Краткое введение в мир матричной алгебры

Линейные модели…

Вадим Хайтов, Марина Варфоломеева

Вы сможете

- Объяснить что такое матрицы и какие бывают их основные разновидности
- Выполнить базовые операции с матрицами с использованием функций R
- Применить в среде R методы матричной алгебры для решения простейших задач

Зачем нужны матрицы?

Матричные объекты

- Есть много типов объектов, для которых такое выражение оказывается наиболее естественным (изображения, описания многомерных объектов и т.д.)
- В матрицах, как и в обычных числах, скрыта информация, которую можно извлекать и преобразовывать по определенным правилам

Структура матриц

$$\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1c} \\ a_{21} & a_{22} & \cdots & a_{2c} \\ \vdots & \vdots & \ddots & \vdots \\ a_{r1} & a_{r2} & \cdots & a_{rc} \end{pmatrix}$$

Размер (порядок) матрицы $r \times c$

Разновидности матриц

Вектор-строка (row matrix)

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 \end{pmatrix}$$

Вектор-столбец (column matrix)

$$\mathbf{B} = \begin{pmatrix} 1 \\ 4 \\ 7 \\ 10 \end{pmatrix}$$

Разновидности матриц

Прямоугольные матрицы (rectangular matrices)

$$\mathbf{C} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \\ 10 & 11 & 12 \end{pmatrix}$$

$$\mathbf{D} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$$

В таком виде обычно представляются исходные данные

Квадратные матрицы (square matrices)

Это наиболее "операбельные" матрицы

$$\mathbf{E} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix}$$

Диагональные матрицы (diagonal matrix)

$$\mathbf{F} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 9 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Квадратные матрицы (square matrices)

Треугольные матрицы (triangular matrices)

$$\mathbf{H} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 0 & 5 & 6 & 7 \\ 0 & 0 & 9 & 10 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

или

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 3 & 5 & 0 & 0 \\ 4 & 7 & 9 & 0 \\ 5 & 8 & 10 & 11 \end{pmatrix}$$

Квадратные матрицы (square matrices)

Единичная матрица (identity matrix)

$$\mathbf{I} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Единичная матрица (обозначение $\mathbf{I}$) занимают особое место в матричной алгебре. Она выполняет ту же роль, которую выполняет единица в обычной алгебре.

Особенность квадратных матриц

Для квадратных матриц могут быть найдены (но не обязательно существуют) некоторые важные для матричной алгебры показатели: *определитель*, *инверсия*, *собственные значения* и *собственные вектора*

Задание

Создайте с помощью R следующие матрицы

```
##      [,1] [,2] [,3]
## [1,]    1    5    9
## [2,]    2    6   10
## [3,]    3    7   11
## [4,]    4    8   12
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    0    0    0    0
## [2,]    0    1    0    0    0
## [3,]    0    0    1    0    0
## [4,]    0    0    0    1    0
## [5,]    0    0    0    0    1
```

Операции с матрицами

Транспонирование матриц

```
A <- matrix(1:12, ncol = 3)
A
```

```
##      [,1] [,2] [,3]
## [1,]    1    5    9
## [2,]    2    6   10
## [3,]    3    7   11
## [4,]    4    8   12
```

Транспонированная матрица $\mathbf{A}$

```
B <- t(A)
B
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    2    3    4
## [2,]    5    6    7    8
## [3,]    9   10   11   12
```

Сложение матриц

```
A + 4
```

```
##      [,1] [,2] [,3]
## [1,]    5    9   13
## [2,]    6   10   14
## [3,]    7   11   15
## [4,]    8   12   16
```

```
A + A
```

```
##      [,1] [,2] [,3]
## [1,]    2   10   18
## [2,]    4   12   20
## [3,]    6   14   22
## [4,]    8   16   24
```

