# Linear regression models with linear algebra and R

Juho Kopra

University of Eastern Finland, School of Computing Partly based on material of https://github.com/genomicsclass/labs

2023-02-16

## 1. Scatter plot and one linear predictor

## 1. Scatter plot and one linear predictor

Let's consider following data set, and draw a scatter plot out of it.

The scatterplot becomes

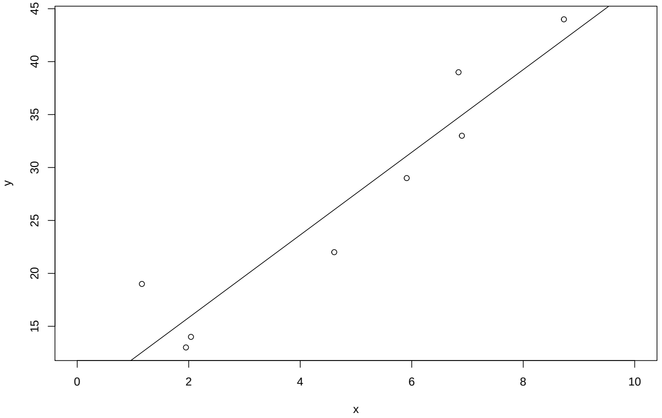


Figure 1: Scatterplot and a linear model with one predictor.

The coefficients of a linear regression fitted to that data set are $\beta_0 = 8.01$ and $\beta_1 = 3.90$.

## Definition

- Importantly, a linear regression model with one variable is a statistical model which can be formulated as
$$Y_i = \beta_0 + \beta_1 x_i + \varepsilon_i \tag{1}$$
and where $i = 1, \ldots, n$ and $\varepsilon_i \sim N(0, \sigma^2)$.
- There the following terminology takes place
  - $Y_i$ is dependent
  - $\beta_0$ is an intercept
  - $\beta_1$ is a slope
  - $\beta_0$ and $\beta_1$ are regression coefficients or model parameters
  - $\varepsilon_i$ is an error term
  - $x_i$ is the value of predictor aka independent aka covariate aka regressor

- In equation (1) we did not write estimated values of parameters down to the equation but used $\beta$-coefficients instead. When we write it down with $\beta$-coefficients we call it general form of linear model.
- Linear model with multiple predictors $X_1, X_2, \dots, X_k$ it can be written as

$$Y_i = \beta_0 + \beta_1 x_{i1} + \cdots \beta_k x_{ik} + \varepsilon_i \tag{2}$$

and where $i = 1, \dots, n$ and $\varepsilon_i \sim N(0, \sigma^2)$.

## The fitting of linear model

- We want to minimize the sum of squares of error terms $\varepsilon_i^2$, which can be written as $\varepsilon_i = Y_i - (\beta_0 + \beta_1 X_i)$. Note that error terms are in the same direction with $Y$-axis (and thus the same direction with $Y_i$).
- We will not go into technical details of minimization here.
- Using R we can fit the above model as follows. Assume that we have already read in data into object `dat`. We use `lm` function to fit the **l**inear **m**odel. The resulting calculation we store into object named `m1`. The data object `dat` has columns named `x` and `y` but the names could be something else as well. The first argument of `lm` is formula, which is special type in R. Formula is handy for fitting many types of models. On the left hand side of the formula one writes the dependent (e.g. what is $Y_i$), then tilde ($\sim$) and then predictor variable(s). For multiple predictors one need to put a plus sign (+) between the predictors.

```r
m1 <- lm(y ~ x, data=dat)
m1
```

```
##
## Call:
## lm(formula = y ~ x, data = dat)
##
## Coefficients:
## (Intercept)            x
##        8.01         3.90
```

## 2. Properties of linear models and interpretation

## Properties of linear models

To have an interpretation for $\beta_0$ and $\beta_1$ we note that:

Expected value of $Y_i$ is if we consider $x_i$ being known and fixed:

$$\begin{aligned} E(Y_i|x_i) &= E(\beta_0 + \beta_1 x_i + \varepsilon_i) \\ &= E(\beta_0) + E(\beta_1 x_i) + E(\varepsilon_i) \\ &= E(\beta_0) + x_i E(\beta_1) + E(\varepsilon_i) \\ &= \beta_0 + \beta_1 x_i \end{aligned}$$

because $x_i$ is constant and $E(\varepsilon_i) = 0$.

