# Регрессионный анализ и метод наименьших квадратов

## Числа, операции над числами и их свойства

**Число** - математический объект, основное понятие математики, используемое для количественной характеристики, сравнения, нумерации объектов и их частей.

- Натуральные числа ( $\mathbb{N}$) (полукольцо натуральных числе с нулем)
  - сложение (+)
  - умножение (·)
  - коммутативность сложения и умножения $a + b = b + a; a \cdot b = b \cdot a$
  - ассоциативность сложения и умножения $(a + b) + c = a + (b + c); (a \cdot b) \cdot c = a \cdot (b \cdot c)$
  - дистрибутивность умножения относительно сложения $(a + b) \cdot c = a \cdot b + a \cdot c$
  - Существование нейтрального элемента: $a + 0 = 0 + a = a$
  - Мультипликативной свойство нуля: $a \cdot 0 = 0 \cdot a = 0$
  - Существует единичный элемент (1): $a \cdot e = e \cdot a = a$
- Целые числа ( $\mathbb{Z}$ ) (кольцо)
  - Существование противоположного элемента относительно сложения: $\forall a \in \mathbb{Z} \exists b \in \mathbb{Z} (a + b = b + a = 0)$
- Рациональные числа ( $\mathbb{Q}$ ) (поле)
  - Существование обратного элемента для ненулевых элементов: $(\forall a \in \mathbb{Q} : a \neq 0) \exists a^{-1} \in \mathbb{Q} : a \cdot a = e$, где $e$ - единичный элемент.
- Действительные (вещественные) числа ( $\mathbb{R}$ )

# Некоторые понятия линейной алгебры

---

## Определения

- скаляр - величина, не обладающая направлением, задается числом.
- вектор - величина, обладающая направлением, может описываться двумя и более числами.
- матрица - вектор в пространстве $R^m n$, имеющий размерность $m$ x $n$, где $m$ строк и $n$ столбцов.

**Скаляр (скалярная величина)** (далее будем обозначать маленьгими греческими

буквами: $\alpha, \beta, \lambda$) - величина, полностью определяемая в любой координатной системе одним числом или функцией, которое не изменяется при изменении пространственной системы координат, т.е. величина, не обладающая направлением, задается числом.

**Вектор** (далее будем обозначать маленькими латинскими буквами: $a, b, x, y$) - в самом общем смысле: последовательность, (кортеж) однородных элементов. В рамках линейной алгебры - это элемент векторного пространства. В простейшем случае вектор определяется как математический объект, характеризующийся величиной и направлением.

Векторы называются **колинеарными** если они лежат на одной прямой. Векторы, лежащие в параллельных плоскостях, называются **компланарными**.

**Векторное пространство**

Множество $\mathcal{L}$ с операциями на нем сложенияи $(+)$ и умножения $(\cdot)$ векторов на векторы и скаляры называется векторным пространством *(или пространство $\mathcal{L}(\mathcal{F})$ над полем $\mathcal{F}$, где $\mathcal{F}$ обычно поле действительных или комплексных чисел)*, если справедливы следующие свойства:

1. коммутативность сложения: для любых векторов $x$ и $y$: $x + y = y + x$;
2. ассоциативность сложения: для любых векторов $x$, $y$ и $z$: $(x + y) + z = x + (y + z)$;
3. существование нулевого вектора: существует нулевой вектор 0 такой, что для любого вектора $x$: $x + 0 = x$;
4. существование противоположного элемента: для любого вектора $x$ существует такой вектор $-x$, что $x + (-x) = 0$;
5. свойство единицы: для любого вектора $x$: $1 \cdot x = x$;
6. ассоциативность умножения на число: для любого вектора $x$ и любых чисел $\alpha$ и $\beta$: $(\alpha\beta)x = \alpha(\beta x)$
7. дистрибутивность относительно сложения векторов: для любых векторов $x$ и $y$ и любого числа $\alpha$: $\alpha(x + y) = \alpha x + \alpha y$;
8. дистрибутивность относительно сложения чисел: для любого вектора $x$ и любых чисел $\alpha$ и $\beta$: $(\alpha + \beta)x = \alpha x + \beta x$.

