta_mat] = sampling_surf_plot(beta,101,201,delta(ii));
    b_mat_selected = beta_mat(:,:,selection);
    b_std{ii}=std(b_mat_selected,0,2);
    plot(ax,sampling_volume,b_std{ii}, "LineWidth",3);
end
hold(ax, "off")
lgd = legend(ax,string(delta(:)), "Location", 'northeast');
lgd.Title.String = "\Delta ";
xlabel(ax, "Объем выборки");
ylabel(ax, "std("+b_str+")");
```



```

ax = get_next_ax();
hold(ax,"on")
for ii = 1:numel(delta)
    [~,~,beta_mat] = sampling_surf_plot(beta,101,201,delta(ii));
    b_mat_selected = beta_mat(:,:,selection);
    b_std{ii}=std(b_mat_selected,0,2);
    plot(ax,sampling_volume,b_std{ii}/delta(ii),"LineWidth",3);
end
title("Нормированное на вариацию ошибки")
hold(ax,"off");
lgd = legend(ax,string(delta(:)),"Location",'northeast');
lgd.Title.String = "\Delta ";
xlabel(ax,"Объем выборки");
ylabel(ax,"std("+b_str+")/\Delta");

```



Основной экспериментальный вывод, вариация коэффициентов регрессии прямо пропорциональна амплитуде ошибки исходных данных

Обозначения статистических параметров

Случайные величины, которые мы изучаем путем их розыгрыша (сэмплирования, выборки) характеризуются векторами этих сэмпллов

Пусть, вектора \vec{X} и \vec{Y} - вектора сэмпллов случайно величины, N - количество измерений.

$\vec{X} = \vec{I}^T \vec{X}/N$, $\vec{Y} = \vec{I}^T \vec{Y}/N$ - среднее значение выборки, \vec{I} - вединичный вектор (состоящий из единиц)

Квадратичные отклонения:

$$S_{XY} = [\vec{X} - \vec{X}\vec{I}]^T [\vec{Y} - \vec{Y}\vec{I}] = \vec{X}^T \vec{Y} - N\vec{X}\vec{Y}$$

$$S_{XX} = [\vec{X} - \vec{X}\vec{I}]^T [\vec{X} - \vec{X}\vec{I}] = \vec{X}^T \vec{X} - N\vec{X}^2$$

Это по сути скалярное произведение, только каждый из векторов смешен относительно своего среднего

Черточка сверху- среднее значение, стрелочка сверху - вектор.

$\overrightarrow{Var}(X) = S_{XX}/(N - 1)$ - вариация,

Если сложить данные в матрицу $M = [\overrightarrow{X}, \overrightarrow{Y}]$, то ковариация этой матрицы (это ковариация двух случайных скалярных величин, которые даются в виде вектор-столбцов их семплов, это не случайные вектора!) будет :

$$Cov(M) = [\overrightarrow{X}, \overrightarrow{Y}][\overrightarrow{X}^T] = \begin{bmatrix} S_{XX} & S_{XY} \\ S_{YX} & S_{YY} \end{bmatrix} / (N - 1)$$

Таким образом, в матрице ковариации выборок двух случайных переменных на диагонали стоят вариации каждой из переменных, а вне диагонали ковариации.

```
clearvars  
N=3;  
X=randn(N,1);Y=randn(N,1);  
cov([X,Y])
```

```
ans = 2x2  
0.2300 -0.1083  
-0.1083 0.0651
```

```
cov(X,Y)
```

```
ans = 2x2  
0.2300 -0.1083  
-0.1083 0.0651
```

```
X=X-mean(X);Y=Y-mean(Y);  
[X'*X,Y'*X;  
 X'*Y,Y'*Y]/(N-1)
```

```
ans = 2x2  
0.2300 -0.1083  
-0.1083 0.0651
```

Если случайные переменные некоррелированы, то их ковариация должна быть близка к нулю. Если коррелированы, то нет. Проведем простой эксперимент с бросанием монетки.

У нас есть две монетки. Пусть орел - это 1, а решка 0.

```
disp("Монетки независимы, ковариация равна нулю:")
```

Монетки независимы, ковариация равна нулю:

```
clearvars  
x1 = rand(1000,1)>0.5;  
x2 = rand(1000,1)>0.5;  
Sxy=(x1'*x2)/(999)
```

```
Sxy = 0.2202
```

```
%Sxy=(x1-mean(x1))'*(x2-mean(x2))/(999)
```

```
clearvars
disp("Монетки склеены разноименными краями, ковариация не равна нулю: " )
```

Монетки склеены разноименными краями, ковариация не равна нулю:

```
x1 = rand(1000,1)>0.5;
x2 = ~x1;
%Sxy=(x1-mean(x1))'*(x2-mean(x2))/(999)
Sxy=(x1'*x2)/(999)
```

Sxy = 0

Как так-то?

Регрессионная модель нулевого порядка

$\vec{X}_{\text{zero order}} = [\vec{I}]$ - матрица предикторов состоит только из одного единичного столбца

$$\vec{b} = b_0 = (\vec{X}^T \vec{X})^{-1} \vec{X}^T \vec{Y} = (\vec{I}^T \vec{I})^{-1} \vec{I}^T \vec{Y} = \frac{\sum_{i=1}^N y_i}{\sum_{i=1}^N 1} = \frac{\sum_{i=1}^N y_i}{N} = \bar{Y}$$

Таким образом, регрессия нулевого порядка описывает зависимость Y от X как среднее от имеющихся результатов наблюдения Y .

Первый порядок (прямая линия)

Матрица модели состоит всего из двух столбцов: $\vec{X}_{\text{first order}} = [\vec{I}, \vec{X}]$, соответственно, $\vec{X}_{\text{first order}}^T = [\vec{I}^T, \vec{X}^T]$

(напомню, что стрелочка в \vec{a} , обозначает всегда вектор-столбец)

Обратим внимание, что $\vec{I}^T \vec{X} = N \bar{X}$, где черточка сверху \bar{X} обозначает среднее арифметическое значение столбца.

$$\vec{b} = \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} = X_{\text{first order}}^\dagger \vec{Y} = \left(\begin{bmatrix} \vec{I}^T & \vec{X}^T \\ \vec{I} & \vec{X} \end{bmatrix} \right)^{-1} \begin{bmatrix} \vec{I}^T \\ \vec{X}^T \end{bmatrix} \vec{Y} = \left(\begin{bmatrix} N & \vec{I}^T \vec{X} \\ \vec{I} & \vec{X}^T \vec{X} \end{bmatrix} \right)^{-1} \begin{bmatrix} \vec{I}^T \\ \vec{X}^T \end{bmatrix} \vec{Y}$$

Учитывая, что (CM) $\vec{X}^T \vec{X} = S_{XX} + N \bar{X}^2$, а (CM) $\vec{X}^T \vec{Y} = S_{XY} + N \bar{X} \bar{Y}$:

$$\vec{b} = \begin{bmatrix} N & N\bar{X} \\ N\bar{X} & S_{XX} + N\bar{X}^2 \end{bmatrix}^{-1} \begin{bmatrix} N\bar{Y} \\ S_{XY} + N\bar{XY} \end{bmatrix}$$

| Обратную матрицу довольно легко инвертировать, формула для инвертирования матрицы 2x2

$$A^{-1} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}^{-1} = \frac{1}{\det(A)} \begin{bmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{bmatrix}, \text{ где определитель } \det(A) = a_{11}a_{22} - a_{21}a_{12}$$

```
A=sym("a",[2 2]);
inv(A)
```

ans =

$$\begin{pmatrix} \frac{a_{2,2}}{a_{1,1}a_{2,2} - a_{1,2}a_{2,1}} & \frac{-a_{1,2}}{a_{1,1}a_{2,2} - a_{1,2}a_{2,1}} \\ \frac{-a_{2,1}}{a_{1,1}a_{2,2} - a_{1,2}a_{2,1}} & \frac{a_{1,1}}{a_{1,1}a_{2,2} - a_{1,2}a_{2,1}} \end{pmatrix}$$

det(A)

ans = $a_{1,1}a_{2,2} - a_{1,2}a_{2,1}$

inv(A)*det(A)

ans =

$$\begin{pmatrix} a_{2,2} & -a_{1,2} \\ -a_{2,1} & a_{1,1} \end{pmatrix}$$

$$\det(X_{first\ order}^T X_{first\ order}) = N \overrightarrow{X}^T \overrightarrow{X} - (\overrightarrow{I}^T \overrightarrow{X})^2 = N \overrightarrow{X}^T \overrightarrow{X} - (N\bar{X})^2 = NS_{XX}$$

Тогда, с учетом формулы выше (см) окончательно получим выражение для коэффициентов аппроксимации:

$$\begin{bmatrix} b_0 \\ b_1 \end{bmatrix} = \frac{1}{N} S_{XX}^{-1} \begin{bmatrix} S_{XX} + N\bar{X}^2 & -N\bar{X} \\ -N\bar{X} & N \end{bmatrix} \begin{bmatrix} N\bar{Y} \\ S_{XY} + N\bar{XY} \end{bmatrix}$$

Отсюда получаем очень простые выражения для коэффициентов аппроксимации:

$$b_1 = \frac{1}{N} S_{XX}^{-1} [-N^2\bar{XY} + NS_{XY} + N^2\bar{XY}] = S_{XX}^{-1} S_{XY}$$

$$b_0 = S_{XX}^{-1} [NS_{XX}\bar{Y} + N^2\bar{X}^2\bar{Y} - NS_{XY}\bar{X} - N^2\bar{X}^2\bar{Y}] = \bar{Y} - S_{XX}^{-1} S_{XY}\bar{X} = \bar{Y} - b_1\bar{X}$$

