Visualizing Regression Results with dotwhisker

Yeiyoung Choo

Spring 2023

In this module, we will learn how to create dot-and-whisker plots, which is one of the common ways to visualize results from regression analysis. As we will see below, the plots show the point estimates of regression coefficients as 'dots' and the confidence intervals of these estimates as 'whiskers'.

We will use *dotwhisker* package introduced by Frederick Solt and Yue Hu in September 2021. The package provides a quick and easy way to generate the coefficient plots we want; and pairs well with dplyr as it builds on ggplot2 architecture. With dotwhisker, we can either use results directly from regression objects or tidied results (`broom::tidy()`) stored in dataframes.

Load Packages

```
library(tidyverse)
library(dotwhisker)
library(broom)
```

Load Data

Let's load the data we need for regression analysis. The data come from a study by Appel & Loyle (2012) on post-conflict justice institutions (truth commissions in particular) and foreign direct investment. We will use the results from our regression models to generate dot-and-whisker plots.

```
al2012 <- readRDS("al2012.rds")  # data abridged and prepped for exercise
```

Approach 1: Using Results from Regression Objects as Input

Recall that the basic syntax for linear regression is `lm(y ~ x, data = data)`. We use `+` to add predictors. For example: `lm(y ~ x1 + x2 + x3, data = data)`. Each model specifies what variables we use to predict the outcome variable. We will store each model as a regression object. For example: `m1 <- lm(y ~ x, data = data)`. Because results stored within the objects, they are readily available when generating dot-and-whisker plots.

Create a regression object

Here we will run our first regression model and store it as an object. Let's call it m1. The outcome variable is `fdiflow`, capturing inflows of foreign direct investment. The variable `truthvictim` is our key predictor, which is implementation of a post-conflict justice institution. Check regression results using `summary()`.

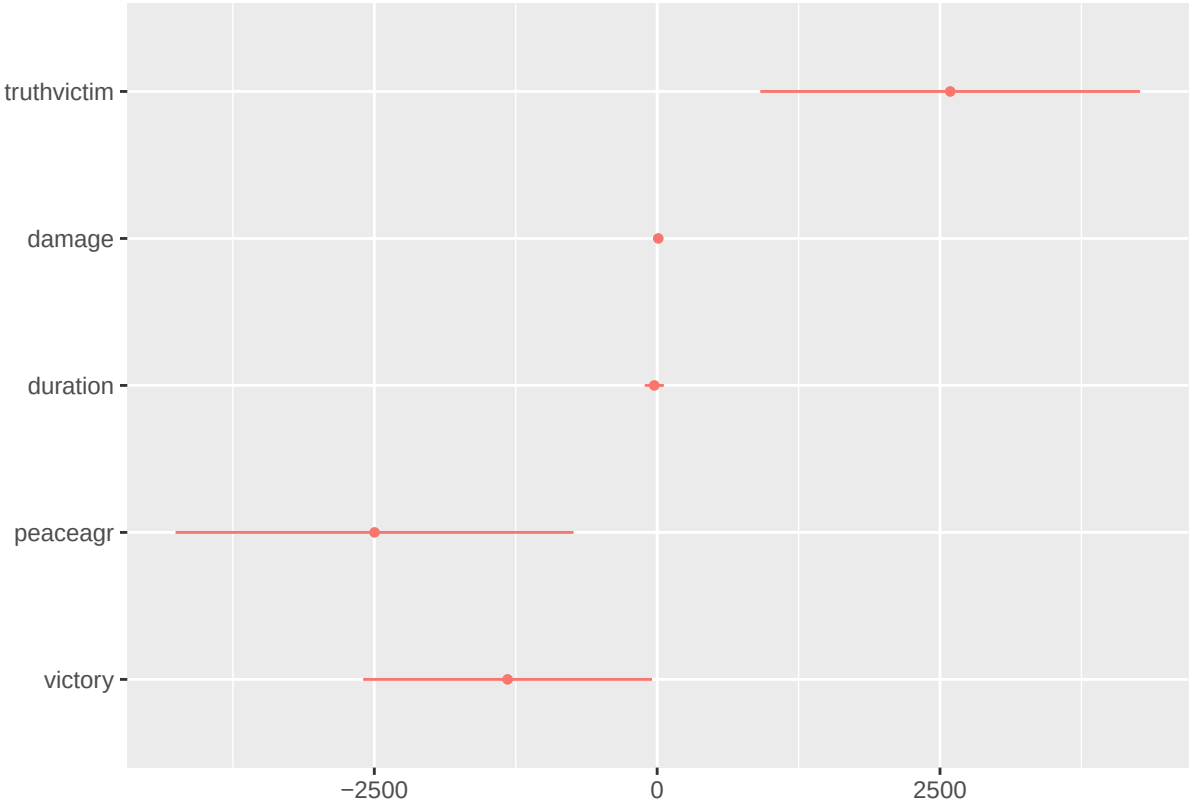
```
# DV: foreign direct investment inflows
m1 <- lm(fdiflow ~ truthvictim + damage + duration +
    peaceagr + victory, data = al2012)
summary(m1)  #see regression results
```

```
##
## Call:
## lm(formula = fdiflow ~ truthvictim + damage + duration + peaceagr +
##     victory, data = al2012)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3907.0 -1163.3  -166.3   270.7 20834.2
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1418.32     510.04   2.781  0.00662 **
## truthvictim  2590.53     844.96   3.066  0.00287 **
## damage         10.01      10.59   0.946  0.34685
## duration      -25.54      42.52  -0.601  0.54959
## peaceagr    -2497.99     885.38  -2.821  0.00590 **
## victory     -1321.90     642.40  -2.058  0.04254 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2753 on 89 degrees of freedom
## Multiple R-squared:  0.1529, Adjusted R-squared:  0.1053
## F-statistic: 3.213 on 5 and 89 DF,  p-value: 0.01031
```