$\mathcal{NB}$ **- Коммутативность относитльно умножения не предусмотрена.**

**Линейная комбинация**

Линейной комбинацией векторов $e_1, e_2, e_3, \dots e_n$ линейного пространства $\mathcal{L}$ называется вектор $x = e_1 \cdot \alpha_1 + e_2 \cdot \alpha_2 + e_3 \cdot \alpha_3 + \dots + e_n \cdot \alpha_n$, где $\alpha_1 \dots \alpha_n$ - числа $\in \mathcal{F}$.

Если все скаляры равны 0, то такая комбинация называется **тривиальной линейной комбинацией**.

**Линейное подпространство,**

Линейное подпространство, или векторное подпространство, — непустое подмножество $\mathcal{K}$ линейного пространства $\mathcal{L}$ такое, что $\mathcal{K}$ само является линейным пространством по отношению к определённым в $\mathcal{L}$ действиям сложения и умножения на скаляр. При этом:

для всяких векторов $x, y \in \mathcal{K}$ вектор $\alpha x + \beta y$ также принадлежит $\mathcal{K}$ для любых $\alpha, \beta \in \mathcal{F}$.

**Линейная зависимость и независимость**

Система $e_1, e_2, e_3, \dots e_n$ линейно независима, если равенство $e_1 \cdot \alpha_1 + e_2 \cdot \alpha_2 + e_3 \cdot \alpha_3 + \dots + e_n \cdot \alpha_n = 0$ возможно только для тривиальной линейной комбинации.

Система $e_1, e_2, e_3, \dots e_n$ линейно зависима, если существует нетривиальная линейная комбинация, для которой справедливо равенство $e_1 \cdot \alpha_1 + e_2 \cdot \alpha_2 + e_3 \cdot \alpha_3 + \dots + e_n \cdot \alpha_n = 0$

Является ли комбинация векторов линейно зависимой:

1.

- $e_1 = [1, 3, 5]$
- $e_2 = [2, 3, 2]$

2.

- $e_1 = [1, 2, 1]$
- $e_2 = [1, 2, 3]$
- $e_2 = [3, 6, 7]$

3.

- $e_1 = [1, 2, 1]$
- $e_2 = [1, 2, 3]$
- $e_2 = [0, 0, 0]$

**Регрессионный анализ** — набор статистических методов исследования влияния одной или нескольких независимых переменных $X_1, X_2, \dots, X_p$ на зависимую переменную $Y$. Независимые переменные иначе называют **регрессорами** или **предикторами**, а зависимые переменные — **критериальными** или **регрессантами**. Терминология зависимых и независимых переменных отражает лишь математическую зависимость переменных, а не причинно-следственные отношения. Наиболее распространённый вид регрессионного анализа — **линейная регрессия**, когда находят линейную функцию, которая, согласно определённым

математическим критериям, наиболее соответствует данным. Например, в методе наименьших квадратов вычисляется прямая (или гиперплоскость), сумма квадратов между которой и данными минимальна.

Какие еще виды регрессии бывают?

- Модификации линейной регрессии - Ridge-регрессия / Lasso-регрессия
- Нелинейная регрессия

**Линейную модель** обычно формулируют следующим образом:

$$y = \beta_0 + \beta_1 \cdot X_1 + \beta_2 \cdot X_2 + \ldots + \beta_n \cdot X_n + e$$

Пусть даны $m$ измерений, где каждому измерению соответствует $n$ характеристик (уровней факторов). Тогда предикторы (характеристики) могут быть объеденены в матрицу $X$ (таблицу) по строкам которой - наблюдения, по столбцам - характеристики. Результат каждого наблюдения $y$ будет формировать ветор **ответов**. Такимо образом, получается система линейных уравнений составленная из отображений $X_m \to y_m$:

$$\begin{cases} \beta_1 \cdot x_{1,1} + \beta_2 \cdot x_{1,2} + \ldots + \beta_n \cdot x_{1,n} = y_1 \\ \beta_1 \cdot x_{2,1} + \beta_2 \cdot x_{2,2} + \ldots + \beta_n \cdot x_{2,n} = y_2 \\ \ldots \\ \beta_1 \cdot x_{m,1} + \beta_2 \cdot x_{m,2} + \ldots + \beta_n \cdot x_{m,n} = y_m \end{cases} \tag{1}$$

Которую также можно записать в матричной форме:

$$X \cdot \beta = y$$

$X$ - $m \times n$ дизайн-матрица (матрица коэффициентов СЛУ), $\beta$ - вектор неизвестных значений, $y$ - вектор значений ("ответов").