```
% проверка правильности выражений для расчета через ковариации и через
% псевдообратную матрицу
clearvars
N = 10
```

N = 10

```

X = linspace(-1,1,N)';
Y = 0.0 + 0.43*X + 0.01*randn(10,1);
% через встроенную функцию:
Kxy = cov(X,Y) % матрица ковариации двух случайных переменных, представленных
векторами их семплов одинакового размера

```

```

Kxy = 2x2
0.4527    0.1965
0.1965    0.0855

```

```

% матричное выражение
[X,Y]'*[X,Y]/(N-1) % матрица ковариации двух случайных переменных X и Y

```

```

ans = 2x2
0.4527    0.1965
0.1965    0.0855

```

```

b1 = Kxy(1,2)/Kxy(1,1);
b0=mean(Y)-b1*mean(X);
b_vect_cov = [b0;b1] % через матрицу ковариации

```

```

b_vect_cov = 2x1
-0.0016
0.4342

```

```

% расчет через псевдообратную матрицу
I = ones(size(X));
b_vect_pseudoInverse = [I,X]\Y

```

```

b_vect_pseudoInverse = 2x1
-0.0016
0.4342

```

$$\vec{\hat{Y}} = [\vec{I}, \vec{X}] \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} = \vec{I}\vec{Y} - b_1 \vec{I}\vec{X} + b_1 \vec{X}$$

$$\vec{\hat{Y}} - \vec{I}\vec{Y} = b_1[\vec{X} - \vec{I}\vec{X}] = S_{XX}^{-1}S_{XY}[\vec{X} - \vec{I}\vec{X}]$$

Формула расчета параметров модели через вариацию-ковариацию для линейной регрессии справедлива и для общего случая (когда у нас имеются несколько предикторов). Задача тогда формулируется как:

$$\vec{Y} = [\vec{I}, \vec{X}_1, \dots, \vec{X}_P] \begin{bmatrix} b_0 \\ b_1 \end{bmatrix} = [\vec{I}, \vec{X}] \begin{bmatrix} b_0 \\ b_1 \end{bmatrix}$$

Вектор параметров модели также дается выражением:

$$\vec{b}_1 = S_{XX}^{-1} \vec{S}_{XY}$$

$$b_0 = \bar{Y} - \frac{\rightarrow^T}{\rightarrow X} b_1$$

Однако, теперь:

$$S_{XX} = (X - \frac{\rightarrow}{I} \frac{\rightarrow^T}{\rightarrow X})^T (X - \frac{\rightarrow}{I} \frac{\rightarrow^T}{\rightarrow X}) = X^T X - N \frac{\rightarrow}{\rightarrow X} \frac{\rightarrow^T}{\rightarrow X} = (N - 1) Cov(X)$$

$$\overrightarrow{S}_{XY} = X^T \overrightarrow{Y} - N \overrightarrow{X} \overrightarrow{Y}$$

\overrightarrow{X} - вектор-столбец средних значений предикторов. Подробный вывод см. [отдельно](#).

```
clearvars
```

```
N=10
```

```
N = 10

X1 = randn(N,1);
X2= randn(N,1);
I = ones(size(X1));
Y = 0.5 + 0.43*X1 + 0.61*X2+ 0.2*randn(N,1);
meanY=mean(Y)
```

```
meanY = 0.1782
```

```
X = [X1,X2];
meanX = I*mean(X)
```

```
meanX = 10x2
-0.6349 -0.0250
-0.6349 -0.0250
-0.6349 -0.0250
-0.6349 -0.0250
-0.6349 -0.0250
-0.6349 -0.0250
-0.6349 -0.0250
-0.6349 -0.0250
-0.6349 -0.0250
-0.6349 -0.0250
```

```
Sxy = (X-meanX)'*(Y-meanY)
```

```
Sxy = 2x1
7.4212
3.3963
```

```
Sxx = (X-meanX)'*(X-meanX)
```

```
Sxx = 2x2
20.4687 -1.6897
-1.6897 6.7716
```

```
cov(X)*(N-1)
```

```
ans = 2x2
20.4687 -1.6897
-1.6897 6.7716
```

```
b1 = Sxx\Sxy;
b0=mean(Y)-mean(X)*b1
```

```
b0 = 0.4553
```

```
b_vect_cov = [b0;b1]
```

```
b_vect_cov = 3x1
0.4553
0.4125
0.6045
```

```
b_vect_pseudoInverse= [I,X]\Y
```

```
b_vect_pseudoInverse = 3x1
0.4553
0.4125
0.6045
```

Альтернативный вариант - через матрицу ковариации всех переменных:

$$Cov([X, \vec{Y}]) = \frac{1}{N-1} \begin{bmatrix} (N-1)Cov(X) & \vec{S}_{XY} \\ \vec{S}_{XY}^T & S_{YY} \end{bmatrix}$$

```
clearvars
N=100
```

```
N = 100
```

```
X1 = randn(N,1); X2= randn(N,1); Y = 0.5 + 0.43*X1 + 0.61*X2+ 0.2*randn(N,1); %
генерим исходные данные
X = [X1,X2];
M = [X,Y]; % складываем все в одну табличку (матрицу свойств)
CovM = cov(M) % считаем ее ковариацию
```

```
CovM = 3x3
0.7897 0.0384 0.3615
0.0384 0.7826 0.4887
0.3615 0.4887 0.4898
```

```
b1_vec = CovM(1:2,1:2)\CovM(1:2,end) % находим коэффициенты предикторов
```

```
b1_vec = 2x1
0.4283
0.6034
```

```
b0=mean(Y) - mean(X)*b1_vec % нашли dummy variable
```

b0 = 0.5112

% для сравнения тоже самое по формулам линейной алгебры:

Матрица ковариации параметров модели

Теперь вернемся к линейной регрессии

$$\vec{b} = X^\dagger \vec{Y} = X^\dagger X \vec{\beta} + X^\dagger \vec{\epsilon} = \vec{\beta} + X^\dagger \vec{\epsilon}$$

$$\mathbb{E}[\vec{b}] = \vec{\beta}, \text{ так как } X^\dagger X = (X^T X)^{-1} X^T X = I, \text{ а } \mathbb{E}[\vec{\epsilon}] = \vec{0}$$

$$Cov(\vec{b}) = (\vec{\beta} - \vec{b})(\vec{\beta} - \vec{b})^T = ((X^\dagger X - I)\vec{b} + X^\dagger \vec{\epsilon})((X^\dagger X - I)\vec{b} + X^\dagger \vec{\epsilon})^T = X^\dagger \vec{\epsilon} \vec{\epsilon}^T (X^\dagger)^T$$

$$X^\dagger Cov(\vec{\delta}) X^{\dagger T}$$

Теперь воспользуемся тем, что вектор ошибки случайный и каждая точка независима от другой, и все они получены из нормального распределения $\delta \sim \sigma^2 N(0, 1)$ - означает, что δ получен из нормального распределения со средним значением равным нулю. Тогда ковариации будет диагональной:

$$Cov(\vec{\delta}) = \begin{bmatrix} \sigma^2 & \dots & 0 \\ 0 & \ddots & 0 \\ 0 & \dots & \sigma^2 \end{bmatrix}$$

$$X^\dagger Cov(\vec{\delta}) X^{\dagger T} = X^\dagger X^{\dagger T} \sigma^2 = (X^T X)^{-1} \sigma^2$$

Последний перевод следует из:

$$X^\dagger X^{\dagger T} = (X^T X)^{-1} X^T [(X^T X)^{-1} X^T]^T = (X^T X)^{-1} X^T X [(X^T X)^{-1}]^T = [(X^T X)^{-1}]^T = (X^T X)^{-1}$$

Таким образом, окончательно получим выражение для матрицы ковариации параметров линейной регрессии:

$$Cov(\vec{\beta}) = (X^T X)^{-1} \sigma^2$$

В этой формуле мы использовали то, что ошибка распределена нормально и для каждой точки дисперсия этого распределения одинакова.

Матрица ковариации вектора параметров линейной регрессии

$$Cov(\vec{Y}_{\text{interp}}) = X_{\text{interp}} Cov(\vec{\beta}) X_{\text{interp}}^T$$

Небольшой численный эксперимент на предмет того, что такое доверительная вероятность.

% проведем численный эксперимент на выполнение статистики
clearvars