Но! Нельзя складывать матрицы разных размеров

```
A + B
```

Биологическое приложение

Предположим, что мы подсчитывали двумя разными методами крупных и мелких животных трех видов в одних и тех же пробах

```
##         Sp1 Sp2 Sp3
## Sample1  10   6   8
## Sample2  13  16   5
## Sample3   8   9  11
## Sample4   9  11  13
## Sample5  10  14   9
```

```
##         Sp1 Sp2 Sp3
## Sample1  52  46  55
## Sample2  45  53  47
## Sample3  57  50  42
## Sample4  48  45  56
## Sample5  48  58  47
```

Биологическое приложение

Общее обилие

```
Large + Small
```

```
##          Sp1 Sp2 Sp3
## Sample1  62  52  63
## Sample2  58  69  52
## Sample3  65  59  53
## Sample4  57  56  69
## Sample5  58  72  56
```

Простое умножение

Умножение на число

```
A * 4
```

```
##      [,1] [,2] [,3]
## [1,]    4   20   36
## [2,]    8   24   40
## [3,]   12   28   44
## [4,]   16   32   48
```

Простое умножение матрицы на вектор возможно только если число элементов в векторе равно числу строк в матрице

```
A * c(10, 11, 12, 13)
```

```
##      [,1] [,2] [,3]
## [1,]   10   50   90
## [2,]   22   66  110
## [3,]   36   84  132
## [4,]   52  104  156
```

Все элементы первой строки матрицы умножаются на первый элемент вектора, все элементы второй строки на второй элемент вектора и т.д.

Биологическое применение

Допустим, учет организмов в части описаний проходил не на всей выборке, а лишь в ее части.

```
Rpocessed_portion <- c(1, 1, 1/2, 1/3, 1/4)
Processed_Factor <- 1/Rpocessed_portion

Small * Processed_Factor

##         Sp1 Sp2 Sp3
## Sample1  52  46  55
## Sample2  45  53  47
## Sample3 114 100  84
## Sample4 144 135 168
## Sample5 192 232 188
```

Скалярное произведение векторов

Допустимо только для векторов одинаковой размерности

$$\mathbf{a} \bullet \mathbf{b} = \begin{pmatrix} a_1 \\ a_3 \\ a_4 \\ a_5 \\ a_6 \\ a_7 \end{pmatrix} \times \begin{pmatrix} b_1 & b_3 & b_4 & b_5 & b_6 & b_7 \end{pmatrix} = x$$

Результат этой операции - число (скаляр)

Биологическое применение

Сколько особей родится в популяции, если мы знаем репродуктивные характеристики всех возрастных групп?

$$\begin{pmatrix} N_1 \\ N_3 \\ N_4 \\ N_5 \\ N_6 \\ N_7 \end{pmatrix} \times \begin{pmatrix} F_1 & F_2 & F_3 & F_4 & F_5 & F_6 & F_7 \end{pmatrix}$$

```
N <- c(20, 40, 32, 45, 80, 50, 10)
Fert <- c( 0,  0,   1,   2,   2,   0,   0)

t(N) %*% (Fert)
```

```
##       [,1]
## [1,]  282
```

Умножение матриц

Умножать можно только в том случае, если число строк одной матрицы равно числу столбцов другой матрицы

```
A %*% B
```

```
##      [,1] [,2] [,3] [,4]
## [1,]  107  122  137  152
## [2,]  122  140  158  176
## [3,]  137  158  179  200
## [4,]  152  176  200  224
```

```
A %*% t(A)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]  107  122  137  152
## [2,]  122  140  158  176
## [3,]  137  158  179  200
## [4,]  152  176  200  224
```