Найти решение означает - найти такой вектор $\beta$, который максимально удовлетворяет системе линейных уравнений. Т.к. наблюдений практически всегда больше чем предикторов, то конкретного решения не существует. Поэтому используются определенные правила, которые устанавливают насколько близко найденное решение по отношению к полученным данным. Одним из таких методов является **метод наименьших квадратов**.

#### Метод наименьших квадратов

Метод применяется для решения задач линейной регрессии.

Формально МНК варажается следующим образом:

$\sum_{i=1}^{M}(y_i - \hat{y}_i)^2 \to \min$, где:

- $M$ — число наблюдений;
- $y_i$ — значение $i$-го наблюдения;
- $\hat{y}_i$ — оценка значения $i$-го наблюдения (т.е. результат: $\hat{y}_i = X_i \cdot \beta$);

**Вывод**

Пусть $X$ - $m \times n$ дизайн-матрица (матрица коэффициентов СЛУ), $\beta$ - вектор неизвестных значений, $y$ - вектор значений ("ответов"). И дана система уравнений:

$$X \cdot \beta = y$$

В случае если количество строк $m > n$ система уравнений не будет иметь решения. Такая система является **переопределённой** и соответственно может быть переформулирована как:

$$X \cdot \beta + \varepsilon = y$$

**Обозначим:**

$$X \cdot \beta = \hat{y}$$

$$\varepsilon = y - \hat{y} = y - X \cdot \beta$$

Где $\hat{y}$ - какой-то вектор ответов при некотором векторе $\beta$. $\varepsilon$ - вектор "ошибок".

Суть задачи сводится к нахождению такого $\beta$ при котором будет получен как можно меньший вектор "ошибок" $\varepsilon$. По факту задача сводится с минимизации расстояния между векторами $\hat{y}$ и $y$. Для этого будет выполнена минимизация скалярного произведения вектора $\varepsilon$, т.к. скалярное произведение вектора с самим собой и есть сумма квадратов. Т.е. $< \varepsilon, \varepsilon >= f(X, y, \beta) \rightarrow min_{\beta}$

Запишем:

$$f(X, y, \beta) = \varepsilon^T \cdot \varepsilon = (y - \hat{y})^T \cdot (y - \hat{y}) = (y - X \cdot \beta)^T \cdot (y - X \cdot \beta)$$

Раскроем скобки:

$$f(X, y, \beta) = y^T \cdot y - y^T \cdot X \cdot \beta - (X \cdot \beta)^T \cdot y + (X \cdot \beta)^T \cdot X \cdot \beta$$

Еще раскроем скобки и напишем без знаков умножения:

$$f(X, y, \beta) = y^T y - y^T X \beta - \beta^T X^T y + \beta^T X^T X \beta$$

Для минимизации этой функции воспользуемся обычным способом математического анализа - приравняем производную к 0 и найдем решение.

$$\frac{\partial f(X,y,\beta)}{\partial \beta} = \frac{\partial (y^T y - y^T X \beta - \beta^T X^T y + \beta^T X^T X \beta)}{\partial \beta} = \frac{\partial (y^T y - 2\beta^T X^T y + \beta^T X^T X \beta)}{\partial \beta}$$

*Примечание: $y^T X \beta$ и $\beta^T X^T y$ - числа, т.о. эти выражения можно безболезненно транспонировать* => $(y^T X \beta)^T = (X\beta)^T y = \beta^T X^T y$

Производные слагаемых, которое не зависят от $\beta$ равны 0.