Shortly put, the above calculation gives $E(Y_i|x_i) = \beta_0 + \beta_1 x_i$, which is called a systematic part of linear model.

- If we replace the $\beta$-notation with their estimates, then we can use that equation to calculate fitted values:
$$\hat{Y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$$
Fitted values are calculated for the same $x_i$ values which were in the original data.

```
fitted(m1)
```

```
##      1      2      3      4      5      6      7      8
## 34.716 42.095 34.951 12.541 15.625 26.010 15.976 31.085
```

- Prediction can be obtained by the same equation but using any value. Here we predict new values using `x=10` and `x=11`.

```
predict(m1,newdata = data.frame(x=c(10,11)))
```

```
##      1      2
## 47.053 50.958
```

## Interpretation

- We want to interpret what happens in the population of our study. That is why we use equation of expected value of $Y_i$, that is $E(Y_i|x_i) = \beta_0 + \beta_1 x_i$ to give interpretation.
- An interpretation of $\beta_1$ becomes available as

$$\begin{aligned} E(Y_i|x_i+1) - E(Y_i|x_i) &= (\beta_0 + \beta_1(x_i+1)) - (\beta_0 + \beta_1 x_i) \\ &= \beta_1(x_i+1) - \beta_1 x_i \\ &= \beta_1 x_i + \beta_1 \cdot 1 - \beta_1 x_i \\ &= \beta_1 \end{aligned}$$

- Thus, if e.g. $\beta_1 = 3.90415$ then the expected value of $Y_i$ increases by 3.90415 units of y if $x_i$ becomes increased with 1 unit of x.
- An interpretation of intercept term $\beta_0$ becomes available if we set $x_i = 0$ since that removes $\beta_1$ from the equation. Thus $E(Y_i|x_i = 0) = \beta_0 + \beta_1 \cdot 0 = \beta_0$.

## 3. Categorical predictors

## 3. Categorical predictors

The predictors may as well be categorical ones. For example, we can model the mice weights where mice have two groups in a study. We are used to do that with a t-test but here we formulate it using linear regression. The results are the very same as with t-test!

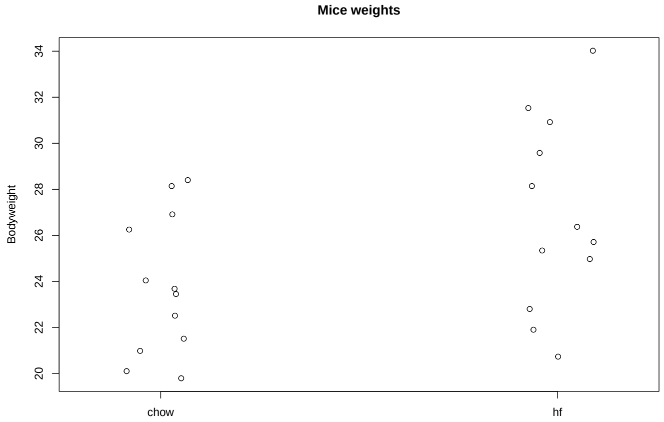


Figure 2: Mice weight with jitter. Example borrowed from Mike Love.

```
m1 <- lm(Bodyweight ~ Diet, data=dat)
```

```
summary(m1)
```

## Dummy variables and indicator function

Let's familiariase ourselves with dummy variables. Dummy variable is a variable which can have either value 0 or 1. For the example above, the dummy variable can be used to indicate that diet of a mouse is `hf` instead of `chow`. Thus, let's create a dummy variable so that it takes a value 1 if diet is `hf` and value 0 if diet is `chow`.

In mathematical formulations, a useful way of reminding ourselves that a predictor is a dummy variable is to use an indicator function. Indicator function $\mathbf{1}(x)$ takes value 1 if the condition `x` is true, and value 0 otherwise. The notation of indicator variable nicely reminds us that it will give value 1 or 0.