НО! Нельзя произвести такое умножение

```
A %*% A
```

Биологическое применение

Простейший пример использования умножения матриц - построение модели динамики демографической структуры популяции Для вычислений необходим начальный *демографический вектор* и *матрица Лесли*

$$\begin{pmatrix} F_1 & F_2 & F_3 & F_4 & F_5 & F_6 & F_7 \\ P_{1-2} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & P_{2-3} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & P_{3-4} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & P_{4-5} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & P_{5-6} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & P_{6-7} & 0 \end{pmatrix} \times \begin{pmatrix} N1_t \\ N3_t \\ N4_t \\ N5_t \\ N6_t \\ N7_t \end{pmatrix} = \begin{pmatrix} N1_{t+1} \\ N3_{t+1} \\ N4_{t+1} \\ N5_{t+1} \\ N6_{t+1} \\ N7_{t+1} \end{pmatrix}$$

Простейшая демографическая модель

Демографический вектор в момент времени t

```
##      Age T1
## 1      0 20
## 2   1-10 40
## 3  11-20 32
## 4  21-35 45
## 5  36-45 80
## 6  46-55 50
## 7  56-65 10
```

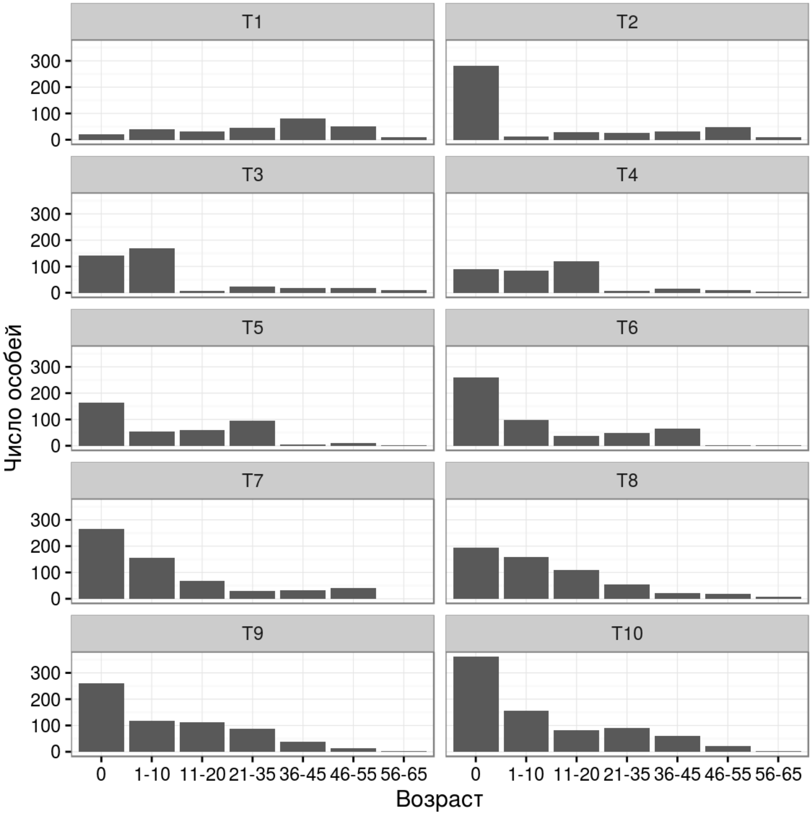
Матрица Лесли

```
##      [,1] [,2] [,3] [,4] [,5] [,6] [,7]
## [1,]  0.0  0.0  1.0  2.0  2.0  0.0    0
## [2,]  0.6  0.0  0.0  0.0  0.0  0.0    0
## [3,]  0.0  0.7  0.0  0.0  0.0  0.0    0
## [4,]  0.0  0.0  0.8  0.0  0.0  0.0    0
## [5,]  0.0  0.0  0.0  0.7  0.0  0.0    0
## [6,]  0.0  0.0  0.0  0.0  0.6  0.0    0
## [7,]  0.0  0.0  0.0  0.0  0.0  0.2    0
```