Generate a dot-and-whisker plot

We now use `dwplot()` from *dotwhisker* package to generate a dot-and-whisker plot, specifying the regression object in the parenthesis. The plot visualizes the regression results you saw above with `summary()`. Specifically, the coefficient estimates on predictor variables are shown as dots. The horizontal line across the dot is the whisker, which shows the 95% confidence interval of the coefficient estimate.

```
dwplot(m1)  #intercept excluded by default
```



Note: When you use dwplot() directly with regression objects, the plot will exclude the intercept by default. Because we use the dot-and-whisker plot to visualize coefficient estimates on predictor variables of interest, we don't need to show the intercept in our plot. If you want to include the intercept, you can set the `show_intercept` argument as follows: `dwplot(m1,show_intercept=TRUE)`.

Notice that coefficient estimates on `damage` and `duration` are hard to interpret from this plot, because of the relative size of the estimates. The regression results from `summrary()` show that the coefficient estimates are 10.01 and -25.54 for them respectively, while others are at thousands, such as 2590.54 for `truthvictim` and -2497.99 for `peaceagr`. While this concern is beyond the scope of today's lesson, it is good to think about how we can scale these variables in the first place to make coefficient estimates more comparable.

For example, Appel and Loyle (2012) additionally report the percentage change in FDI inflows with changes in predictor variables. Holding others constant, what's the percentage change in FDI inflows when we increase conflict damage from the 25th to 75th percentile level? Or when we switch from not having post-conflict institutions to having them in place? The percentage change estimates are 85 and 358, which are more comparable than 10.01 and 2590.94 from above.

