

# Estimate Many Similar Models and Display Coefficients

*Ryan T. Moore*

*14 October 2016*

## 1 The Problem

Suppose you want to estimate a set of similar models, and display the coefficients in a plot. For example, you want to regress a measure of social trust ( $Y$ ) on three different personality measures ( $X_1$ ,  $X_2$ , and  $X_3$ ) one at a time. Or, you are interested in the relationship between media heterogeneity and legal freedoms of the press, conditional on levels of democracy, but you have three different ways to measure legal freedoms.

## 2 A Solution

Copy the function below (or from this [Gist](#)) and run the function `similar_models()`, such as

```
similar_models(df, "y", c("x1", "x2", "x3"))`
```

### 2.1 Describing the Models

We want to perform a set of univariate regressions, estimating these three similar models:

$$Y_i = \beta_0 + \beta_1 x_1 + \epsilon_i \quad (1)$$

$$Y_i = \beta_0 + \beta_2 x_2 + \epsilon_i \quad (2)$$

$$Y_i = \beta_0 + \beta_3 x_3 + \epsilon_i \quad (3)$$

Or, we want to estimate similar models, but always include  $w$ :

$$Y_i = \beta_0 + \beta_1 x_1 + \gamma w_i + \epsilon_i \quad (4)$$

$$Y_i = \beta_0 + \beta_2 x_2 + \gamma w_i + \epsilon_i \quad (5)$$

$$Y_i = \beta_0 + \beta_3 x_3 + \gamma w_i + \epsilon_i \quad (6)$$

Then, we want a plot of  $\beta_1$ ,  $\beta_2$ ,  $\beta_3$  and their associated confidence intervals.

First, we load some required libraries:

```
library(ggplot2)
library(tidyr)
library(dplyr)
```

### 2.2 The Function

The function `similar_models` takes the data, the outcome, the set of related variables that you want to include, one-by-one, and any variables that you want to include in every model.

```

similar_models <- function(data, outcome, related_variables, always_include = NULL,
                           verbose = FALSE){

  ## Initialize storage:
  n_models <- length(related_variables)
  storage <- matrix(NA, n_models, 3)

  ## Loop over names
  for(model.idx in 1:n_models){
    if(verbose == TRUE){
      cat(paste("Starting ", related_variables[model.idx], "\n"))
    }
    ## Create string for formula:
    this_formula <- paste0(outcome, " ~ ", related_variables[model.idx])

    if(length(always_include) > 0){
      additional_terms <- ""
      for(always.idx in 1:length(always_include)){
        additional_terms <- paste(additional_terms, "+", always_include[always.idx])
      }

      this_formula <- paste(this_formula, additional_terms)
    }

    this_formula <- as.formula(this_formula)

    ## Estimate:
    lm_out <- lm(this_formula, data = data)
    ## Store coef and CI:
    results <- c(coef(lm_out)[2], confint(lm_out)[2, 1], confint(lm_out)[2, 2])
    storage[model.idx, ] <- results
  }

  storage <- as.data.frame(storage)
  ## Add variable names:
  storage$Variable <- related_variables
  names(storage)[1:3] <- c("Coefficient", "Lower", "Upper")

  ## Sort:
  storage$Variable <- factor(storage$Variable,
                            levels = storage$Variable[order(storage$Coefficient)])

  return(storage)
}

```

## 2.3 Sample Data

Create some data with variables  $y$ ,  $x_1$ ,  $x_2$ ,  $x_3$ , and  $w$ :

```

set.seed(418)
n <- 100
df <- data.frame(x1 = rnorm(n), x2 = rnorm(n), x3 = rnorm(n), w = rnorm(n))
df <- df %>% mutate(y = x1 + 2*x2 + 3*x3 + 2*w + rnorm(n))