Indicator function is

$$\mathbf{1}(x) = \begin{cases} 1 & \text{if condition } x \text{ is true} \\ 0 & \text{otherwise} \end{cases}$$

For example we may write

$$Y_i = \beta_0 + \beta_1 \mathbf{1}(X_i = \texttt{hf}) + \varepsilon_i$$

where $Y_i$ and $x_i$ stand for values of `Bodyweight` and `Diet` respectively, and $i = 1, \ldots, n$ and $\varepsilon_i \sim N(0, \sigma^2)$. In the above, the $\mathbf{1}(X_i = \texttt{hf})$ gives a value 1 if `Diet` is `hf`. Otherwise it gives 0.

Thus, the expected value of $Y_i$ is $\beta_0$ if diet is `chow`. If the diet is `hf`, the expected values of $Y_i$ is $\beta_0 + \beta_1$.

## 4. Basics of linear algebra

## 4. Basics of linear algebra

Linear algebra is a branch of mathematics where instead of numbers (aka scalars) we operate on vectors and matrices. Linear algebra is a useful tool for solving systems of linear equations, and thus suitable for handling mathematics behind linear regression. Next, we go through some of the key concepts so that we can move forward to understand how linear regression can be solved.

## Transpose

Transpose of matrix $\mathbf{A}$ is denoted as $\mathbf{A}^T$

$$\mathbf{A} = \begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \end{pmatrix} \implies \mathbf{A}^T = \begin{pmatrix} 1 & 3 & 5 \\ 2 & 4 & 6 \end{pmatrix}$$

Above matrix $A$ is $n$ times $p$ matrix, where $n = 2, p = 3$.

```
A <- matrix(c(1,2,
              3,4,
              5,6),nrow=3,ncol=2,byrow=T)
A
```

```
##      [,1] [,2]
## [1,]    1    2
## [2,]    3    4
## [3,]    5    6
```

```
t(A)
```

```
##      [,1] [,2] [,3]
## [1,]    1    3    5
## [2,]    2    4    6
```

## Vector

A matrix with only one column is called a vector.

$$d = \begin{pmatrix} 1 \\ 3 \\ 5 \end{pmatrix}$$

A transpose of a vector is a row vector.

$$d^T = \begin{bmatrix} 1 & 3 & 5 \end{bmatrix}$$

## Cross product

A cross product (aka matrix multiplication) of row vector with a column vector is

$$\begin{bmatrix} 1 & 2 & 3 \end{bmatrix} \begin{pmatrix} 4 \\ 5 \\ 6 \end{pmatrix} = \begin{bmatrix} 1 \cdot 4 + 2 \cdot 5 + 3 \cdot 6 \end{bmatrix} = \begin{bmatrix} 32 \end{bmatrix}$$

```r
d <- c(4,5,6)
d
```

```
## [1] 4 5 6
```

```r
t(d)
```

```
##      [,1] [,2] [,3]
## [1,]    4    5    6
```

```r
t(t(d))
```

```
##      [,1]
## [1,]    4
## [2,]    5
## [3,]    6
```

```r
c(1,2,3) %*% d
```

```
##      [,1]
## [1,]   32
```

## Identity matrix

Identity matrix is $\mathbf{I}$ has 1 on diagonal and 0 elsewhere. E.g.

$$\mathbf{I}_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

```
diag(3)
```

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

## Square matrix and inverse of a matrix

If matrix has the same number of rows and columns, it is called a square matrix.

If a square matrix **B** has determinant, which is not zero, then **B** is invertible, which means that it is possible to calculate an inverse of matrix **B** that is $\mathbf{B}^{-1}$.

```
# Let's create a square matrix
B <- t(A) %*% A
B
```

```
##      [,1] [,2]
## [1,]   35   44
## [2,]   44   56
```

```
det(B) #determinant is not zero => invertible
```

```
## [1] 24
```

```
solve(B)
```

```
##         [,1]    [,2]
## [1,]  2.3333 -1.8333
## [2,] -1.8333  1.4583
```

For invertible matrix **B** it follows that $\mathbf{B}\mathbf{B}^{-1} = \mathbf{I}$ and also $\mathbf{B}^{-1}\mathbf{B} = \mathbf{I}$.