Create another regression object

Let's run our second regression model. Again, store it as an object, calling it m2. The second model still has the key predictor variable `truthvictim`, but uses political variables (political constraints and the polity score) instead of conflict variables as controls.

```
# DV: foreign direct investment inflows
m2 <- lm(fdiflow ~ truthvictim + polcon + polity2,
    data = al2012)
summary(m2)  #check results
```

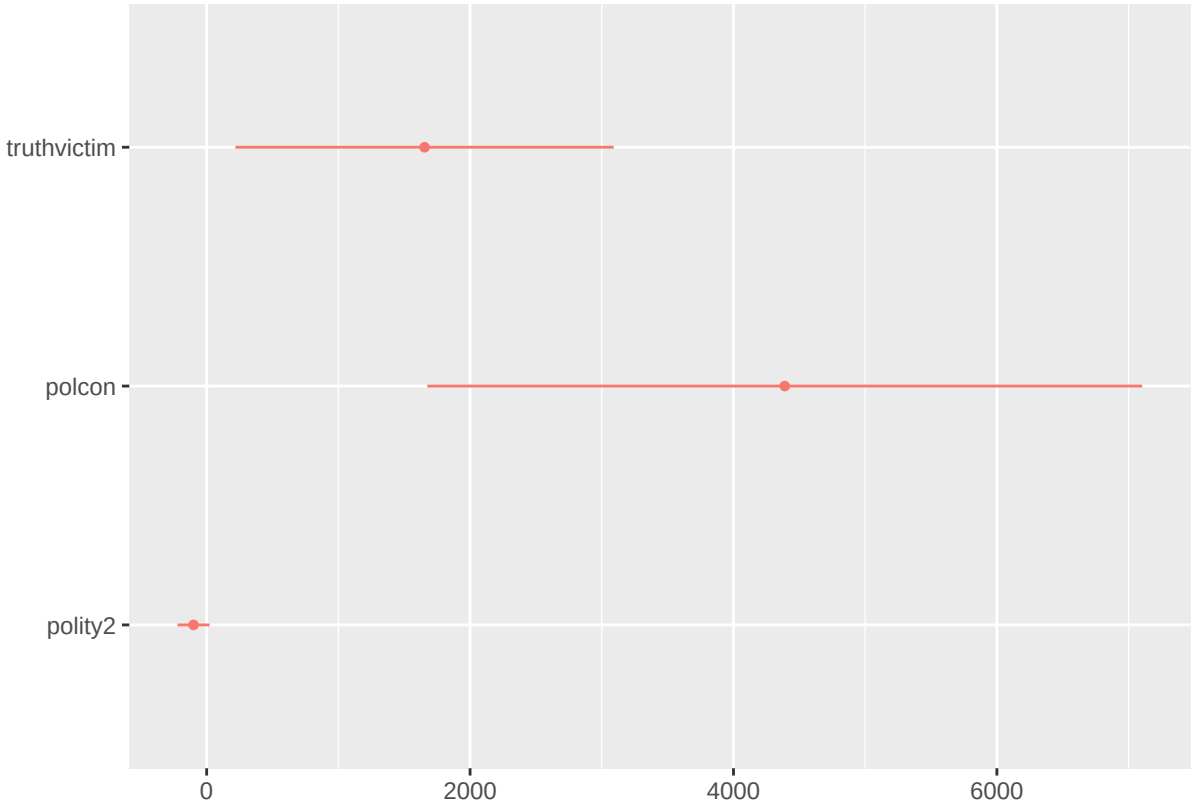
```
##
## Call:
## lm(formula = fdiflow ~ truthvictim + polcon + polity2, data = al2012)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3578.5 -1015.7  -144.8   209.3 21085.1
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -632.59     459.17  -1.378  0.17168
## truthvictim  1654.68     722.89   2.289  0.02439 *
## polcon       4389.13    1366.29   3.212  0.00182 **
## polity2       -99.64      61.24  -1.627  0.10718
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2719 on 91 degrees of freedom
## Multiple R-squared:  0.1555, Adjusted R-squared:  0.1277
## F-statistic: 5.586 on 3 and 91 DF,  p-value: 0.00146
```

Generate a dot-and-whisker plot

We use the object `m2` to generate a dot-and-whisker plot. The plot shows coefficient estimates for the predictor variables used in our Model 2.The coefficient estimate for `truthvictim` is different from what we had with Model 1, but it is hard to draw the comparison when we generate plots separately.

```
dwplot(m2)  #intercept excluded by default
```



Note: `dwplot()` shows 95% confidence interval by default. We rarely need to adjust this setting, but you can specify if you want to show 90% confidence interval for example: `dwplot(m2, ci=.90)`.