Демографическая струкутра в момент времени $t+1$

```
Pop$T2 <- as.vector( Lesl %*% (Pop$T1 ))
Pop$T3 <- as.vector( Lesl %*% (Pop$T2 ))
Pop$T4 <- as.vector( Lesl %*% (Pop$T3 ))
Pop$T5 <- as.vector( Lesl %*% (Pop$T4 ))
Pop$T6 <- as.vector( Lesl %*% (Pop$T5 ))
Pop$T7 <- as.vector( Lesl %*% (Pop$T6 ))
Pop$T8 <- as.vector( Lesl %*% (Pop$T7 ))
Pop$T9 <- as.vector( Lesl %*% (Pop$T8 ))
Pop$T10 <- as.vector( Lesl %*% (Pop$T9 ))
```

Демографическая струкутра в момент времени $t + 1$



Вычисление корреляций через произведение матриц

Используем известные нам данные по размеру головного мозга

```
brain <- read.csv("data/IQ_brain.csv", header = TRUE)
br <- brain[complete.cases(brain), -1]
br <- as.matrix(br)
br_scaled <- scale(br) #Стандартизуем значения

cor_matrix <- t(br_scaled) %*% br_scaled / (nrow(br_scaled) - 1)

cor_matrix
```

```
##                FSIQ     VIQ      PIQ   Weight  Height MRINACount
## FSIQ         1.0000  0.9451  0.93443 -0.05148 -0.1184      0.334
## VIQ          0.9451  1.0000  0.77602 -0.07609 -0.1190      0.300
## PIQ          0.9344  0.7760  1.00000  0.00251 -0.0932      0.378
## Weight      -0.0515 -0.0761  0.00251  1.00000  0.6996      0.513
## Height      -0.1184 -0.1190 -0.09316  0.69961  1.0000      0.588
## MRINACount   0.3337  0.3003  0.37778  0.51338  0.5884      1.000
```

Некоторые свойства произведения матриц

1. Если существует произведение матриц $\mathbf{BC}$, то не обязательно существует $\mathbf{CB}$

```
B <- matrix(1:24, ncol = 4)
C <- matrix(1:12, ncol = 3)

B %*% C
```

```
##      [,1] [,2] [,3]
## [1,]  130  290  450
## [2,]  140  316  492
## [3,]  150  342  534
## [4,]  160  368  576
## [5,]  170  394  618
## [6,]  180  420  660
```

НО!

```
C %*% B
```

Такое произведение невозможно

Некоторые свойства произведения матриц

1. Всегда существует такое произведение матриц $\mathbf{CC'}$ и $\mathbf{C'C}$

```
C %*% t(C)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]  107  122  137  152
## [2,]  122  140  158  176
## [3,]  137  158  179  200
## [4,]  152  176  200  224
```

```
t(C) %*% C
```

```
##      [,1] [,2] [,3]
## [1,]   30   70  110
## [2,]   70  174  278
## [3,]  110  278  446
```

Некоторые свойства произведения матриц

1. Произведение матриц $\mathbf{BC}$ как правило не равно $\mathbf{CB}$

```
B <- matrix(1:9, ncol = 3)
C <- matrix(11:19, ncol = 3)

B %*%  C
```

```
##      [,1] [,2] [,3]
## [1,]  150  186  222
## [2,]  186  231  276
## [3,]  222  276  330
```

```
C %*% B
```

```
##      [,1] [,2] [,3]
## [1,]   90  216  342
## [2,]   96  231  366
## [3,]  102  246  390
```

Некоторые свойства произведения матриц

1. $[\mathbf{BC}]' = \mathbf{C}'\mathbf{B}'$

```
t(B %*% C)
```

```
##      [,1] [,2] [,3]
## [1,]  150  186  222
## [2,]  186  231  276
## [3,]  222  276  330
```

```
t(C) %*% t(B)
```

```
##      [,1] [,2] [,3]
## [1,]  150  186  222
## [2,]  186  231  276
## [3,]  222  276  330
```

Некоторые свойства произведения матриц

1. Произведение $\mathbf{BB'}$ и $\mathbf{B'B}$ всегда дает симметричную матрицу

```
B %*% t(B)
```

```
##      [,1] [,2] [,3]
## [1,]   66   78   90
## [2,]   78   93  108
## [3,]   90  108  126
```

```
t(B) %*% B
```

```
##      [,1] [,2] [,3]
## [1,]   14   32   50
## [2,]   32   77  122
## [3,]   50  122  194
```

Определитель матрицы

Определитель матрицы - это некоторое число.