- $f(x) = x^T A x$
    - $f'(x) = 2Ax$

- $f(x) = x^T A$
    - $f'(x) = A$

*Принципы дифференцирования по вектору см. The Matrix Cookbook*

Далее:

$\frac{\partial f(X,y,\beta)}{\partial \beta} = -2X^T y + 2X^T X\beta = 0$

$X^T X\beta = X^T y$

Что бы выразить $\beta$ - умножаем правую и левую чать уравнения на обратную матрицу ($(X^T X)^{-1}$) к матрице $X^T X$:

$(X^T X)^{-1} X^T X\beta = (X^T X)^{-1} X^T y$

$\beta = (X^T X)^{-1} X^T y$

**Это основное уравнение коэффициентов линейной регрессии ___**

Где $y$, $\beta$, $\varepsilon$ - вектора, $X$ - матрица

**Применение QR разложения для нахождения решения по методу наименьших квадратов**

**QR разложение**: матрица $A$ размера $m \times n$ с комплексными элементами может быть представлена в виде: $A = QR$, где $Q$ — матрица размера $m \times n$ с ортонормированными столбцами (или унитарная (квадратная матрица с комплекными элементами) матрица размера $m \times m$), а $R$ — верхнетреугольная матрица размера $n \times n$.

В частном случае, когда матрица $A$ состоит из вещественных чисел, $Q$ является ортогональной матрицей (то есть $Q^T Q = E$, где $E$ - единичная матрица.

Для решения в уравнении $X^T X\beta = X^T y$ можно заменить $X = QR$.

Тогда с учетом свойства: $Q^T Q = E$

$(QR)^T QR\beta = (QR)^T y$

$R^T Q^T QR\beta = R^T Q^T y$

Выражение урощается до:

$R^T ER\beta = R^T Q^T y$

И соответственно:

$R\beta = Q^T y$

Система еще не решена окончательно, т.к. $\beta$ не выражено явно.

Пусть $\nu = Q^T y$, тогда решение системы $R\beta = \nu$ - тривиально т.к. $R$ - верхнетреугольная матрица.

или также как в предыдущем случае с помощью умножения обеих частей на обратную матрицу:

$\beta = R^{-1}Q^T y$

Возможен и другой способ, при котором выполняется замена $X^T X = QR$:

$X^T X\beta = X^T y$

$QR\beta = X^T y$

$R\beta = Q^T X^T y$

**Параметры вариации:**

- Вектор ошибок:

$\varepsilon = y - \hat{y} = y - X \cdot \beta$

- Сумма квадратов:

$SS = < \varepsilon, \varepsilon >$

- "Остаточная" вариация / вариация модели (Deviance):

$\sigma^2 = \frac{SS}{N - rank(X)}$

- Вариационно-ковариационная матрица $\beta$:

$cov(\beta|X) = \sigma^2(X^T \cdot X)^{-1}$

- Стандартное отклонение $\beta$:

$SD_\beta = \sqrt{diag(cov(\beta|X))}$

In [19]:
```
using DataFrames, CSV, GLM

df = download("https://raw.githubusercontent.com/PharmCat/edu/main/csv/muscule.c

df[1:3, :]
```

Out[19]: 3×3 DataFrame

| Row | Strip | conc | length |
|---|---|---|---|
| | String3 | Float64 | Float64 |
| 1 | S01 | 2.2 | 15.8 |
| 2 | S01 | 4.4 | 20.8 |
| 3 | S01 | 6.6 | 22.6 |

In [20]:
```
df
```

Out[20]: 60×3 DataFrame *35 rows omitted*

| Row | Strip | conc | length |
|---|---|---|---|
| | String3 | Float64 | Float64 |
| **1** | S01 | 2.2 | 15.8 |
| **2** | S01 | 4.4 | 20.8 |
| **3** | S01 | 6.6 | 22.6 |
| **4** | S01 | 8.8 | 23.8 |
| **5** | S02 | 2.2 | 20.6 |
| **6** | S02 | 4.4 | 26.8 |
| **7** | S02 | 6.6 | 28.4 |
| **8** | S02 | 8.8 | 27.0 |
| **9** | S03 | 0.55 | 7.2 |
| **10** | S03 | 1.1 | 15.4 |
| **11** | S03 | 2.2 | 22.8 |
| **12** | S03 | 4.4 | 27.4 |
| **13** | S04 | 0.55 | 2.2 |
| ⋮ | ⋮ | ⋮ | ⋮ |
| **49** | S18 | 2.2 | 15.4 |
| **50** | S18 | 4.4 | 19.0 |
| **51** | S18 | 6.6 | 19.4 |
| **52** | S19 | 2.2 | 29.0 |
| **53** | S19 | 4.4 | 34.0 |
| **54** | S19 | 6.6 | 37.0 |
| **55** | S20 | 2.2 | 22.2 |
| **56** | S20 | 4.4 | 29.0 |
| **57** | S20 | 6.6 | 32.2 |
| **58** | S21 | 2.2 | 23.0 |
| **59** | S21 | 4.4 | 27.4 |
| **60** | S21 | 6.6 | 30.4 |