## Solving systems of equations using linear algebra

Matrices can be used to represent a systems of equations. To understand the basic concepts, let's have a look at following system of equations:

$$\begin{aligned} a+b+c&=6\\ 3a-2b+c&=2\\ 2a+b-c&=1 \end{aligned}$$

$$\begin{pmatrix}1 & 1 & 1\\ 3 & -2 & 1\\ 2 & 1 & -1\end{pmatrix}\begin{pmatrix}a\\ b\\ c\end{pmatrix}=\begin{pmatrix}6\\ 2\\ 1\end{pmatrix}$$

Taking the inverse of square matrix and multiplying the equation from the left hand side with that solves the system.

$$\implies \begin{pmatrix} a \\ b \\ c \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 3 & -2 & 1 \\ 2 & 1 & -1 \end{pmatrix}^{-1} \begin{pmatrix} 6 \\ 2 \\ 1 \end{pmatrix}$$

The same as above with R

```
A <- matrix(c(1, 1, 1,
              3,-2, 1,
              2, 1,-1),nrow=3,ncol=3,byrow=T)
d <- c(6,2,1)
solve(A) %*% d
```

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

## 5. Linear models formulated with linear algebra

## 5. Linear models formulated with linear algebra

Let's consider the previous example of two mice populations in Chapter 3. We have already seen how linear regression connects nicely with t-tests. We will now go through how it can be formulated using linear algebra.

Let $x_i = \mathbf{1}(\texttt{Diet = hf})$ which means that $x_i = 1$ if diet is `hf`, and $x_i = 0$ if diet is `chow`.

Lets define notation of matrices and vectors

$$\mathbf{Y} = \begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{pmatrix}, \mathbf{X} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \\ 1 & x_n \end{pmatrix}, \boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} \text{ and } \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

So we can write linear regression using the notation above as

$$\begin{pmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{pmatrix} = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \\ 1 & x_n \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} + \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}$$

and we can use matrix notation, so the above can be written as

$$\mathbf{Y} = \mathbf{X}\beta + \varepsilon$$

$$\mathbf{Y} = \mathbf{X}\beta + \varepsilon$$

The **X** is what we call a design matrix. We can show design matrix in R from an object of `lm` by:

```r
model.matrix(~ Diet, data=dat)
# or
model.matrix(m1)
```

## 6. Contrasts

## 6. Contrasts

As a running example to learn about more complex linear models, we will be using a dataset which compares the different frictional coefficients on the different legs of a spider. Specifically, we will be determining whether more friction comes from a pushing or pulling motion of the leg.

```
spider <- read.csv("spider_wolff_gorb_2013.csv", skip=1)
```

## Initial visual inspection of the data

Each measurement comes from one of our legs while it is either pushing or pulling. So we have two variables. In RStudio we may view the data as

```
View(spider)
```

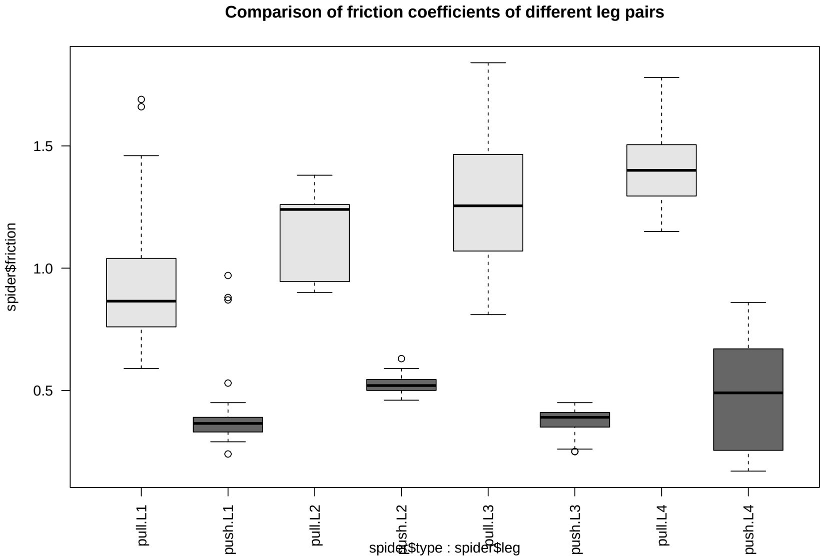
or just print it to console using `spider` or have a glimpse of first 6 rows

```
head(spider)
```

```
##   leg type friction
## 1  L1 pull     0.90
## 2  L1 pull     0.91
## 3  L1 pull     0.86
## 4  L1 pull     0.85
## 5  L1 pull     0.80
## 6  L1 pull     0.87
```