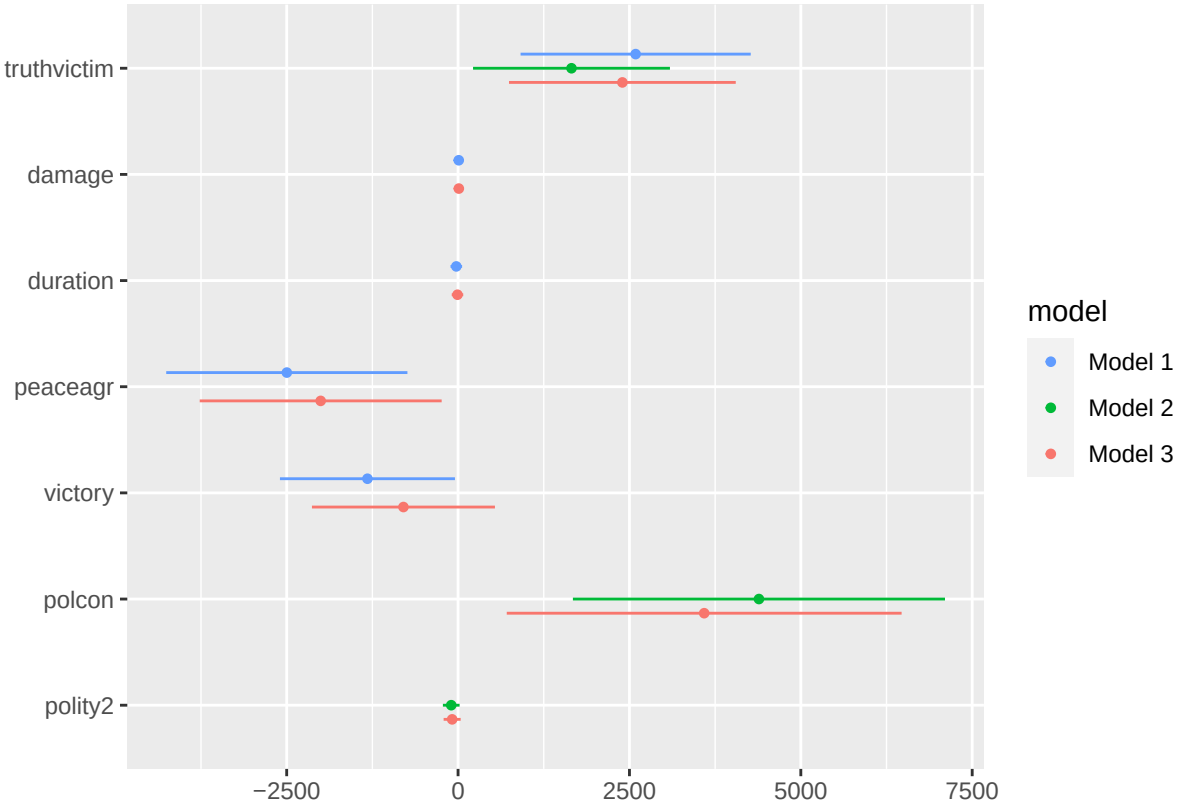
Plot results from multiple regression models

Here we learn how to visualize results from multiple regression models in one plot. This allows us to easily compare coefficient estimates across different models. First, we run regression models and save as objects.

```
m1 <- lm(fdiflow ~ truthvictim + damage + duration +
    peaceagr + victory, data = al2012)  #conflict variables
m2 <- lm(fdiflow ~ truthvictim + polcon + polity2,
    data = al2012)  #political variables
m3 <- lm(fdiflow ~ truthvictim + damage + duration +
    peaceagr + victory + polcon + polity2, data = al2012)  #all
```