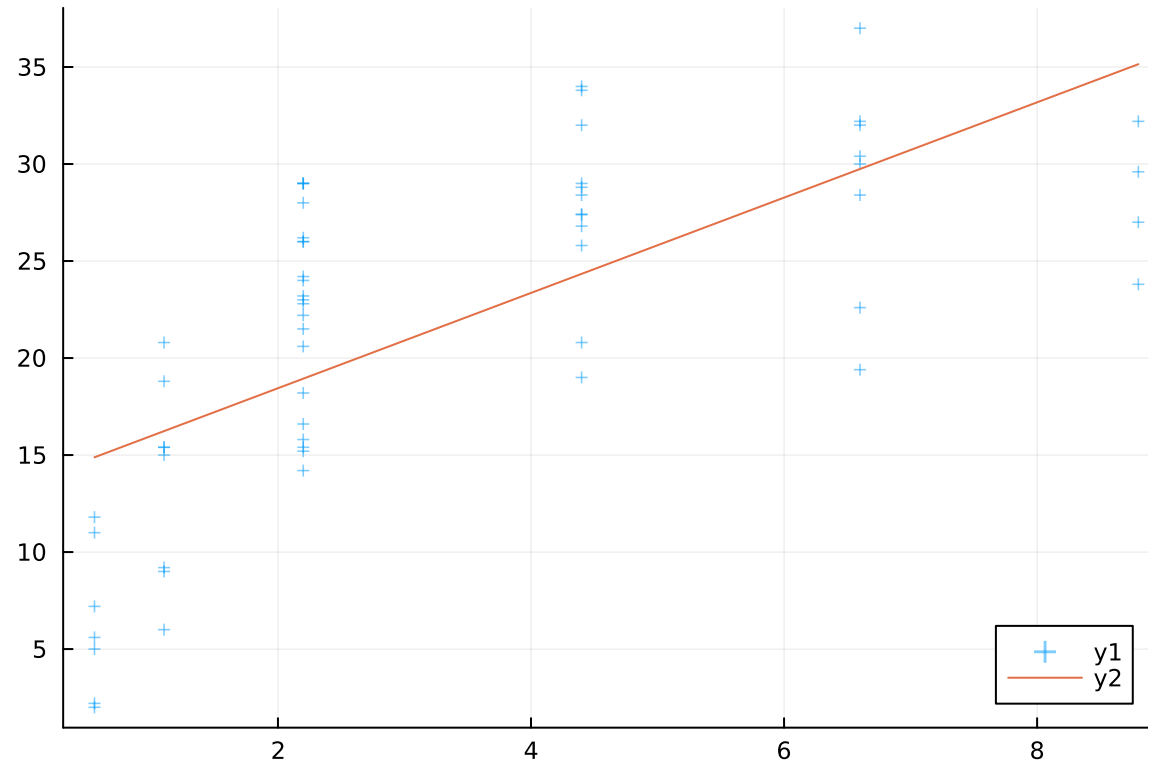
In [2]:
```
ols = lm(@formula(length ~ conc), df)
```

Out[2]:
```
StatsModels.TableRegressionModel{LinearModel{GLM.LmResp{Vector{Float64}}, GLM.D
ensePredChol{Float64, LinearAlgebra.CholeskyPivoted{Float64, Matrix{Float64}, V
ector{Int64}}}}, Matrix{Float64}}

length ~ 1 + conc

Coefficients:
```

| | Coef. | Std. Error | t | Pr(>\|t\|) | Lower 95% | Upper 95% |
|---|---|---|---|---|---|---|
| (Intercept) | 13.533 | 1.42292 | 9.51 | <1e-12 | 10.6847 | 16.3813 |
| conc | 2.4559 | 0.347849 | 7.06 | <1e-08 | 1.75961 | 3.1522 |