```

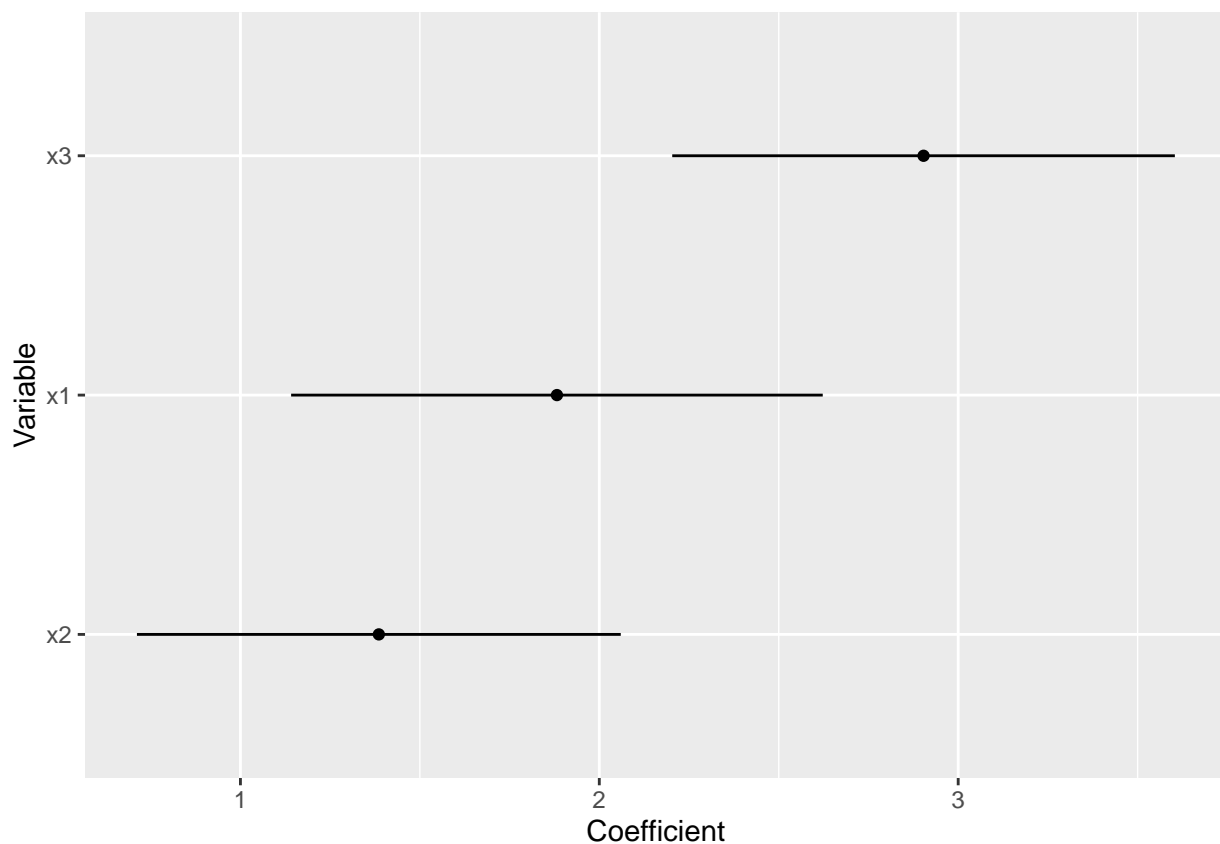
## 2.4 Estimate the Similar Models

Store the variable names as a vector of strings, then estimate the similar models in Equations 1 to 3.

```
related_vars <- c("x1", "x2", "x3")

lm_results <- similar_models(df, "y", related_vars)

## Plot the results:
lm_results %>% ggplot(aes(x = Coefficient, y = Variable)) + geom_point() +
  geom_segment(aes(x = Lower, xend = Upper, y = Variable, yend = Variable))
```



## 2.5 Including other Predictors

Specify the variable(s) you want to always include, such as  $w$ . You could include `c("w", "z", "q", \dots)`.

This estimates models 4 to 6.

```
lm_results_w <- similar_models(df, "y", related_vars, always_include = "w")

## Plot:
lm_results_w %>% ggplot(aes(x = Coefficient, y = Variable)) + geom_point() +
  geom_segment(aes(x = Lower, xend = Upper, y = Variable, yend = Variable))
```