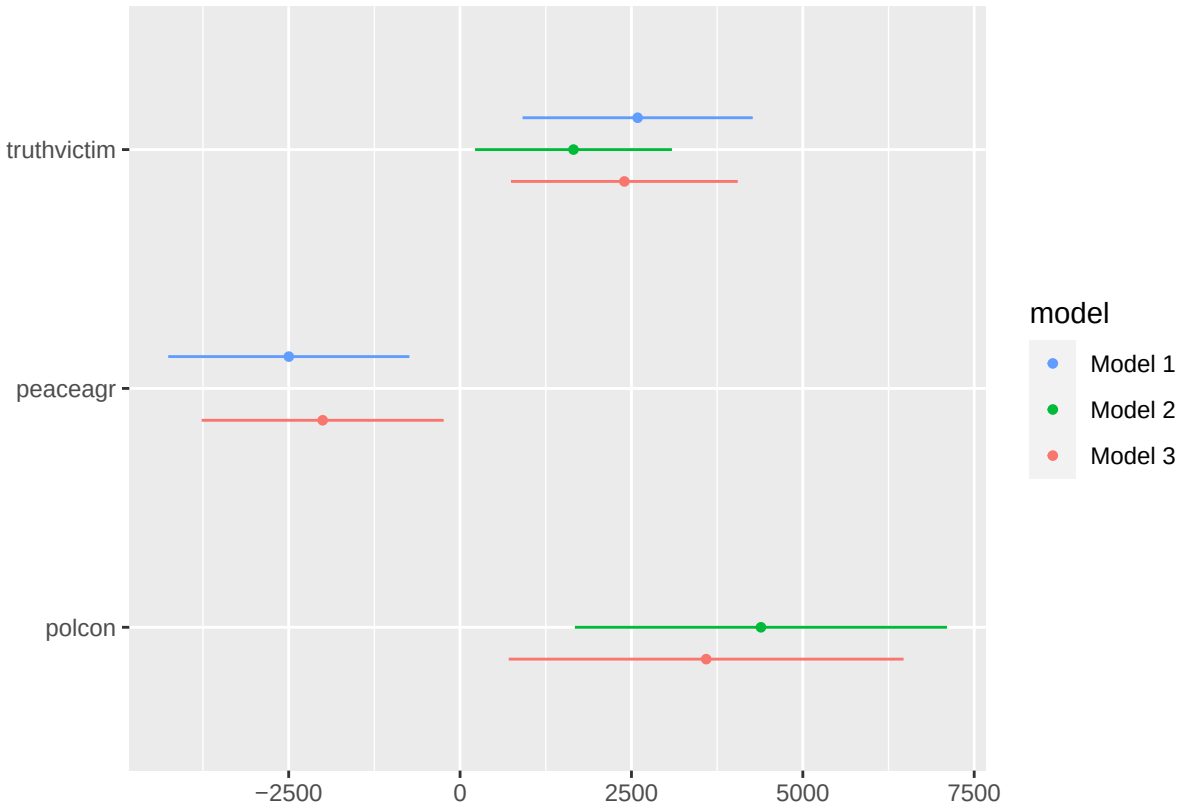
Next, we use `list()` within `dwplot()` to create a single dot-and-whisker plot that shows regression results from all of our models. Similar to a regression table that displays results from multiple models in different columns, this plot shows coefficient estimates from all models as well as how some of the estimates vary across them. Instead of numbers, we see dots-and-whisker. For example, we can easily see how the coefficient estimate on `truthvictim` varies across three models.

```
dwplot(list(m1, m2, m3))
```



Note that the plot gets crowded as we add more predictors. For effective visualization, it is a good idea to focus on key variables. Often times, a coefficient plot like this complements a regression table, so the reader can always find full results in the table. Let's focus on `truthvictim`, which captures post-conflict justice institutions, and `peaceagr` and `polcon`, the conflict and the political variables our analysis found to be statistically significant. (Of course, what is considered important depends on the study and the researcher's point of view.)

```
dwplot(list(m1, m2, m3), vars_order = c("truthvictim",
    "peaceagr", "polcon"))
```