По значению этого числа можно *определить* есть ли у матрицы некоторые свойства (например, обратима ли матрица).

Определитель бывает только у квадратных матриц.

Матрицы, имеющие определитель равный нулю, называются *сингулярными* матрицами.

```
det(B)
```

```
## [1] 0
```

```
BB <- t(B) %*% B
det(BB)
```

```
## [1] 0
```

Обращение (инверсия) матриц

В матричной алгебре нет процедуры деления. Вместо нее используют обращение матриц.

$$\mathbf{X}^{-1}\mathbf{X} = \mathbf{I}$$

Произведение инверсии матрицы и исходной матрицы дает единичную матрицу

Обращение (инверсия) матриц

Только квадратные матрицы, имеющие определитель неравный нулю, могут иметь обратную матрицу.

Поэтому для квадратных матриц справедливо $\mathbf{X}\mathbf{X}^{-1} = \mathbf{X}^{-1}\mathbf{X}$

Решение в среде R

Создадим матрицу

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
## [3,]    7    8   10
```

Ее определитель

```
det(X)
```

```
## [1] -3
```

Решение в среде R

Обратная матрица

```
solve(X)
```

```
##        [,1]  [,2] [,3]
## [1,] -0.667 -1.33    1
## [2,] -0.667  3.67   -2
## [3,]  1.000 -2.00    1
```

По определению, $\mathbf{X}^{-1}\mathbf{X} = \mathbf{I}$

```
round(solve(X) %*% X )
```

```
##      [,1] [,2] [,3]
## [1,]    1    0    0
## [2,]    0    1    0
## [3,]    0    0    1
```

Применение обращенных матриц

Решение систем линейных уравнений

Простейший случай использования обратных матриц - решение систем линейных уравнений

$$\begin{cases} 1x + 2y + 3z = 2 \\ 4x + 5y + 6z = 4 \\ 7x + 8y + 10z = 10 \end{cases}$$

Эту систему можно представить в матричном виде

$$\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 10 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 2 \\ 4 \\ 10 \end{pmatrix}$$

Тогда

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 10 \end{pmatrix}^{-1} \begin{pmatrix} 2 \\ 4 \\ 10 \end{pmatrix}$$

Задание

Решите приведенную систему уравнений с использованием матричной алгебры

$$\begin{cases} 1x + 2y + 3z = 2 \\ 4x + 5y + 6z = 4 \\ 7x + 8y + 10z = 10 \end{cases}$$

Решение

```
Coef <- matrix(c(1 , 2 , 3 ,
         4 , 5 , 6 ,
         7 , 8 , 10), byrow = T, ncol = 3)
Val <- c(2,4,10)