```

M = 1000;% количество тестов
N = 100;% количество точек в выборке
beta2 = zeros(M,2);
P=2;% количество параметров регрессионной модели
for jj = 1:M
    x = randn(N,1);
    y = 1 + 2*x + randn(N,1);
    X = [ones(N,1),x];
    beta = X\y;
    e= X*beta - y;
    sigma = e'*e/(N-P);% считаем вариацию ошибки
    Cov_beta = (X'*X)\eye(P)*sigma;% считаем матрицу ковариации
    Var_beta = diag(Cov_beta);% считаем вариацию (диагональные элементы матрицы
    ковариации)
    beta2(jj,:)=beta(2) + sqrt(Var_beta(2))*[1 -1];
end
sum((2<=beta2(:,1)) & (2>=beta2(:,2)))/M

```

ans = 0.6870

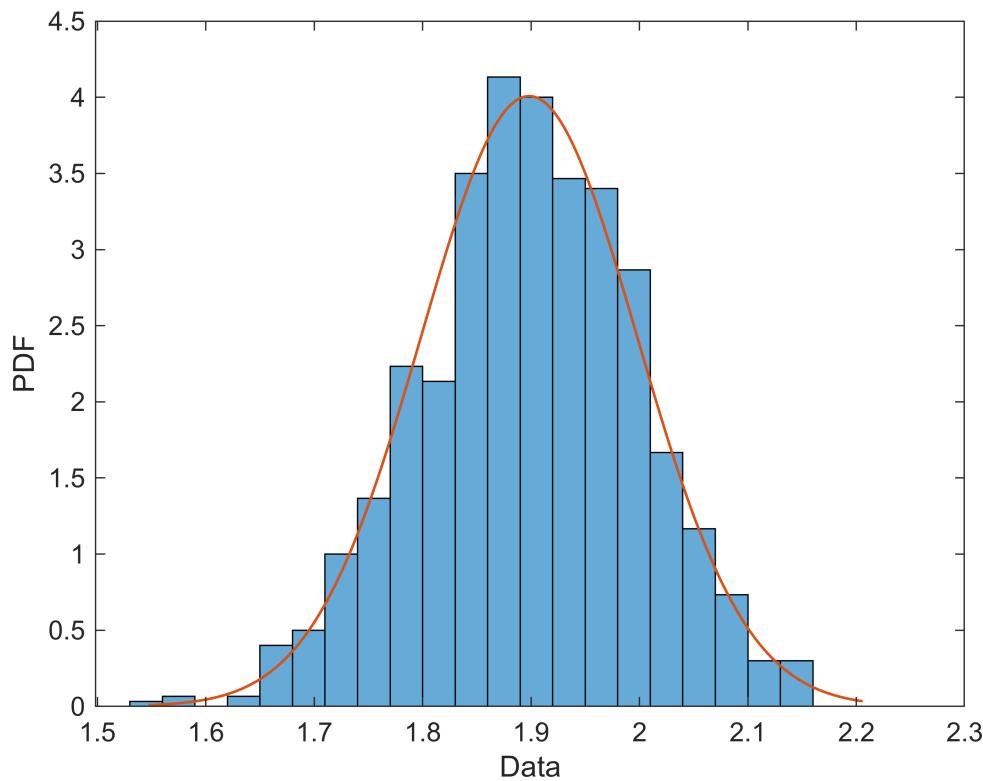
```
d = fitdist(beta2(:,2),"Normal")
```

```

d =
NormalDistribution

Normal distribution
mu = 1.89824 [1.89206, 1.90442]
sigma = 0.0995464 [0.0953667, 0.104112]
```

```
plot(d)
```



Проблема мультиколлинеарности и перебоучения, уменьшение размерности при помощи метода анализа главных компонент

Для примера рассмотрим последовательность работы PCA + линейная регрессия

Общая стратегия применения метода анализа главных компонент

- Разбиваем исходные данные на обучающую выборку и выборку для тестирования обученной модели
- Проводим анализ "лишних" данных и выкидываем их при помощи метода анализа главных компонент
- Обучаем регрессионную модель на обучающей выборке
- Преобразуем данные для тестирования в базис ортогональной системы координат главных компонент
- Рассчитываем прогноз при помощи обученной модели

Пример линейной регрессии с методом анализа главных компонент (LinReg + PCA)

Первый этап - загрузка и обработка данных.

В настоящее время для работы с данными большого объема большое распространение получил HDF5 (**Hierarchical Data Format**). Изначально разработан для суперкомпьютерных вычислений, в настоящее время поддерживается некоммерческой организацией HDF Group.

Это бинарный формат , который позволяет хранить данные в структурированном виде. Можно сказать, что это одна из попыток стандартизировать работу с большими данными в различных системах научных расчетов (matlab,julia,python,R), так и в языках общего назначения.

Содержит иерархию из двух основных типов объектов:

- Datasets — наборы данных, многомерные массивы объектов одного типа
- Groups — группы, являются контейнерами для наборов данных и других групп (по сути это папки)

Содержимое файлов HDF5 организовано подобно файловой системе, для доступа к данным применяются пути, сходные с POSIX-синтаксисом, например, `/path/to/resource`. Метаданные хранятся в виде набора именованных атрибутов объектов. Данные могут храниться в сжатом виде.

Внутренний формат данных матлаба - mat-файлы, начиная с версии 7.3 основаны на формате HDF5.

Воспользуемся встроенной в лискрипт утилитой для загрузки файла. В качестве примера будем пользоваться некоторыми метеорологическими данными из открытого источника ([Bottle Database – CalCOFI](#)). Они были загружены в формате csv и потом пересохранены в формат hdf5.

```
play_with_hdf5 = false; % эту галочку надо установить, если хочется поиграться с
% "сырыми" данным
% по умолчанию загружаются уже "просеянные" данные, сохраненные в мат-файле
if play_with_hdf5
% Create a structure to store imported HDF5 data
data = struct();

filename2 = "E:\projects\datasets\bases\bottle\bottle.h5";

data.Datasets(1).Name = "Btl_Cnt";
data.Datasets(1).Value = h5read(filename2, "/Btl_Cnt");

data.Datasets(2).Name = "Depthm";
data.Datasets(2).Value = h5read(filename2, "/Depthm");

data.Datasets(3).Name = "NO2uM";
data.Datasets(3).Value = h5read(filename2, "/NO2uM");

data.Datasets(4).Name = "O2ml_L";
data.Datasets(4).Value = h5read(filename2, "/O2ml_L");

data.Datasets(5).Name = "R_Depth";
data.Datasets(5).Value = h5read(filename2, "/R_Depth");

data.Datasets(6).Name = "R_PHAEO";
data.Datasets(6).Value = h5read(filename2, "/R_PHAEO");
```

```

data.Datasets(7).Name = "R_SALINITY";
data.Datasets(7).Value = h5read(filename2, "/R_SALINITY");

data.Datasets(8).Name = "STheta";
data.Datasets(8).Value = h5read(filename2, "/STheta");

data.Datasets(9).Name = "Salnty";
data.Datasets(9).Value = h5read(filename2, "/Salnty");

data.Datasets(10).Name = "T_degC";
data.Datasets(10).Value = h5read(filename2, "/T_degC");