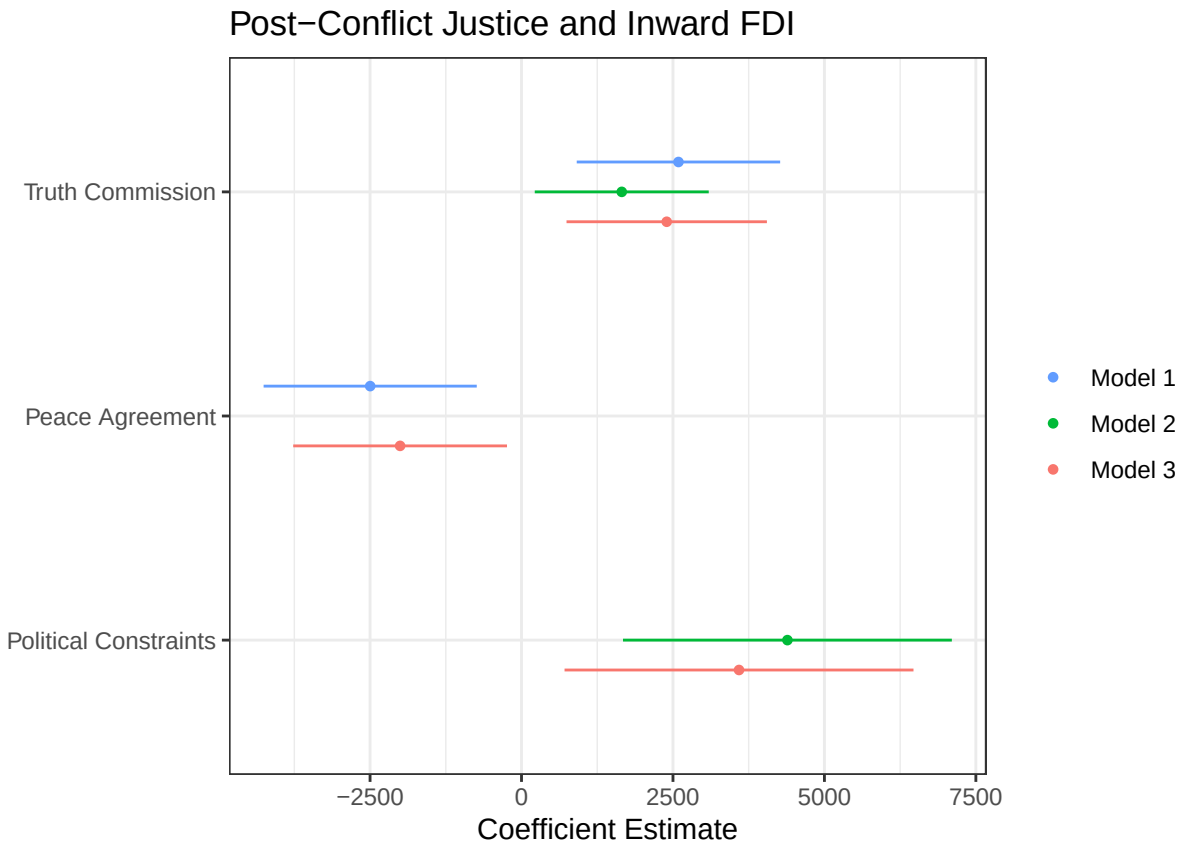


Customize the plot

As with ggplot, we can customize the dot-and-whisker plot to enhance information and visuals. At minimum, we want to select key variables, label our models and variables properly, and add information that makes it clear what this plot shows.

```
plot1 <-
  dwplot(list(m1,m2,m3),
         model_order=c("Model 1","Model 2","Model 3"),
         vars_order=c("truthvictim","peaceagr","polcon")) %>% #select, order variables
  relabel_predictors(                    #relabel to spell out variable names
    c(truthvictim="Truth Commission",
      peaceagr="Peace Agreement",
      polcon="Political Constraints")) +
  theme_bw()+                            #optional - cleaner background
  xlab("Coefficient Estimate") +         #label x-axis
  ylab("") +
  ggtitle("Post-Conflict Justice and Inward FDI") +
  theme(legend.title=element_blank())

plot1
```




```
#Save plot as a PNG file
ggsave("dwplot1.png", plot1, width=6.5, height=4.5, dpi=400)
 # ggsave("your filepath/your filename.png"...)
```

Approach 2: Using Tidied Results as Input

The `dwplot()` can also take inputs from regression results stored in tidy dataframes. We use the `tidy()` function in `broom` package to tidy results and store them in dataframes.