We can make a boxplot

```
boxplot(spider$friction ~ spider$type * spider$leg,
        col=c("grey90","grey40"), las=2,
        main="Comparison of friction coefficients of different leg pairs")
```



## A linear model with two variables

In order to model both the leg pair differences (L1, L2, L3, L4) and the push vs. pull difference, we need to include both terms in the R formula. Let's see what kind of design matrix will be formed with two variables in the formula:

```
X <- model.matrix(~ type + leg, data=spider)
```

```
View(X)
```

To estimate coefficients for this model, we use `lm` with the formula `~ type + leg`. We'll save the linear model to `fitTL` standing for a *fit* with *Type* and *Leg*.

```
fitTL <- lm(friction ~ type + leg, data=spider)
summary(fitTL)
```

```
##
## Call:
## lm(formula = friction ~ type + leg, data = spider)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -0.4639 -0.1344 -0.0053  0.1055  0.6951
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.0539     0.0282   37.43  < 2e-16 ***
## typepush     -0.7790     0.0248  -31.38  < 2e-16 ***
## legL2         0.1719     0.0457    3.76    2e-04 ***
## legL3         0.1605     0.0325    4.94  1.4e-06 ***
## legL4         0.2813     0.0344    8.18  1.0e-14 ***
```

```r
coefs <- coef(fitTL)
coefs
```

```
## (Intercept)    typepush       legL2       legL3       legL4
##     1.05392    -0.77901     0.17192     0.16049     0.28134
```

R uses the name `Coefficients` to denote the component containing the least squares **estimates**. It is important to remember that the coefficients are parameters that we do not observe, but only estimate.

## Mathematical representation

The model we are fitting above can be written as

$$Y_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \beta_3 x_{i,3} + \beta_4 x_{i,4} + \varepsilon_i, i = 1, \dots, n$$

or consider the indicator variable style

$$\texttt{friction} = \beta_0 + \beta_1 \mathbf{1}(\texttt{type=push}) + \beta_2 \mathbf{1}(\texttt{leg=L2}) + \beta_3 \mathbf{1}(\texttt{leg=L3}) + \beta_4 \mathbf{1}(\texttt{leg=L4}) + \varepsilon_i, \ i = 1, \dots, n$$

We can now form the matrix **X** depicted above and obtain the least square estimates with:

$$\hat{\beta} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{Y}$$

```
Y <- spider$friction
X <- model.matrix(~ type + leg, data=spider)
beta.hat <- solve(t(X) %*% X) %*% t(X) %*% Y
t(beta.hat)
```

```
##      (Intercept) typepush   legL2   legL3   legL4
## [1,]      1.0539 -0.77901 0.17192 0.16049 0.28134
```

We can see that these values agree with the output of `lm`.

## Examining the estimated coefficients

We can make the same plot as before, with arrows for each of the estimated coefficients in the model (code not shown).