clear filename2

% Display results
data

```

Формируем таблицу данных

```

all_data = data.Datasets
all_data_names = arrayfun(@(x)x.Name,all_data);
all_data_cell= arrayfun(@(x)(x.Value),all_data,"UniformOutput",false);
all_data_mat = cat(2,all_data_cell{:});
flag = any(isnan(all_data_mat),2)|any(all_data_mat== -1,2);% находим строки таблицы,
% содержащие NaN'ы
data_mat = all_data_mat(~flag,:);% удаляем эти строки
else
    data = load(fullfile(get_folder(),"bottle.mat"));
    data_mat = data.data_mat;
    all_data_names = data.all_data_names;
end
data_table = array2table(data_mat,VariableNames=all_data_names)

```

data_table = 208770×10 table

	Btl_Cnt	Depthm	NO2uM	O2ml_L	R_Depth	R_PHAEO	R_SALINITY
1	513064	0	0	5.8600	0	0.1900	33.4920
2	513065	1	0	5.8600	1	0.1900	33.4920
3	513066	10	0	5.8600	10	0.2300	33.4900
4	513067	11	0	5.8600	11	0.2400	33.4900
5	513068	20	0	5.8600	20	0.2200	33.4900
6	513069	30	0.0100	5.8500	30	0.1800	33.4900
7	513071	50	0.1100	5.1900	50	0.2100	33.4730
8	513073	66	0.0600	4.6400	66	0.2400	33.5420
9	513074	75	0.0900	4.4100	75	0.2300	33.5740
10	513076	99	0.0400	4.0100	99	0.1800	33.6360

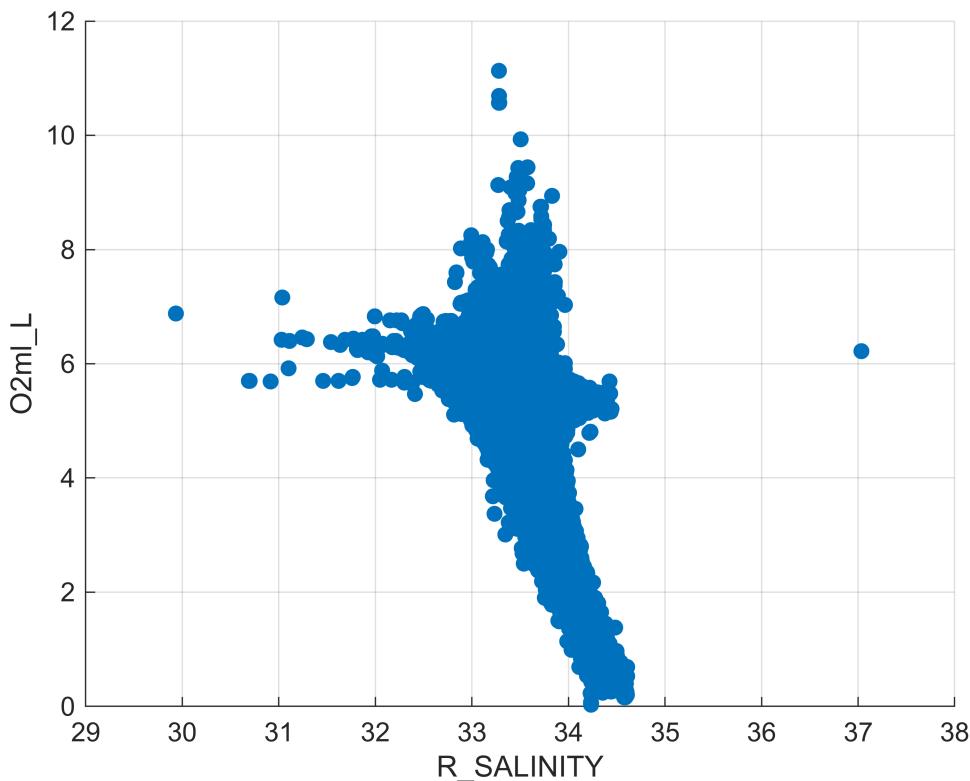
	Btl_Cnt	Depthm	NO2uM	O2ml_L	R_Depth	R_PHAEO	R_SALINITY
11	513077	100	0.0400	3.9900	100	0.1800	33.6390
12	513078	118	0.0100	3.6700	118	0.1600	33.7090
13	513111	0	0.0400	5.9600	0	0.1100	33.4670
14	513112	2	0.0400	5.9600	2	0.1100	33.4670
15	513113	10	0.0400	6	10	0.1500	33.4640
16	513114	11	0.0400	6.0100	11	0.1600	33.4630
17	513115	20	0.0500	6.0100	20	0.2000	33.4640
18	513116	30	0.0700	6.0200	30	0.2200	33.4660
19	513119	50	0.1600	5.2300	50	0.1800	33.3940
20	513120	62	0.0100	4.8100	62	0.1400	33.3680
21	513121	75	0.1300	4.3300	75	0.1400	33.5180
22	513122	93	0.0200	3.9900	93	0.0900	33.6300
23	513123	100	0.0100	3.8900	100	0.0800	33.6660
24	513125	125	0	3.5100	125	0.0400	33.7850
25	513126	134	0	3.3500	134	0.0300	33.8280
26	513173	0	0	5.9600	0	0.1000	33.2120
27	513174	2	0	5.9600	2	0.1000	33.2120
28	513175	10	0	5.9700	10	0.1100	33.2140
29	513176	12	0	5.9700	12	0.1100	33.2140
30	513177	20	0	5.9500	20	0.1400	33.2680
31	513178	30	0	5.9200	30	0.1900	33.3340
32	513179	31	0	5.9200	31	0.1900	33.3410
33	513182	64	0.2000	5.7800	64	0.2000	33.2510
34	513183	75	0.0500	5.4500	75	0.1300	33.1910
35	513184	78	0	5.3400	78	0.1100	33.1910
36	513185	97	0	4.6100	97	0.0700	33.3910
37	513370	0	0.0100	5.8800	0	0	33.3760
38	513371	1	0.0100	5.8800	1	0	33.3760
39	513372	10	0	5.8900	10	0.0400	33.3750
40	513373	20	0	5.9500	20	0.0600	33.3670
41	513374	29	0.0100	6	29	0.0700	33.3510
42	513378	50	0.0100	5.9200	50	0.1500	33.3400
43	513379	61	0.0100	5.9500	61	0.1800	33.2730

	Btl_Cnt	Depthm	NO2uM	O2ml_L	R_Depth	R_PHAE0	R_SALINITY
44	513380	75	0.1200	4.9500	75	0.1400	33.3210
45	513381	93	0.0100	3.8700	93	0.0800	33.6270
46	513382	100	0.0100	3.6500	100	0.0700	33.6970
47	513384	125	0.0100	3.2200	125	0.0200	33.8620
48	513385	133	0.0100	3.1100	133	0	33.9040
49	513461	0	0	5.7900	0	0.1100	33.5610
50	513462	1	0	5.7900	1	0.1100	33.5610
51	513463	10	0	5.7700	10	0.1300	33.5610
52	513464	11	0	5.7700	11	0.1300	33.5610
53	513465	20	0	5.7700	20	0.1300	33.5610
54	513466	29	0.0100	5.7600	29	0.1200	33.5610
55	513467	30	0.0100	5.7900	30	0.1200	33.5420
56	513470	50	0.2300	5.6700	50	0.1300	33.3890
57	513471	62	0.0500	5.1800	62	0.1400	33.3820
58	513472	75	0.0200	4.6800	75	0.1000	33.4900
59	513473	76	0.0200	4.6500	76	0.1000	33.5000
60	513474	96	0	4.3000	96	0.0900	33.5970
61	513493	0	0	5.9000	0	0.3000	33.4290
62	513494	2	0	5.9000	2	0.3000	33.4290
63	513495	10	0	5.8600	10	0.3300	33.4300
64	513496	11	0	5.8600	11	0.3300	33.4300
65	513497	20	0	5.9200	20	0.3300	33.4300
66	513498	30	0.0100	5.9600	30	0.3400	33.4290
67	513501	50	0.2100	5.4200	50	0.2500	33.3900
68	513502	62	0.1300	5.0200	62	0.1700	33.4160
69	513503	75	0.0100	4.5700	75	0.0900	33.5300
70	513504	76	0	4.5400	76	0.0800	33.5400
71	513505	94	0	4.0900	94	0.0800	33.6600
72	513506	100	0	3.9400	100	0.0700	33.7050
73	513508	125	0	3.3900	125	0.0200	33.8660
74	513509	136	0	3.1900	136	0	33.9160
75	513524	0	0	5.8500	0	0.0700	33.4970
76	513525	1	0	5.8500	1	0.0700	33.4970

	Btl_Cnt	Depthm	NO2uM	O2ml_L	R_Depth	R_PHAEO	R_SALINITY
77	513526	9	0	5.7800	9	0.0700	33.4970
78	513527	10	0	5.7900	10	0.0700	33.4970
79	513528	20	0	5.8700	20	0.0600	33.4970
80	513529	28	0	5.9300	28	0.0500	33.4960
81	513530	30	0	5.9100	30	0.0600	33.4950
82	513533	50	0.0800	5.8800	50	0.1700	33.2680
83	513534	61	0.0500	5.5500	61	0.2100	33.2640
84	513535	75	0.1100	5.2400	75	0.1500	33.3000
85	513536	77	0.1200	5.1900	77	0.1400	33.3100
86	513537	95	0.0100	4.6000	95	0.0500	33.4780
87	513538	100	0.0100	4.4200	100	0.0400	33.5270
88	513556	0	0	5.7900	0	0.0300	33.5380
89	513557	1	0	5.7900	1	0.0300	33.5380
90	513558	10	0	5.8500	10	0.0900	33.5320
91	513559	20	0	5.9400	20	0.0900	33.5170
92	513560	29	0	5.9700	29	0.0900	33.5160
93	513561	30	0	5.9600	30	0.0900	33.5190
94	513562	38	0	5.8900	38	0.0800	33.5440
95	513563	47	0.0100	5.8300	47	0.0900	33.5520
96	513564	50	0.0800	5.7300	50	0.1100	33.5290
97	513565	61	0.2700	5.3500	61	0.1600	33.4120
98	513566	75	0.0100	5.1700	75	0.0800	33.2040
99	513567	94	0.0100	4.4300	94	0.0500	33.4890
100	513568	100	0.0100	4.2400	100	0.0400	33.5620
:							

Можно посмотреть на данные в файле

```
x_var_index = 7;
y_var_index = 4;
ax = get_next_ax();
scatter(ax,data_table,x_var_index,y_var_index,"filled","AlphaData",0.5)
grid(ax,"on")
```



Формирование обучающей выборки и выборки для тестирования модели

```
dependent_variable = all_data_names(10) % выбираем зависимую переменную, которую будем предсказывать
```

```
dependent_variable =
"T_degC"

train_data_fraction=0.78;% доля исходных данных, которые идут на обучение
row_number = size(data_mat,1);
train_indices = 1:floor(row_number*train_data_fraction);% индексы тех строк, которые идут на обучение
test_indices = train_indices(end):row_number;% строки текстовой выборки
Y = data_table.(dependent_variable); % массив всех данных для выбранной переменной
flag = all_data_names~=dependent_variable;flag(1)=false;
X = table2array(data_table(:,flag)); % в предикторы идут все столбцы кроме столбца зависимой переменной и первого столбца (с индексами)
% формируем матрицы выборок
YTrain = Y(train_indices);YTest = Y(test_indices);
XTrain = X(train_indices,:);XTest = X(test_indices,:);
```

Главные компоненты для матрицы предикторов обучающей выборки находим SVD разложением:

$$X_{train} = U \Sigma V^T = \sum_{i=1}^r \sigma_i \vec{u}_i \vec{v}_i^T$$

Напомню, что матрица U - это ортонормированный базис пространства столбцов, а V - ортонормированный базис пространства строк. Значения σ_i - это сингулярные значения, отсортированные в порядке убывания, r - ранг матрицы. Последняя форма записи сингулярного разложения - представляет исходную матрицу в виде суммы одноранговых матриц отсортированных в порядке их "вклада" в общую матрицу.

```
mu_svd = mean(XTrain) % средние значения по каждому из столбцов
```

```
mu_svd = 1x8
72.5192    0.0541    4.8447   72.5192    0.1878   33.5034   25.1933   33.5034
```

```
[U,S,V] = svd(XTrain -mu_svd, 'econ'); % X = U*S*V'
```

Когда функция svd вызывается с признаком econ это означает, что она возвращает матрицы уменьшенной размерности, если X_{train} - матрица размером $m \times n$, то для $m > n$ рассчитываются только первые n столбцов матрицы U , а S - это матрица $n \times n$. Таким образом из матрицы сингулярных векторов выкидываются все нулевые сингулярные векторы и она становится диагональной.