Tidy results from regression models

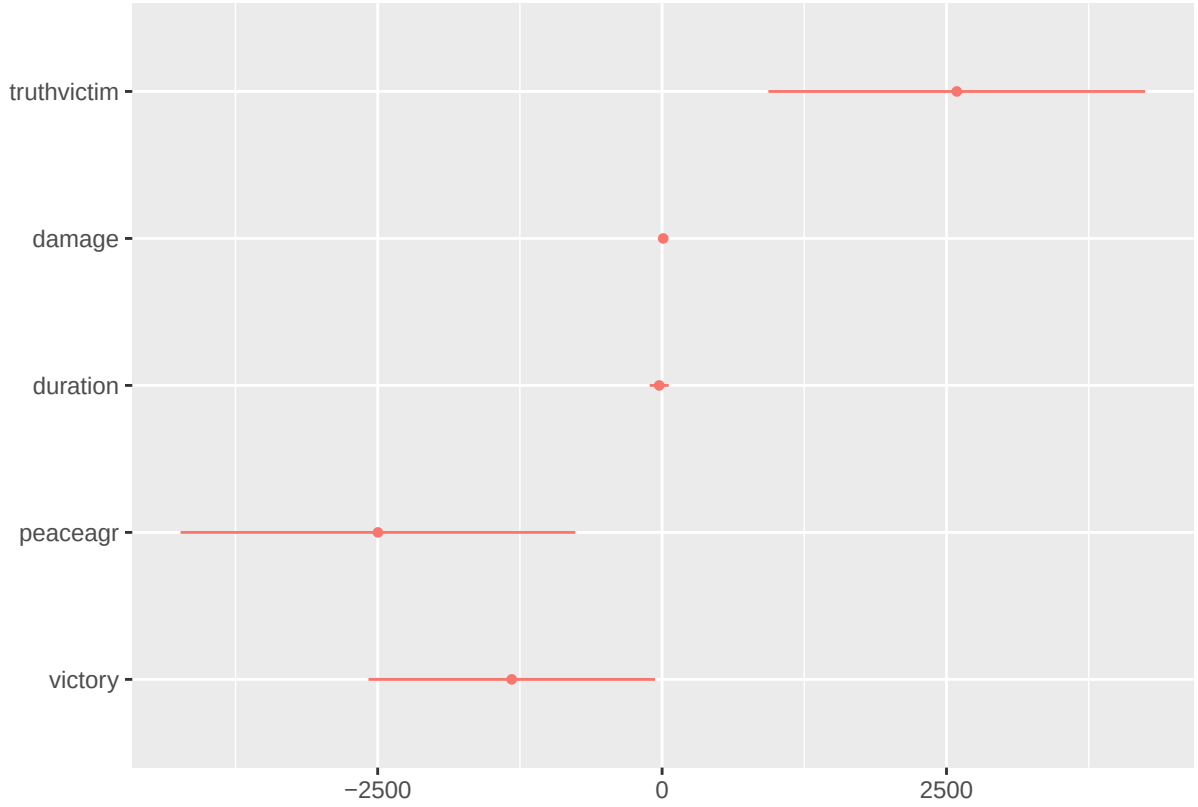
We use `broom::tidy()` to tidy regression results. Let's call our tidy dataframes `m1df`, `m2df`, and `m3df` to distinguish them from the regression objects.

```
m1df <- tidy(m1)  #create a tidy dataframe from Model 1
m2df <- tidy(m2)
m3df <- tidy(m3)
```