solve(Coef) %*% Val
```

```
##       [,1]
## [1,]  3.33
## [2,] -6.67
## [3,]  4.00
```

Подбор параметров линейной регрессии методом наименьших квадратов с использованием матричной алгебры

Линейная регрессия в матричном виде

При подборе коэффициентов методом наименьших квадратов нам надо решить следующее матричное уравнение:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta}$$

Здесь

$\mathbf{y}$ - вектор предсказанных значений

$\mathbf{X}$ - модельная матрица $\begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix}$

$\boldsymbol{\beta}$ - вектор коэффициентов модели

Решение этого уравнения

Умножим обе части уравнения на транспонированную матрицу $\mathbf{X}'$

$$\mathbf{X}'\mathbf{y} = \mathbf{X}'\mathbf{X}\boldsymbol{\beta}$$

Матрица $\mathbf{X}'\mathbf{X}$ - это всегда квадратная матрица. Ее можно обратить.

Тогда

$$\boldsymbol{\beta} = [\mathbf{X}'\mathbf{X}]^{-1}[\mathbf{X}'\mathbf{y}]$$

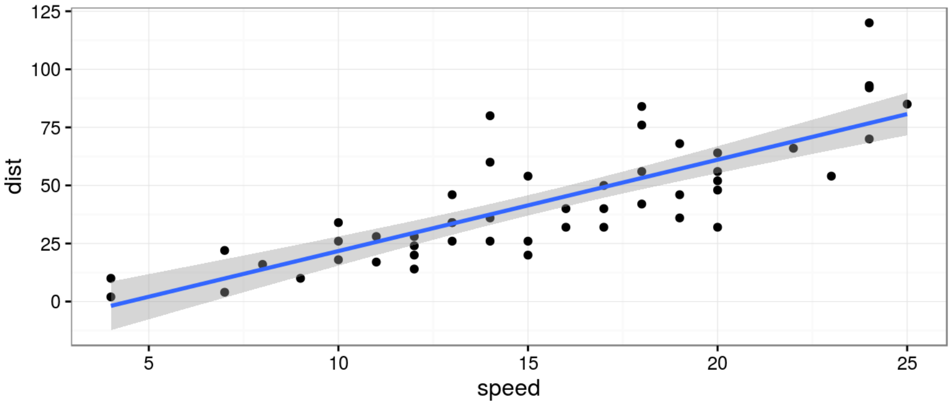
Подбираем коэффициенты с помощью фунции `lm()`

```
data(cars)
Mod <- lm(dist ~ speed, data = cars)
coefficients(Mod)
```

```
## (Intercept)       speed
##      -17.58        3.93
```

Графическое отражение, построенное с помощью `geom_smooth()`

```
library(ggplot2)
theme_set(theme_bw())
ggplot(cars, aes(x = speed, y = dist)) + geom_point() + geom_smooth(method = "lm"
```



Вычисление коэффициентов линейной регрессии вручную

Находим вектор коэффициентов на основе уравнения $\boldsymbol{\beta} = [\mathbf{X}'\mathbf{X}]^{-1}[\mathbf{X}'\mathbf{y}]$

```
X <- model.matrix(~speed, data = cars)
Y <- cars$dist
betas <- solve(t(X) %*% X) %*% (t(X) %*% Y)
betas
```

```
##                [,1]
## (Intercept) -17.58
## speed         3.93
```

Вычисление вариационно-ковариационной матрицы

Подобранные параметры - это лишь *оценки* некоторых параметров, описывающих связь между зависимой переменной и предиктором в популяции.

Варьирование параметров описывает *вриацонно-ковариационная матрица*.

В среде `R`, если модель задана, например, с помощью функции `lm()`, эта матрица вычисляется так

```
vcov(Mod)
```

```
##             (Intercept)  speed
## (Intercept)       45.68 -2.659
## speed             -2.66  0.173
```

Вычисление вариационо-ковариационной матрицы вручную

$$\mathbf{V}(\boldsymbol{\beta}) = s^2[\mathbf{X}'\mathbf{X}]^{-1}$$

где

$$s^2 = \frac{\sum_{i=1}^{n} e_i^2}{n-k}$$

$\sum_{i=1}^{n} e_i^2$ - сумма квадратов остатков
n - объем выборки
k - число параметров в модели

Вычисление вариационо-ковариационной матрицы вручную

Вычисляем $\sum\limits_{i=1}^{n} e_i^2$

```
predict_values <- X %*% betas

resid_values <- cars$dist - predict_values

s2 <- sum(resid_values^2)/(length(resid_values) - length(betas))

s2
```

```
## [1] 237
```

Вычисление вариационо-ковариационной матрицы вручную

Вычисляем вариационно-ковариационную матрицу

$$\mathbf{V}(\boldsymbol{\beta}) = s^2 [\mathbf{X}' \mathbf{X}]^{-1}$$