```
US = U*S; % дальше эти матрицы будут нужны в виде произведения (отдельные матрицы не нужны)
100*diag(S.^2)/sum(diag(S.^2)) % отношение квадратов сингулярных значений показывает долю вариации
```

```
ans = 8x1
99.9832
0.0126
0.0023
0.0010
0.0008
0.0001
0.0000
0.0000
```

```
% исходных данных, которую объясняет каждая из компонент
norm((XTrain-mu_svd) - US*V') % убеждаемся, что формула сингулярного разложения работает
```

```
ans = 1.5524e-11
```

Операция умножения исходной матрицы на левые сингулярные вектора приводит ее к нормированному базису:

$$X_{train}V = U\Sigma$$

Матрица, которая стоит в равенстве справа - это ортогональная матрица (так как $(U\Sigma)^T U\Sigma = \Sigma^T U U^T \Sigma = \Sigma^2$ - диагональная), но не ортонормированная, то есть, столбцы этой матрицы ортогональны, но их модули не равны единице. Далее в соответствии с методом анализа главных компонент и теоремой Экхарта-Янга, мы уменьшаем размерность задачи, то есть, оставляем от исходной матрицы предикторов только матрицу ранга $r' < r$. То есть, в сумме, стоящей вы выражении для сингулярного разложения справа, оставляем только первые r' членов:

```

dimentinality = 5; % количество сингулярных значений (оно же размерность матрицы
после
% того как мы выкинем часть базисных векторов)
US_reduced = US(:,1:dimentinality);% уменьшаем размерность U*S
I = ones(size(US,1),1);

```

Решаем задачу регрессии - находим параметры модели (для линейной регрессии обучение состоит из одной итерации):

```
b = [I,US_reduced]\YTrain
```

```

b = 6×1
12.9935
-0.0280
1.1569
3.5669
3.5002
-0.8896

```

Раскладываем тестовые данные по базису правых сингулярных векторов

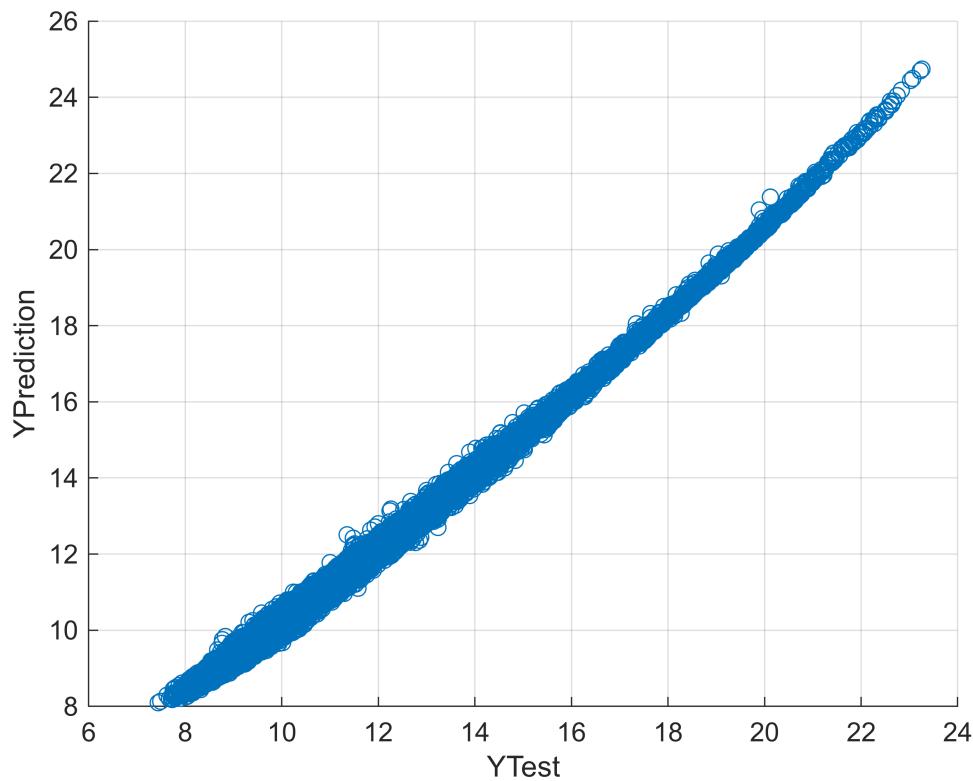
```
US_test = (XTest-mean(XTest))*V(:,1:dimentinality);
```

Рассчитываем предсказание модели

```

I = ones(size(US_test,1),1);
Yprediction = [I,US_test]*b;
YTest = Yprediction;
ax = get_next_ax();
scatter(ax,YTest,Yprediction)
xlabel("YTest")
ylabel("YPrediction")
grid on

```



Посчитаем корреляцию предсказания и тестовых данных:

$$Cor(X, Y) = \frac{Cov(X, Y)}{\sigma_A \sigma_B}$$

где σ - стандартное отклонение ($\sigma_X = \sqrt{Cov(X)}$), соответственно когда мы смотрим матрицу корреляции между

векторами сэмплов двух случайных величин, мы имеем симметричную матрицу, на диагонале которой стоят единицы, а офф-диагональные элементы дают коэффициенты корреляции, чем ближе коэффициент корреляции двух величин к единице, тем более они коррелированы.

```
corrcoef(YTest, Yprediction) %
```

```
ans = 2x2
1.0000    0.9989
0.9989    1.0000
```

Это был, относительно низкоуровневый подход, мы применяли базовые функции линейной алгебры - сингулярное разложение и mldivide (которое, как мы обсуждали, работает по методу QR-факторизации), сейчас попробуем сделать тоже самое, но при помощи более высокоуровневых функций.

Далее будем использовать statistical toolbox, в котором есть большой арсенал функций для машинного обучения, в том числе для регрессионного анализа.

Мы будем пользоваться двумя функциями:

pca(Xtrain) - функция для анализа данных методом главных компонент, Xtrain - это, соответственно, матрица предикторов для обучающей выборки

fitlm(scoreTrain,Ytrain) - эта функция проводит обучение регрессионной модели на обучающей выборке из матрицы предикторов **scoreTrain** и зависимой переменной **Ytrain** (это одна из колонок матрицы данных)

```
[coeff,scoreTrain,latent,tsquared,explained,mu] = pca(XTrain,"Centered",true);%  
coeff - это матрица V,
```

Warning: Columns of X are linearly dependent to within machine precision.
Using only the first 7 components to compute TSQUARED.

В качестве входного аргумента функции **pca** - матрица предикторов, дополнительно можно передавать алгоритм (по умолчанию используется 'svd', но может быть использован и алгоритм, основанный на спектральном разложении матрицы ковариации 'eig', а также 'als', про последний написано, что он лучше работает с пропущенными значениями (например, NaN) ; 'centered', true - пара аргументов, которая показывает надо ли центрировать данные (вычесть средние)

Функция **pca** возвращает следующие переменные: coeff, scoreTrain, latent, tsquared, explained, mu. coeff - это матрица V сингулярного разложения

Выходной аргумент scoreTrain - это произведение $U\Sigma$.

latent - это вектор квадратов сингулярных значений (он же вектор собственных значений матрицы ковариации)

tsquared - Hotelling's T-squared statistic is a statistical measure of the multivariate distance of each observation from the center of the data set.

explained - показывает какую долю вариации (выражается в процентах) объясняет данная спектральная компонента. Как показано выше, по сути доля вариации - это квадрат сингулярного значения нормированный на сумму квадратов всех сингулярных значений.

```
mean(coeff./V)
```

```
ans = 1×8  
1 1 -1 1 -1 1 1 -1
```

```
mean(US./scoreTrain)
```

```
ans = 1×8  
1 1 -1 1 -1 1 1 -1
```

```
transpose(sort(eig((XTrain-mu)'*(XTrain-mu)), "descend")/size(US,1)) % считаем через  
собственные значения матрицы ковариации
```

```
ans = 1×8  
103 ×  
6.7423 0.0008 0.0002 0.0001 0.0001 0.0000 0.0000 -0.0000
```

```
transpose(diag(S*S)./latent/(size(US,1)))
```

```
ans = 1x8
1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
```

```
transpose(100*(diag(S.*S)./sum(diag(S.*S)))./explained) % показывает в процентах
объясняемую данной матрицей долю вариации
```

```
ans = 1x8
1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000    1.0000
```

Смотрим какое число сингулярных значений объясняет 95 процентов вариации:

```
idx = find(cumsum(explained)>99,1)
```

```
idx = 1
```

Обучаем регрессионную модель на тренировочных данных, в которых была уменьшена размерность.