In [3]:
```
using Plots
scatter(df.conc, df.length, m = (0.5, :+, 3))
plot!(x-> 2.4559*x + 13.533)
```

Out[3]:




In [21]:
```
using LinearAlgebra

X = hcat(ones(size(df, 1)), df.conc)
y = df.length

Q, R = qr(X'*X)

# Это требуется не во всех случаях
# особенность в том что функция qr возвращает матрицу Q в "сжатом" формате Compa
# поэтому для простого применения ее надо перевести в "обычный" вид
Q = Matrix(Q)


#Rβ=Q'X'y

β = R \ Q' * X' * y
```

Out[21]:
```
2-element Vector{Float64}:
 13.532987487197701
  2.4559015816084973
```

In [23]:
```
y
```

Out[23]:
```
60-element Vector{Float64}:
 15.8
 20.8
 22.6
 23.8
 20.6
 26.8
 28.4
 27.0
  7.2
 15.4
 22.8
 27.4
  2.2
  ⋮
 15.4
 19.0
 19.4
 29.0
 34.0
 37.0
 22.2
 29.0
 32.2
 23.0
 27.4
 30.4
```

In [ ]:
```
```

In [5]:
```
y_hat = X * β

e  = y - y_hat

sum_sq = (e' * e)
```

Out[5]:
```
2383.732500333971
```

In [6]:
```
deviance(ols)
```

Out[6]:
```
2383.7325003339706
```

In [7]:
```
σ² = sum_sq/(length(e) - rank(X))
```

Out[7]:
```
41.098836212654675
```

In [8]:
```
β_vcov = σ² * inv(X' * X)
```

Out[8]:
```
2×2 Matrix{Float64}:
  2.02471   -0.402625
 -0.402625   0.120999
```

In [9]:
```
β_sd = sqrt.(diag(β_vcov))
```

Out[9]:
```
2-element Vector{Float64}:
 1.4229246124284716
 0.34784935383776977
```

In [10]:
```
deviance(ols)
```

Out[10]:
```
2383.7325003339706
```

Тоже самое если выполнять QR разложение для X

$$R\beta = Q^T y$$

In [11]:
```
#R β = Q^T y
Q, R = qr(X)
Q    = Matrix(Q)
β    = R \ Q'*y
```

Out[11]:
```
2-element Vector{Float64}:
 13.532987487197762
  2.455901581608487
```

Или без разложения с помощью нахождения обратной матрицы

$$\beta = (X^T X)^{-1} X^T y$$

In [12]:
```
β    = inv(X' * X) * X' * y
```

Out[12]:
```
2-element Vector{Float64}:
 13.532987487197751
  2.4559015816084884
```

Аналогично выполняется анализ двух и более факторов:

In [24]:
```
using CategoricalArrays

df = download("https://raw.githubusercontent.com/PharmCat/edu/main/csv/1freparma

transform!(df, :factor => categorical, renamecols=false)

df[1:3, :]
```

Out[24]: 3×4 DataFrame

| Row | subject | factor | response | time |
|---|---|---|---|---|
| | Float64 | Cat... | Float64 | Float64 |
| 1 | 1.0 | 1.0 | 46.951 | 1.0 |
| 2 | 1.0 | 1.0 | 46.3568 | 2.0 |
| 3 | 1.0 | 1.0 | 49.1365 | 3.0 |

In [14]:
```
ols = lm(@formula(response ~ time + factor + time * factor), df)
```

Out[14]:
```
StatsModels.TableRegressionModel{LinearModel{GLM.LmResp{Vector{Float64}}, GLM.D
ensePredChol{Float64, CholeskyPivoted{Float64, Matrix{Float64}, Vector{Int6
4}}}}, Matrix{Float64}}

response ~ 1 + time + factor + time & factor

Coefficients:
──────────────────────────────────────────────────────────────────────────────
──
                      Coef.  Std. Error      t  Pr(>|t|)   Lower 95%  Upper 9
5%
──────────────────────────────────────────────────────────────────────────────
──
(Intercept)         44.4231     0.745711  59.57    <1e-99  42.954      45.8922
time                 0.504845   0.120182   4.20    <1e-04   0.268078    0.7416
12
factor: 1.0          5.42936    1.05459    5.15    <1e-06   3.35174     7.5069
8
time & factor: 1.0   0.284751   0.169963   1.68    0.0952  -0.0500885   0.6195
89
──────────────────────────────────────────────────────────────────────────────
──
```