```
covbetas <- s2 * solve(t(X) %*% X)
covbetas
```

```
##             (Intercept)  speed
## (Intercept)       45.68 -2.659
## speed             -2.66  0.173
```

Применение матричной алгебры для построения графиков регрессионных моделей

Шаг 1. Формируем искусственный датасет со всеми возможными значениями предиктора

```
MyData <- data.frame(speed = seq(min(cars$speed), max(cars$speed)))
head(MyData)
```

```
##   speed
## 1     4
## 2     5
## 3     6
## 4     7
## 5     8
## 6     9
```

Применение матричной алгебры для построения графиков регрессионных моделей

Шаг 2. Формируем модельную матрицу для искусственно созданных данных

```
X <- model.matrix( ~ speed, data = MyData)
head(X)
```

```
##   (Intercept) speed
## 1           1     4
## 2           1     5
## 3           1     6
## 4           1     7
## 5           1     8
## 6           1     9
```

Применение матричной алгебры для построения графиков регрессионных моделей

Шаг 3. Вычисляем предсказанные значения для искусственно созданных данных

```
MyData$predicted <- X %*% betas
```

Применение матричной алгебры для построения графиков регрессионных моделей

Шаг 5. Вычисляем границы доверительных интервалов

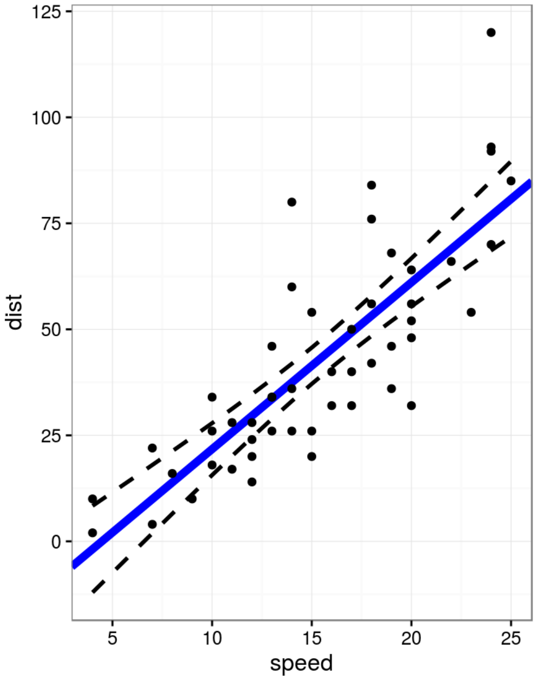
```
# Вычисляем стандартные отшибки путем перемножения матриц
  MyData$se <- sqrt(diag(X %*% covbetas %*% t(X)))