```
scoreTrain95 = scoreTrain(:,1:idx);
mdl = fitlm(scoreTrain95,YTrain)
```

```
mdl =
Linear regression model:
y ~ 1 + x1
```

Estimated Coefficients:

	Estimate	SE	tStat	pValue
(Intercept)	12.994	0.0049595	2619.9	0
x1	-0.028005	6.04e-05	-463.66	0

Number of observations: 162840, Error degrees of freedom: 162838

Root Mean Squared Error: 2

R-squared: 0.569, Adjusted R-Squared: 0.569

F-statistic vs. constant model: 2.15e+05, p-value = 0

```
scoreTest95 = (XTest-mu)*coeff(:,1:idx);
```

Дальше мы используем обученную модель `mdl` для того, чтобы предсказать значения в текстовой выборке, для этого используется функция `predict`, в которую в качестве первого аргумента передается обученная модель, а в качестве второго - новая матрица предикторов:

```
YTest_predicted = predict(mdl,scoreTest95);
norm(YTest_predicted - Yprediction)
```

```
ans = 409.1237
```

```
% расхождение между предсказаниями statistics toolbox и расчету
% при помощи базовых функций линейной алгебры
```

Пример использования метода анализа главных компонент с картинками

Мы любим примеры с картинками, поэтому попробуем, применить линейную регрессию для предсказания куска картинки. Теперь наше регрессионное уравнение будет иметь вид:

$$Y = XB + E$$

Сверху над Y, B, E пропали стрелочки, так как теперь это матрицы

Столбцы матрицы B - это вектора коэффициентов линейной регрессии, i -й столбец, соответственно, описывает нам i -й столбец матрицы Y . Для каждого столбца матрицы отклика мы находим вектор-столбец параметров линейной регрессии.

Будем рассматривать задачу следующим образом. Разделим картинку на четыре части

левая верхняя - будет матрица предикторов обучающей выборки (матрица X)

правая верхняя - матрица откликов (матрица Y)

левая нижняя часть - матрица предикторов тестовой выборки (X_{test})

правая нижняя часть - матрица отклика тестовой выборки

Наша задача состоит в том, чтобы не значая правую нижнюю часть картинки ее пресказать при помощи линейной регрессионной модели, которую мы обучем на обучающей выборке (то есть на верхней картинке)

```
clearvars
% Import image
folder = get_folder();
selected_image= "test1.jpg"; %
figs_folder = folder + "\figs";
full_file = fullfile(figs_folder,selected_image);
figs = imread(full_file);
M = size(figs,1)-2;N = size(figs,2)-2;
```

```
data= double((im2gray(figs)));
cut_size = [1,354,755];
data = data(cut_size(1):end,cut_size(2):cut_size(3));
imshow(uint8(data))
ax = gca;
title(ax, "Исходная картинка")
```

Исходная картинка



```
M = size(data,1);
N = size(data,2);
t_dims = [91,197]; % размер тестовой картинки

% test_image_dimentions
Mlast = M-t_dims(1)-1; % число строк в обучающей выборке
Nlast = N-t_dims(2)-1;% число столбцов предикторов в обучающей выборке
XTrain = data(1:Mlast,1:Nlast); % формируем матрицу предикторов
YTrain = data(1:Mlast,(Nlast+1):end); % формируем матрицу зависимой переменной
XTest = data((Mlast+1):end,1:Nlast);
YTest = data((Mlast+1):end,(Nlast+1):end);

imshow(uint8(join_matrix(XTrain,YTrain,XTest,YTest,10,0)))
```



```
Itrain = ones(size(XTrain,1),1);
B1 = [Itrain,XTrain]\YTrain;
Ipred = ones(size(XTest,1),1);
Ypredicted1 =[Ipred, XTest]*B1;
disp("Линейная регрессия без уменьшения размерности матрицы предикторов:")
```

Линейная регрессия без уменьшения размерности матрицы предикторов:

```
imshow(uint8(join_matrix(XTrain,YTrain,XTest,Ypredicted1,10,0)))
```



Добавляем метод главных компонент, чтобы понизить размерность матрицы предикторов обучающей выборки

```
% combined with pca
[coeff,scoreTrain,~,~,explained,mu] = pca(XTrain);% coeff - это матрица V,
mu_svd = mean(XTrain);% средние значения по каждому из столбцов
%M= size(XTrain,1);
%N = size(XTrain,2);
[U,S,V] = svd(XTrain -mu_svd, 'econ'); % X = U*S*V'
US = U*S;
MAX_S = size(S,1);

dimentionality = 4; % количество сингулярных значений
US_reduced = US(:,1:dimentionality);% уменьшаем размерность U*S
I = ones(size(US,1),1);
b = [I,US_reduced]\YTrain;
```

Раскладываем тестовые данные по базису правых сингулярных векторов

```
US_test = (XTest-mu_svd)*V(:,1:dimentionality);
```

Рассчитываем предсказание модели

```
I = ones(size(US_test,1),1);  
Yprediction = [I,US_test]*b;  
disp("Линейная регрессия без уменьшения размерности матрицы предикторов:")
```

Линейная регрессия без уменьшения размерности матрицы предикторов:

```
imshow(uint8(join_matrix(XTrain,YTrain,XTest,Yprediction,10,0)))
```



ЛИТЕРАТУРА

1. N.Draper, H.Smith . Applied regression analysis. Third edition. Wiley Series in Probability and Statistics.
2. Gilbert Strang - Introduction to Linear Algebra (2016, Wellesley-Cambridge Press)
3. John D'Errico (2025). Optimization Tips and Tricks (<https://www.mathworks.com/matlabcentral/fileexchange/8553-optimization-tips-and-tricks>), MATLAB Central File Exchange. Retrieved January 30, 2025.

Выводы.

1. Рассмотрели задачу линейной регрессии с точки зрения статистики. Основный аспекты терминологии: зависимая переменная (Y), матрица предикторов (X), тестовая и обучающая выборки, регрессионная модель.
2. Путем анализа серии регрессий убедились, что средние значения параметров линейной регрессии \vec{b} стремится к истинным значениям модели $\vec{\beta}$ при условии, что матожидание ошибки равно нулю. Также установили, что вариация ошибки приводит к вариации параметров регрессионной модели, причем, вариация параметров оказывается прямо пропорциональной вариации ошибки.
3. Сформулировали задачу линейной регрессии и получили выражения для коэффициентов модели через матрицу ковариации.
4. Получили выражения для матрицы ковариации параметров регрессионной модели, которая может быть использована для расчета доверительного интервала параметров регрессионной модели.
5. На двух примерах посмотрели последовательность операций при применении метода анализа главных компонент в задаче линейной регрессии для задач с одной выходной переменной и несколькими (multiooutput linear regression).

Вывод формулы для коэффициентов линейной регрессии через статистические параметры

Дополнение Шура (Schur complement) и инвертирование блочных матриц

Если у нас есть блочная матрица вида:

$$M = \begin{bmatrix} A & B \\ C & D \end{bmatrix} \quad (\text{то есть, матрица, у которой элементы - матрицы соответствующего размера})$$

дополнение Шура $Sh = D - CA^{-1}B$

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} + A^{-1}BSh^{-1}CA^{-1} & -A^{-1}BSh^{-1} \\ -Sh^{-1}CA^{-1} & Sh^{-1} \end{bmatrix}$$

Формула справедлива когда матрицы Sh и A - обратимы

<https://www.cis.upenn.edu/~jean/schur-comp.pdf>

Матричная форма линейной регрессии в статистической терминологии

Матрица модели: $X_M = [\vec{I}, X]$, теперь X - матрица предикторов (размером $N \times P$)

$$\begin{array}{l} \vec{b} = \begin{bmatrix} b_0 \\ \vec{b}_1 \end{bmatrix} = X_M^\dagger \vec{Y} = \left(\begin{bmatrix} \vec{I}^T \\ X^T \end{bmatrix} \left[\begin{bmatrix} \vec{I}, X \end{bmatrix} \right]^{-1} \begin{bmatrix} \vec{I}^T \\ X^T \end{bmatrix} \vec{Y} \right) = \left(\begin{bmatrix} N & \vec{I}^T \\ X^T \vec{I} & X^T X \end{bmatrix} \right)^{-1} \begin{bmatrix} \vec{I}^T \\ X^T \end{bmatrix} \vec{Y} (*) \end{array}$$

Вначале обратим внимание на кросс-диагональные

элементы блочной матрицы:

$$\vec{I}^T X \frac{1}{N} = [\sum_{i=1}^N X_{i1}, \dots, \sum_{i=1}^N X_{ij}, \dots, \sum_{i=1}^N X_{iP}] / N = [\bar{X}_1, \dots, \bar{X}_P] = \vec{\bar{X}}$$

То есть, $\vec{\bar{X}}$ - это вектор размером $P \times 1$, элементами которого являются средние значения столбцов матрицы X . С учетом этого, выражение (*) удобней переписать в виде:

$$\begin{bmatrix} b_0 \\ \vec{b}_1 \end{bmatrix} = X_M^\dagger \vec{Y} = [X_M^T X_M]^{-1} X_M^T \vec{Y} = \left(\begin{bmatrix} N & N \vec{\bar{X}}^T \\ N \vec{\bar{X}} & X^T X \end{bmatrix} \right)^{-1} \begin{bmatrix} N \vec{\bar{Y}} \\ X^T Y \end{bmatrix}$$

$$\text{Вначале рассмотрим блочную матрицу } X_M^T X_M = \begin{bmatrix} N & N \vec{\bar{X}}^T \\ N \vec{\bar{X}} & X^T X \end{bmatrix}.$$

Согласно формуле Шура, блочная матрица $M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$ может быть инвертирована через дополнение

$$\text{Шура: } Sh = D - CA^{-1}B, M^{-1} = \begin{bmatrix} A^{-1} + A^{-1}BSh^{-1}CA^{-1} & -A^{-1}BSh^{-1} \\ -Sh^{-1}CA^{-1} & Sh^{-1} \end{bmatrix}$$

Для нашей конкретной матрицы:

$$A^{-1} = \frac{1}{N} \text{ (просто скаляр), } B = N \vec{\bar{X}}^T \text{ (вектор-строка), } C = N \vec{\bar{X}} \text{ (вектор-столбец), } D = X^T X \text{ (матрица } P \times P)$$

$$[X_M^T X_M]^{-1} = \begin{bmatrix} N^{-1} + N^{-1}N \vec{\bar{X}}^T Sh^{-1}N \vec{\bar{X}} N^{-1} & -N^{-1}N \vec{\bar{X}}^T Sh^{-1} \\ -Sh^{-1}N \vec{\bar{X}} N^{-1} & Sh^{-1} \end{bmatrix} = \begin{bmatrix} N^{-1} + \vec{\bar{X}}^T Sh^{-1} \vec{\bar{X}} & \vec{\bar{X}}^T Sh^{-1} \\ -Sh^{-1} \vec{\bar{X}} & Sh^{-1} \end{bmatrix}$$

$$\begin{bmatrix} b_0 \\ \vec{b}_1 \end{bmatrix} = [X_M^T X_M]^{-1} X_M^T \vec{Y} = \begin{bmatrix} N^{-1} + \vec{\bar{X}}^T Sh^{-1} \vec{\bar{X}} & \vec{\bar{X}}^T Sh^{-1} \\ -Sh^{-1} \vec{\bar{X}} & Sh^{-1} \end{bmatrix} \begin{bmatrix} N \vec{\bar{Y}} \\ X^T Y \end{bmatrix}$$

Посмотрим внимательно на выражение для вектора параметров \vec{b}_1 :

$$\vec{b}_1 = Sh^{-1} [X^T \vec{Y} - N \vec{\bar{X}} \vec{\bar{Y}}]$$

Оно напоминает выражение для b_1 из формулы [для прямой линии](#), только там b_1 - это был скаляр.

Выражение в квадратных скобках имеет похожий смысл, как и выражение для [Sxy](#), там \vec{X} и \vec{Y} были векторами, а их средние значения - скалярами, теперь X - это матрица, $\vec{\bar{X}}$ - вектор-столбец средних значений матрицы X в каждом из столбцов. $X^T \vec{Y}$ - вектор, каждый элемент которого - скалярное

произведение столбца матрицы X на вектор \vec{Y} , вектор $\vec{\overline{XY}}$ - вектор, каждый элемент которого - произведение среднего значения столбца на \vec{Y} , то есть, среднее значения зависимой переменной \vec{Y} .
Теперь рассмотрим подробнее на дополнение Шура:

$Sh = X^T X - X^T \vec{I} \frac{1}{N} \vec{I}^T X$ - это матрица размером $P \times P$.

С учетом определения вектор-столбца средних, его можно переписать в виде:

$$Sh = X^T X - N \vec{\overline{X}} \vec{\overline{X}}^T$$

Данное выражение практически полностью совпадает с выражением для S_{XX} , когда рассматривалась одномерная задача.

Обозначим $\vec{\overline{X}} = \vec{\overline{XX}}^T$ матрицу размером $P \times P$, элемент этой матрицы, стоящий на i -й строке j -го столбца равен произведению средних значений этих столбцов:

$$\overline{X}_{ij} = \overline{X}_i \cdot \overline{X}_j$$