In [15]:
```
using CategoricalArrays

# ftdf3.csv

df = download("https://raw.githubusercontent.com/PharmCat/edu/main/csv/ftdf3.csv

transform!(df, :factor => categorical, renamecols=false)

df[1:3, :]
```

Out[15]: 3×9 DataFrame

| Row | subject | response | factor | r1 | r2 | s2 | factor2 | p | nrhoresp |
|---|---|---|---|---|---|---|---|---|---|
| | Float64 | Float64 | Cat... | String1 | String1 | Float64 | String1 | Int64 | Float64 |
| **1** | 1.0 | 36.2898 | 1.0 | A | D | 1.0 | G | 1 | 27.2595 |
| **2** | 1.0 | 35.2288 | 1.0 | A | D | 1.0 | G | 2 | 26.5341 |
| **3** | 1.0 | 34.7597 | 1.0 | A | D | 1.0 | G | 3 | 26.7089 |

### Д.3.

Провести анализ таблицы ftdf3.csv - изучить влияние факторов "r1" и "factor2" на переменную "response".

Изучить модель влияния факторов "factor" и "factor2" на переменную "response", какой из этих фактором значимо влияет на зависимую переменную?

В примере ниже выделены вектор ответов $y$ и матрица $X$, найти вектор $\beta$ и вариационнл-ковариационную матрицу $cov(\beta|X)$.

In [16]:
```
y = df.response
X = Matrix(df[:, [:p, :nrhoresp]])
```

```
nothing
```

- Что такое стандартизация?

Стандартизация - это приведение даннных к единому масшабу таким образом, что получаем следующую статистику распределения данных:

Среднее: $\mu = 0$

Стандартное отклонение: $\sigma = 1$

Используя следующий вид:

$z = \frac{x - \mu}{\sigma}$

Стандартизация используется для значений данных, которые имеют нормальное распределение. Кроме того, применяя стандартизацию, мы стремимся сделать среднее значение набора данных равным 0, а стандартное отклонение эквивалентным 1.

Таким образом, набор данных становится более очевидным и удобным для анализа, а оценки получаются несмещенными.

- Для чего применяется стандартизация

Для того чтобы наша модель работала хорошо, очень важно, чтобы данные имели одинаковый масштаб с точки зрения функции, чтобы избежать смещения в результате. Так, например, имея множество числовых независимых переменных, нам необходимо представить их в одном масштабе, чтобы сделать правильные выводы. В противном случае, разброс может получится максимальным, что усложнит выводы.

- Какие виды стандартизации бывают

Есть несколько способов стандартизации. Можно использовать функцию StandardScaler из библиотеки sklearn.preprocessing. Можно делать это по формуле, приводя среднее к нулю, а стандартное отклонение к единице.

- Что такое Ridge - регрессия?

Ридж-регрессия или гребневая регрессия (ridge regression) - это один из методов понижения размерности. Часто его применяют для борьбы с переизбыточностью данных, когда независимые переменные коррелируют друг с другом (т.е. имеет место мультиколлинеарность). Следствием этого является плохая обусловленность матрицы $X^T X$ и неустойчивость оценок коэффициентов регрессии. Оценки, например, могут иметь неправильный знак или значения, которые намного превосходят те, которые приемлемы из физических или практических соображений.

Особенностью данного вида регрессии является то, что теперь мы прибавляем к классической функции линейной регрессии штраф $\lambda \sum |\beta|$, где $\lambda$ - размер штрафа, а

$\beta$ - коэффициенты регрессии.

Чем меньше $\lambda$, тем выше дисперсия и ниже смещение. Чем больше $\lambda$, тем ниже дисперсия и выше смещение.

- Что такое Lasso - регрессия?

Почти идентична ридж-регрессии. В ней штраф — это сумма модулей значений коэффициентов.

Лассо-регрессия - это тип линейной регрессии, который использует сжатие (shrinkage). Сжатие - это когда значения данных сжимаются к центральной точке, например, к среднему значению. Процедура лассо поощряет простые, разреженные модели (т.е. модели с меньшим количеством параметров). Этот конкретный тип регрессии хорошо подходит для моделей, демонстрирующих высокий уровень мультиколлинеарности, или когда вы хотите автоматизировать определенные части выбора модели, такие как выбор переменных / исключение параметров.

