Imagine that one has a data matrix $X \in \mathbb{R}^{n \times p}$ consisting of n observations, each with p features, as well as a response vector $y \in \mathbb{R}^n$. We want to build a model for y using the feature columns in X. In ***ordinary least squares (OLS)***, one seeks a vector of coefficients $\hat{\beta} \in \mathbb{R}^p$ such that

$$\hat{\beta} = \underset{\beta \in \mathbb{R}^p}{\text{argmin}} \quad \|y - X\beta\|_2^2.$$

In ***non-negative least squares (NNLS)***, we seek a vector coefficients $\hat{\beta} \in \mathbb{R}^p$ such that it minimizes $\|y - X\beta\|_2^2$ subject to the additional requirement that each element of $\hat{\beta}$ is non-negative.

There are a number of ways to perform NNLS in R. The first two methods come from Reference 1, while I came up with the third. (I'm not sharing the third way Reference 1 details because it claims that the method is buggy.)

Let's generate some fake data that we will use for the rest of the post:

```
set.seed(1)
n <- 100; p <- 10
x <- matrix(rnorm(n * p), nrow = n)
y <- x %*% matrix(rep(c(1, -1), length.out = p), ncol = 1) + rnorm(n)
```

Method 1: the `nnls` package

```
library(nnls)
mod1 <- nnls(x, y)
mod1$x
# [1] 0.9073423 0.0000000 1.2971069 0.0000000 0.9708051
# [6] 0.0000000 1.2002310 0.0000000 0.3947028 0.0000000
```

Method 2: the `glmnet` package

The `glmnet()` function solves the minimization problem

$$\hat{\beta} = \underset{\beta \in \mathbb{R}^p}{\text{argmin}} \quad \frac{1}{2n}\|y - X\beta\|_2^2 + \lambda \left[\frac{1-\alpha}{2}\|\beta\|_2^2 + \alpha\|\beta\|_1 \right],$$

where α and λ are hyperparameters the user chooses. By setting $\alpha = 1$ (the default) and $\lambda = 0$, `glmnet()` ends up solving the OLS problem. By setting `lower.limits = 0`, this forces the coefficients to be non-negative. We should also set `intercept = FALSE` so that we don't have an extraneous intercept term.

```
library(glmnet)
mod2 <- glmnet(x, y, lambda = 0, lower.limits = 0, intercept = FALSE)
coef(mod2)
# 11 x 1 sparse Matrix of class "dgCMatrix"
# s0
# (Intercept) .
# V1          0.9073427
# V2          .
# V3          1.2971070
# V4          .
# V5          0.9708049
# V6          .
# V7          1.2002310
# V8          .
# V9          0.3947028
# V10         .
```

Method 3: the `bvls` package

NNLS is a special case of ***bounded-variable least squares (BVLS)***, where instead of having constraints $\beta_j \geq 0$ for each $j = 1, \dots, p$, one has constraints $a_j \leq \beta_j \leq b_j$ for each j. BVLS is implemented in the `bvls()` function of the `bvls` package:

```
library(bvls)
mod3 <- bvls(x, y, bl = rep(0, p), bu = rep(Inf, p))
mod3$x
# [1] 0.9073423 0.0000000 1.2971069 0.0000000 0.9708051
# [6] 0.0000000 1.2002310 0.0000000 0.3947028 0.0000000
```

In the above, `bl` contains the lower limits for the coefficients while `bu` contains the upper limits for the coefficients.

References:

1. Things I Thought At One Point. Three ways to do non-negative least squares in R.