```
% убедимся в том, что выражения действительно совпадают
clearvars
```

```
N=3
```

```
X=sym("X",[N 2],"real");
I = sym(ones(N,1))
```

```
I =
```

$$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

```
Xmean = simplify(X'*I*I'*X)/N
```

```
Xmean =
```

$$\begin{pmatrix} \frac{(X_{1,1} + X_{2,1} + X_{3,1})^2}{3} & \frac{(X_{1,1} + X_{2,1} + X_{3,1})(X_{1,2} + X_{2,2} + X_{3,2})}{3} \\ \frac{(X_{1,1} + X_{2,1} + X_{3,1})(X_{1,2} + X_{2,2} + X_{3,2})}{3} & \frac{(X_{1,2} + X_{2,2} + X_{3,2})^2}{3} \end{pmatrix}$$

```
XmeanV = mean(X) % функция mean автоматически считает среднее в каждой колонке
```

```
XmeanV =
```

$$\begin{pmatrix} \frac{X_{1,1}}{3} + \frac{X_{2,1}}{3} + \frac{X_{3,1}}{3} \\ \frac{X_{1,2}}{3} + \frac{X_{2,2}}{3} + \frac{X_{3,2}}{3} \end{pmatrix}$$

```
simplify(N*XmeanV*XmeanV' - Xmean)
```

ans =

$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Теперь, что такое $X^T X$, это матрица, в которой на i-й строке j-го столбца стоит скалярное произведение i-го столбца на j-й столбец матрицы X

$$[X^T X]_{ij} = \overrightarrow{X_i} \cdot \overrightarrow{X_j}$$

Таким образом, дополнение Шура - это матрица, в которой на i-й строке j-го столбца стоит:

$$[Sh]_{ij} = \overrightarrow{X_i} \cdot \overrightarrow{X_j} - N \overline{X_i} \cdot \overline{X_j}$$

% убедимся в том, что выражения действительно совпадают

clearvars

N=3

N = 3

```
X=sym("X",[N 2],"real");
I = sym(ones(N,1))
```

I =

$$\begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$$

```
Xmean = simplify(X'*I*I'*X)/N
```

Xmean =

$$\begin{pmatrix} \frac{(X_{1,1} + X_{2,1} + X_{3,1})^2}{3} & \frac{(X_{1,1} + X_{2,1} + X_{3,1})(X_{1,2} + X_{2,2} + X_{3,2})}{3} \\ \frac{(X_{1,1} + X_{2,1} + X_{3,1})(X_{1,2} + X_{2,2} + X_{3,2})}{3} & \frac{(X_{1,2} + X_{2,2} + X_{3,2})^2}{3} \end{pmatrix}$$

XmeanV = mean(X) % функция mean автоматически считает среднее в каждой колонке

XmeanV =

$$\begin{pmatrix} \frac{X_{1,1}}{3} + \frac{X_{2,1}}{3} + \frac{X_{3,1}}{3} \\ \frac{X_{1,2}}{3} + \frac{X_{2,2}}{3} + \frac{X_{3,2}}{3} \end{pmatrix}$$

```
simplify(N*XmeanV*XmeanV' - Xmean)
```

ans =

$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Видно, что это матрица **ковариации** (умноженная на число измерений) матрицы, в которой столбцы - это сэмплы случайной величины:

$$Sh = (N - 1)Cov(X)$$

```
% убедимся в том, что выражения действительно совпадают
clearvars
```

N=3

N = 3

```
X=sym("X",[N 2],"real");
```

```
XmeanV = mean(X)'% функция mean автоматически считает среднее в каждой колонке
```

XmeanV =

$$\begin{pmatrix} \frac{X_{1,1}}{3} + \frac{X_{2,1}}{3} + \frac{X_{3,1}}{3} \\ \frac{X_{1,2}}{3} + \frac{X_{2,2}}{3} + \frac{X_{3,2}}{3} \end{pmatrix}$$

```
Sh = simplify(X'*X - N*XmeanV*XmeanV')
```

Sh =

$$\begin{pmatrix} \frac{2X_{1,1}^2}{3} - \frac{2X_{1,1}X_{2,1}}{3} - \frac{2X_{1,1}X_{3,1}}{3} + \frac{2X_{2,1}^2}{3} - \frac{2X_{2,1}X_{3,1}}{3} + \frac{2X_{3,1}^2}{3} & X_{1,1}X_{1,2} + X_{2,1}X_{2,2} + X_{3,1}X_{3,2} - \\ X_{1,1}X_{1,2} + X_{2,1}X_{2,2} + X_{3,1}X_{3,2} - (X_{1,2} + X_{2,2} + X_{3,2}) \left(\frac{X_{1,1}}{3} + \frac{X_{2,1}}{3} + \frac{X_{3,1}}{3} \right) & \frac{2X_{1,2}^2}{3} - \frac{2X_{1,2}X_{2,2}}{3} - \frac{2X_{1,2}}{3} \end{pmatrix}$$

```
Sigma= cov(X)*(N-1)
```

Sigma =

$$\begin{pmatrix} \sigma_7^2 + \sigma_6^2 + \sigma_5^2 & \sigma_1 \\ \sigma_1 & \sigma_4^2 + \sigma_3^2 + \sigma_2^2 \end{pmatrix}$$

where

$$\sigma_1 = \sigma_7 \sigma_4 + \sigma_6 \sigma_3 + \sigma_5 \sigma_2$$

$$\sigma_2 = \frac{X_{2,2}}{3} - \frac{2 X_{1,2}}{3} + \frac{X_{3,2}}{3}$$

$$\sigma_3 = \frac{X_{1,2}}{3} - \frac{2 X_{2,2}}{3} + \frac{X_{3,2}}{3}$$

$$\sigma_4 = \frac{X_{1,2}}{3} + \frac{X_{2,2}}{3} - \frac{2 X_{3,2}}{3}$$

$$\sigma_5 = \frac{X_{2,1}}{3} - \frac{2 X_{1,1}}{3} + \frac{X_{3,1}}{3}$$

$$\sigma_6 = \frac{X_{1,1}}{3} - \frac{2 X_{2,1}}{3} + \frac{X_{3,1}}{3}$$

$$\sigma_7 = \frac{X_{1,1}}{3} + \frac{X_{2,1}}{3} - \frac{2 X_{3,1}}{3}$$

```
simplify(Sh-Sigma) % убеждаемся, что это матрица ковариации
```

ans =

$$\begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}$$

Таким образом, окончательное выражение для вектора параметров линейной регрессии:

$$\vec{b}_1 = [X^T X - N \vec{X} \vec{X}^T]^{-1} [X^T \vec{Y} - N \vec{X} \vec{Y}]$$

```
function M = join_matrix(UL,UR,DL,DR,w,val)
% функция складывает четыре матрицы в одну, вставляя между ними нул, так,
% чтобы была рамка на картинке
arguments
```

UL

UR

DL