# Вычисляем доверительные интервалы
MyData$CiUp  <- MyData$predicted + 1.96 *MyData$se

MyData$CiLow  <- MyData$predicted - 1.96 *MyData$se
```

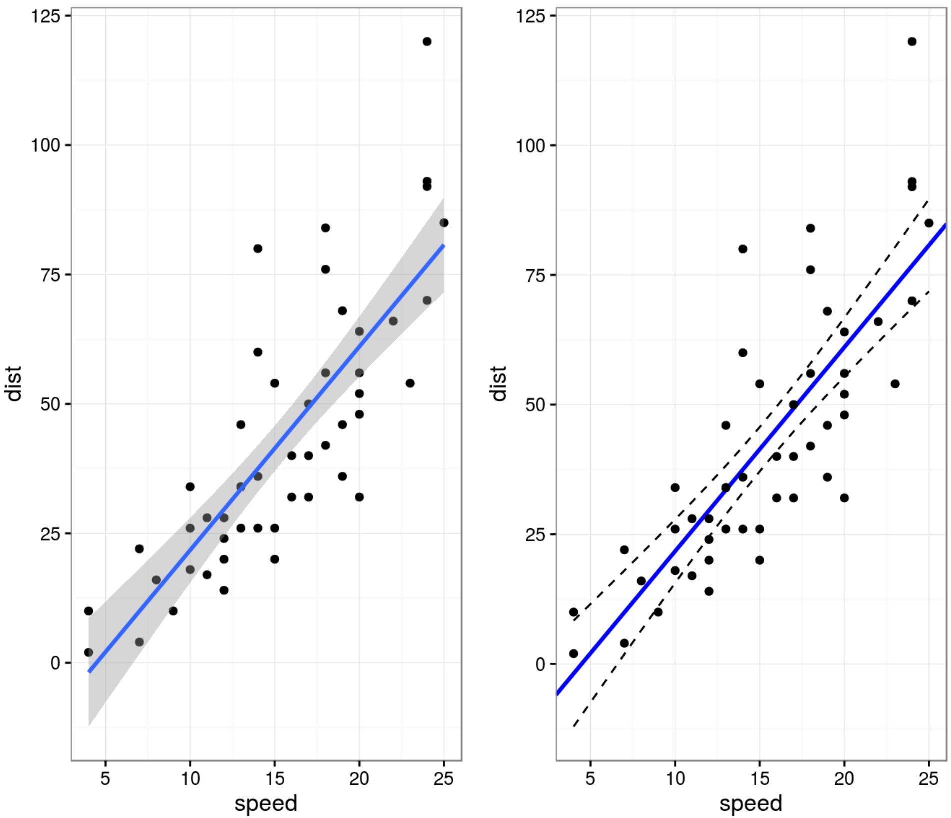
Применение матричной алгебры для построения графиков регрессионных моделей

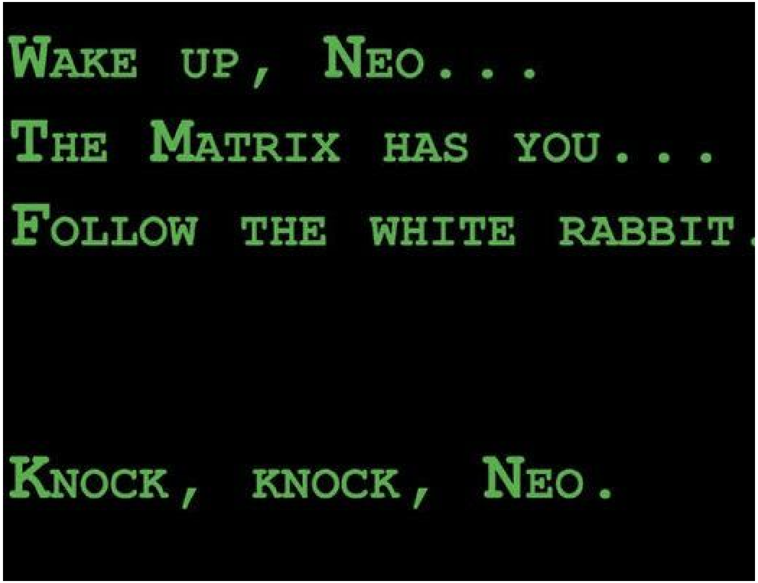
Шаг 6. Строим график




```
ggplot(MyData, aes(x = speed, y = predicted)) +
  geom_line(aes(x = speed, y = CiUp),
            linetype = 2, size = 1) +
  geom_line(aes(x = speed, y = CiLow),
            linetype = 2, size = 1)+
  geom_abline(slope = betas[2], intercept = betas[1],
              color = "blue", size=2) +
  geom_point(data = cars, aes(x = speed, y = dist)) +
  ylab("dist")
```

Сравним результаты




WAKE UP, NEO...
THE MATRIX HAS YOU...
FOLLOW THE WHITE RABBIT.
KNOCK, KNOCK, NEO.

Not The End

Что почитать

- Legendre P., Legendre L. (2012) Numerical ecology. Second english edition. Elsevier, Amsterdam. Глава 2. Matrix algebra: a summary.