Дом материалы по теме: https://timeseriesreasoning.com/contents/deep-dive-into-variance-covariance-matrices/

## Ссылки

- **Julia**
  - Ссылка: https://julialang.org/
- **Документация к основным пакетам Julia**
  - *Математика и анализ*
    - Roots.jl (нахождение корней) https://juliamath.github.io/Roots.jl/stable/
    - Optim.jl (Поиск минимума/максимума) https://julianlsolvers.github.io/Optim.jl/stable/#user/minimization/
    - ForwardDiff.jl (Дифференцирование) https://juliadiff.org/ForwardDiff.jl/stable/
    - QuadGK.jl (Численное интегрирование) https://juliamath.github.io/QuadGK.jl/stable/
    - DifferentialEquations.jl (Численное решение дифференциальных уравнений) https://diffeq.sciml.ai/stable/
  - *Статистика*
    - Distributions.jl (Основной пакет для работы с распределениями) https://juliastats.org/Distributions.jl/stable/
    - Statistics - Базовый пакет с основными статистическими функциями (не требует установки, но надо подключать `using Statistics` )

        - HypothesisTests.jl (основные статистические критерии) https://juliastats.org/HypothesisTests.jl/stable/
        - GLM.jl (Общая/обобщенная линейная модель) https://juliastats.org/GLM.jl/stable/
    - *Другое*
        - Plots.jl (Графики) https://docs.juliaplots.org
        - DataFrames.jl (Таблицы) https://dataframes.juliadata.org/stable/
        - CSV.jl (Импорт/экспорт CSV файлов) https://csv.juliadata.org/stable/
- **Литература**
    - Математика. Базовый курс / Б.Ш. Гулиян, Р.Я. Хамидуллин. Москва : Синергия, 2013. - 712 с. - ISBN 978-5-4257-0109-1.
    - Фадеева Л. Н., Лебедев А. В., Теория вероятностей и математическая статистика: учебное пособие. - 2-е изд., перераб. и доп. - М.: Эксмо, 2010. - 496 с.
    - The Matrix Cookbook
    - Линейная алгебра и ее применения - Стренг Г.
    - Матричный анализ - Хорн Р., Джонсон Ч.
- **Дополнительная литература:**
    - Математическая статистика в медицине, В. А. Медик, М. С. Токмачев, 978-5-279-03195-5
    - *Медико-биологическая статистика. Гланц. Пер. с англ. — М., Практика, 1998. — 459 с.*
    - *Байесовская статистика: Star Wars, LEGO, резиновые уточки и многое другое*
    - *Занимательная статистика. Манга. Син Такахаси, 2009, 224с, ISBN: 978-5-97060-179-2*

In [17]:
```
using Distributions

dist = Normal(20, 2.5)
p    = cdf(dist, 24) - cdf(dist, 21)
println("Нижн. гр: ", quantile(dist, 0.04))
println("Нижн. гр: ", quantile(dist, 0.96))

##############################################################################
#Нижн. гр: 15.623284821869575
#Нижн. гр: 24.376715178130425
ub = 20 + 2.5*3
```

```
lb = 20 + 2.5
p = 0.04
while true
    pv = ccdf(dist, lb)
    if abs(p-pv) < 0.0001 break end
    lb += 0.00001
end
println("Bound: ", lb)

################################################################################
#Нижн. гр: 15.623284821869575
#Нижн. гр: 24.376715178130425
ub = 20 + 2.5*3
lb = 20 + 2.5
p  = 0.04
if ccdf(dist, lb) > p && ccdf(dist, ub) < p
    println("Condition OK")
else
    println("Condition FALSE")
end
val = NaN
while true
    val = (lb+ub)/2
    pv  = ccdf(dist, val)
    if abs(p-pv) < 0.000000001 break end

    if pv < p
        ub = val
    else
        lb = val
    end
end
println("Bound: ", val)
```

```
Нижн. гр: 15.623284821869575
Нижн. гр: 24.376715178130425
Bound: 24.37381999992906
Condition OK
Bound: 24.376715198159218
```