```

DR
w = 10
val = 0;
end
r1 = size(UL,1);r2 = size(DL,1);
c1 = size(DL,2);c2 = size(DR,2);
M = ones(r1+r2+w,c1+c2+w)*val;
M(1:r1,1:c1) = UL;
M(1:r1,(c1+w+1):end)=UR;
M((r1+w+1):end,1:c1) = DL;
M((r1+w+1):end,(c1+w+1):end) = DR;
end
function rus_name = rus(type)
possible_types = ["stand" "legA" "trig" "legP" "custom"];
possible_rus_names = ["Стандартный базис" "Присоединенные полиномы Лежандра"
"Тригонометрический базис" "Полиномы лежандра" "Кастомный"];
flag = possible_types==type;
if any(flag)
    rus_name = possible_rus_names(flag);
else
    rus_name = type;
end

end
function val = convert_rating_to_number(rating)
%rating = string(rating);
switch rating(1)
    case "A"
        val = 7+count(rating,"A");
    case "B"
        val = 8-count(rating,"B");
    case "C"
        val = 4-count(rating,"C");
end
end
function f = norm_distribution_function(mu,sig)
% возвращает анонимную функцию для нормального распределения
f =@(x) exp(-(x-mu).^2./(2*sig^2))./(sig*sqrt(2*pi));
end
function [V,s] = vandermatrix(t,P,type,polyprod_function)
arguments
    t double
    P (1,1) double {mustBeInteger,mustBePositive} = 2
    type (1,1) string {mustBeMember(type,[ "stand" "legA" "trig" "legP" "custom" ]) }
= "stand"
    polyprod_function = []
end
% создаем матрицу Вандермонда
% type - тип полинома (стандартный базис, полиномы лежандра,
% тригонометрические полиномы)

```

```

t = t(:);
N = numel(t);
V = zeros(N,P);
if ~ (type=="custom")
    [Pfun,s] = producing_function(type,t); % возвращаем производящую функцию
для колонки матрицы вандермонда
else
    assert(~isempty(polyprod_function)||
~isa(polyprod_function,"function_handle"),"Если выбрана кастомная производящая
функция, то нужно ее предоставить")
    s = normalize(t);
    Pfun = @(i)polyprod_function(i,s.x);
end
for jj = P:-1:1
    V(:,jj) = Pfun(jj);
end
end
function dist= make_dist(type,options)
% возвращает генератор N случайных чисел, распределенных в соответствии
% с двух параметрическими распределениями: нормальным, дельта-равномерным,
% Вейбулла и Хи-квадрат
arguments
    type (1,1) string {mustBeMember(type,[ "Normal" "Uniform" "Weibull" "Chi2"])}
    options.mu (1,1) =0
    options.sig (1,1)=0
end
mu=options.mu;
sig2=options.sig^2;
switch type
    case "Normal"
        dist = @(N) mu + sig2*randn([N 1]);
    case "Uniform"
        dist = @(N) sqrt(sig2)*(rand([N 1])-0.5)+ mu;
    case "Weibull"
        dist = @(N) wblrnd(mu,sqrt(sig2),N,1);
    case "Chi2"
        dist = @(N) (mu + sig2*randn([N 1]).^2);
end
end
function norm_struct = normalize2(t)
% функция возвращает структуру, в которой хранятся данные для нормировки
if ~issorted(t)
    t = sort(t,"ascend");
end
tmin = t(1);
tmax = t(end);
x = 2.0*((t - tmin) / (tmax - tmin))- 1;
norm_struct = struct("tmin",tmin,"tmax",tmax,"x",x); %t,(max(t) - min(t))
end

```

```

function t = denormalize2(s)
    t = 0.5*(s.x + 1.0)*(s.tmax - s.tmin) + s.tmin;
end
function Pn = leg_polyA(i,t) % производящая функция для присоединенных полиномов
Лежандра
    persistent P
    persistent tleg
    leg_type = 'norm';
    if isempty(tleg)||isempty(P)||(~isequal(t,tleg))
        P = transpose(legendre(i-1,t,leg_type)); % встроенная функция по сути
возвращает уже матрицу Вандермонда
        tleg = t;
    end
    if i<=size(P,2)
        Pn = P(:,i);
        return
    end
    P = transpose(legendre(i,t,leg_type));
    tleg = t;
    Pn = P(:,end);
end
function Pn = trig_poly(i,t) % производящая функция для тригонометрических полиномов
if i==1
    Pn = ones(size(t));
    return
end
if mod(i,2)==0
    Pn = cos(i*t*pi);
else
    Pn = sin(i*t*pi);
end
end
function [P,s] = producing_function(type,t)
% функция возвращает производящую функцию для полинома
s = normalize2(t);
switch type
    case "stand" % стандартный базис полинома
        P = @(i) s.x.^(i-1);
    case "legA" % присоединенные полиномы Лежандра
        P = @(i) leg_polyA(i,s.x);
    case "legP" % полиномы Лежандра
        P = @(i) legendreP(i-1,s.x); % стандартная функция для полиномов лежандра
    case "trig" % тригонометрический базис
        P = @(i) trig_poly(i,s.x);
end
end
function [new_ax,fig_handle] = get_next_ax(index, axes_name_value_pairs)
% функция, которая возвращает новые оси на новой фигуре (нужна чтобы

```

```

% кратинки в ливскрипте нормально строились)
arguments
    index = []
    axes_name_value_pairs cell = {}
end
persistent N;
if isempty(index)
    if isempty(N)
        N=1;
    else
        N = N+1;
    end
    fig_handle = figure(N);
    clf(fig_handle);
    new_ax = axes(fig_handle,axes_name_value_pairs{:});
    %disp("fig"+ N)
else
    fig_handle = figure(index);
    clf(fig_handle);
    new_ax = axes(fig_handle,axes_name_value_pairs{:});
end
end
function ax = get_named_ax(title_string)
    ax = get_next_ax();
    title(title_string);
end
function ax = draw_vector(ax,ttl,names,type,varargin)
% функция строит двух- и трех-мерные вектора, а также рассеянные данные из
% матрицы
% ax - оси (если пустые, то создаются новые)
% ttl - заголовок картинки
% names - имена векторов
% type:
%     "vector" - аргументы, которые передаются после интерпретируются
%                 как отдельные вектора
%     "point" - в этом случае передается матрица в качестве аргумента и
%               столбцы матрицы строятся при помощи функций scatter и scatter3 в
%               зависимости от размерности массива
arguments
    ax =[]
    ttl string =strings(0,1)
    names string =strings(0,1)
    type string {mustBeMember(type,["vector" "point"]))}="vector"
end
arguments (Repeating)
    varargin double
end
was_empty = isempty(ax); % это признак того, что все строится на новых осях
if was_empty
    ax = get_next_ax();

```

```

else
    hold(ax,"on");
    % if ~isempty(ax.Legend)
    %     leg_before = ax.Legend.String;
    % else
    %     leg_before = strings(0,1);
    % end
end

if strcmp(type,"vector")
    is_3D = numel(varargin{1})==3;
    if is_3D
        [x,y,z] = make_xy(varargin{1});
        plot3(ax,x,y,z,'LineWidth',2,'Marker','o');
        hold on
        for iii = 2:numel(varargin)
            [x,y,z] = make_xy(varargin{iii});
            plot3(ax,x,y,z,'LineWidth',2,'Marker','o');
        end
        grid on
        hold off
    else
        [x,y] = make_xy(varargin{1});
        plot(ax,x,y,'LineWidth',2,'Marker','o');
        hold on
        for iii = 2:numel(varargin)
            [x,y] = make_xy(varargin{iii});
            plot(ax,x,y,'LineWidth',2,'Marker','o');
        end
        grid on
        hold off
    end
    if isempty(names)|| (numel(names)~=numel(varargin))
        legend(ax,string(1:numel(varargin)));
    else
        % if ~was_empty
        %     names= [names(:);leg_before(:)];
        % end
        legend(ax,names);
    end
    xlim(ax,[-1 1]);
    ylim(ax,[-1 1]);
    if ~isempty(ttl)
        title(ax,ttl);
    end
else
    %data_number = numel(varargin); % число массивов данных
    is_3D = numel(varargin)==3;
    data = varargin{1};

```

```

if size(data,2)>1
    data = transpose(data);
    is_transpose = true;
else
    is_transpose = false;
end
if ~is_transpose
    for iii = 2:numel(varargin)
        data = [data,varargin{iii}];
    end
else
    for iii = 2:numel(varargin)
        data = [data,transpose(varargin{iii})];
    end
end

if is_3D
    scatter3(ax,data(:,1),data(:,2),data(:,3));
else
    scatter(ax,data(:,1),data(:,2));
end

end
if ~was_empty
    hold(ax,"off");
end
end
function [x,y,z] = make_xy(col)
% добавляет к координатам вектора нули так, чтобы при помощи функции plot
% строилась линия
switch numel(col)
case 1
    x = [col(1)];
    y = 0;
    z = 0;
case 2
    x = [0 col(1)];
    y = [0 col(2)];
    z = zeros(1,2);
case 3
    x = [0 col(1)];
    y = [0 col(2)];
    z = [0 col(3)];
end
end
function folder = get_folder()
% текущая папка
folder = fileparts(matlab.desktop.editor.getActiveFilename);
end

```

```

function [sampling_volume,tests_number,beta_mat] =
sampling_surf_plot(beta,Nmax,Mmax,delta)
    % N
    P = numel(beta);
    sampling_volume = 2:Nmax;
    tests_number = 2:Mmax;
    beta_mat = zeros([Nmax-1,Mmax-1,P]);
    for N=sampling_volume % первый индекс - объем выборки
        x = transpose(linspace(-1,1,N));% столбец координат
        X = vandermatrix(x,P); % матрица вандермонда
        Yo = X*beta;% истинное значение
        for ii = tests_number% второй индекс - номер теста
            Y = Yo + delta*randn(N,1);
            b_values = X\Y;
            beta_mat(N-1,ii-1,:) = b_values;
        end
    end
end

```