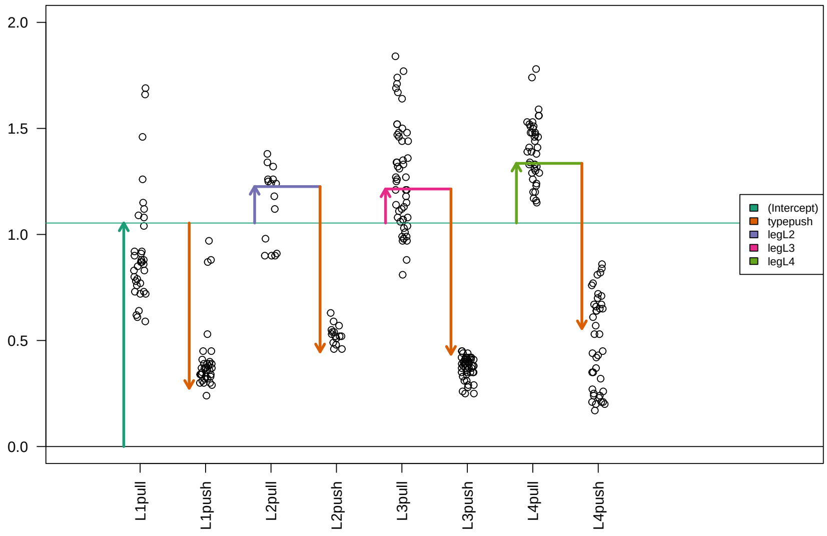


Figure 4: Diagram of the estimated coefficients in the linear model.

## Contrasting coefficients

Sometimes, the comparison we are interested in is represented directly by a single coefficient in the model, such as the push vs. pull difference, which was `coefs[2]` above. However, sometimes, we want to make a comparison which is not a single coefficient, but a combination of coefficients, which is called a *contrast*. To introduce the concept of *contrasts*, first consider the comparisons which we can read off from the linear model summary:

```
coef(fitTL)
```

```
## (Intercept)    typepush       legL2       legL3       legL4
##     1.05392    -0.77901     0.17192     0.16049     0.28134
```

Here we have the intercept estimate, the push vs. pull estimated effect across all leg pairs, and the estimates for the L2 vs. L1 effect, the L3 vs. L1 effect, and the L4 vs. L1 effect. What if we want to compare two groups and one of those groups is not L1? The solution to this question is to use *contrasts*.

A *contrast* is a combination of estimated coefficient: $\mathbf{c}^\top \hat{\beta}$, where $\mathbf{c}$ is a column vector with as many rows as the number of coefficients in the linear model. If $\mathbf{c}$ has a 0 for one or more of its rows, then the corresponding estimated coefficients in $\hat{\beta}$ are not involved in the contrast.

If we want to compare leg pairs L3 and L2, this is equivalent to contrasting two coefficients from the linear model because, in this contrast, the comparison to the reference level *L1* cancels out:

$$(\mathsf{L3} - \mathsf{L1}) - (\mathsf{L2} - \mathsf{L1}) = \mathsf{L3} - \mathsf{L2}$$

An easy way to make these contrasts of two groups is to use the `contrast` function from the *contrast* package. We just need to specify which groups we want to compare. We have to pick one of *pull* or *push* types, although the answer will not differ, as we will see below.

```
library(contrast) #Available from CRAN
L3vsL2 <- contrast(fitTL,list(leg="L3",type="pull"),list(leg="L2",type="pull"))
L3vsL2
```

```
## lm model parameter contrast
##
##   Contrast     S.E.     Lower    Upper     t  df Pr(>|t|)
##  -0.011429 0.043197 -0.096465 0.073606 -0.26 277   0.7915
```

The first column `Contrast` gives the L3 vs. L2 estimate from the model we fit above.

We can show that the least squares estimates of a linear combination of coefficients is the same linear combination of the estimates. Therefore, the effect size estimate is just the difference between two estimated coefficients. The contrast vector used by `contrast` is stored as a variable called `X` within the resulting object (not to be confused with our original **X**, the design matrix).

```
coefs[4] - coefs[3]
```

```
##     legL3
## -0.011429
```

```
(cT <- L3vsL2$X)
```

```
##   (Intercept) typepush legL2 legL3 legL4
## 1           0        0    -1     1     0
## attr(,"assign")
## [1] 0 1 2 2 2
## attr(,"contrasts")
## attr(,"contrasts")$type
## [1] "contr.treatment"
##
## attr(,"contrasts")$leg
## [1] "contr.treatment"
```

```
cT %*% coefs
```

```
##          [,1]
## 1 -0.011429
```

## Remarks

There will be an update on exercises:

- Topic of confounding (the very last) will be removed. I try to figure a task about contrasts.