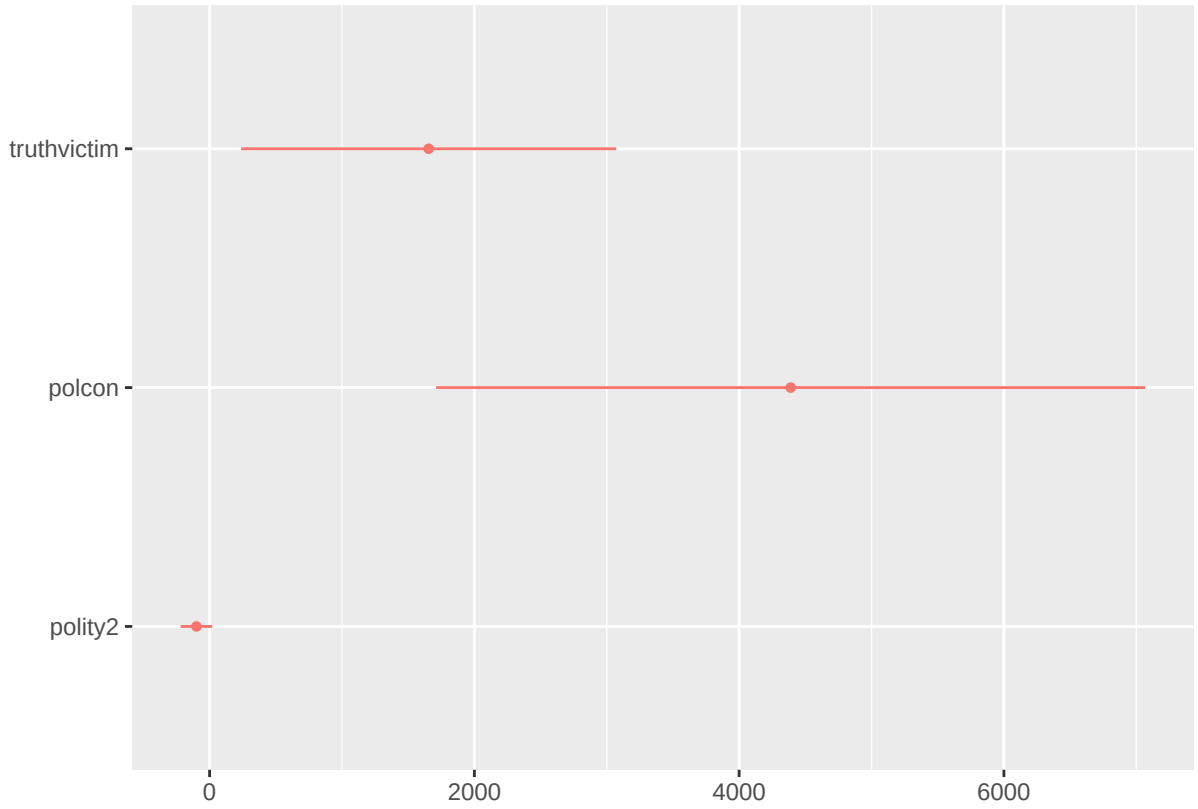
Generate dot-and-whisker plots from tidy dataframes

Instead of regression objects, we now use tidy dataframes as input for `dwplot()`. We see that `dwplot(m1df)` generates the same plot as `dwplot(m1)` except that it does not omit the intercept by default.

```
dwplot(m1df)  #same as dwplot(m1)
```


truthvictim
damage
duration
peaceagr
victory
−2500
0
2500

```
dwplot(m2df)
```


truthvictim
polcon
polity2
0
2000
4000
6000

The `tidy()` function is useful when storing regression results for select variables. We can select variables later when using `dwplot()`, but doing so here allows us to have concise dataframes as input. We use the `filter()` function to choose or drop variables. (We could not do this with regression objects above, since these are not dataframes.)

```
# for example
m1df <- tidy(m1) %>%
    filter(term != "(Intercept)")  #drop intercept
```

Plot results from multiple regression models

Let's replicate the final dot-and-whisker plot we had above by modifying our tidy dataframes first. We also add a column called `model` to specify each model. This will identify our regression models when we combine multiple tidy dataframes into one and use it as input for `dwplot()`.

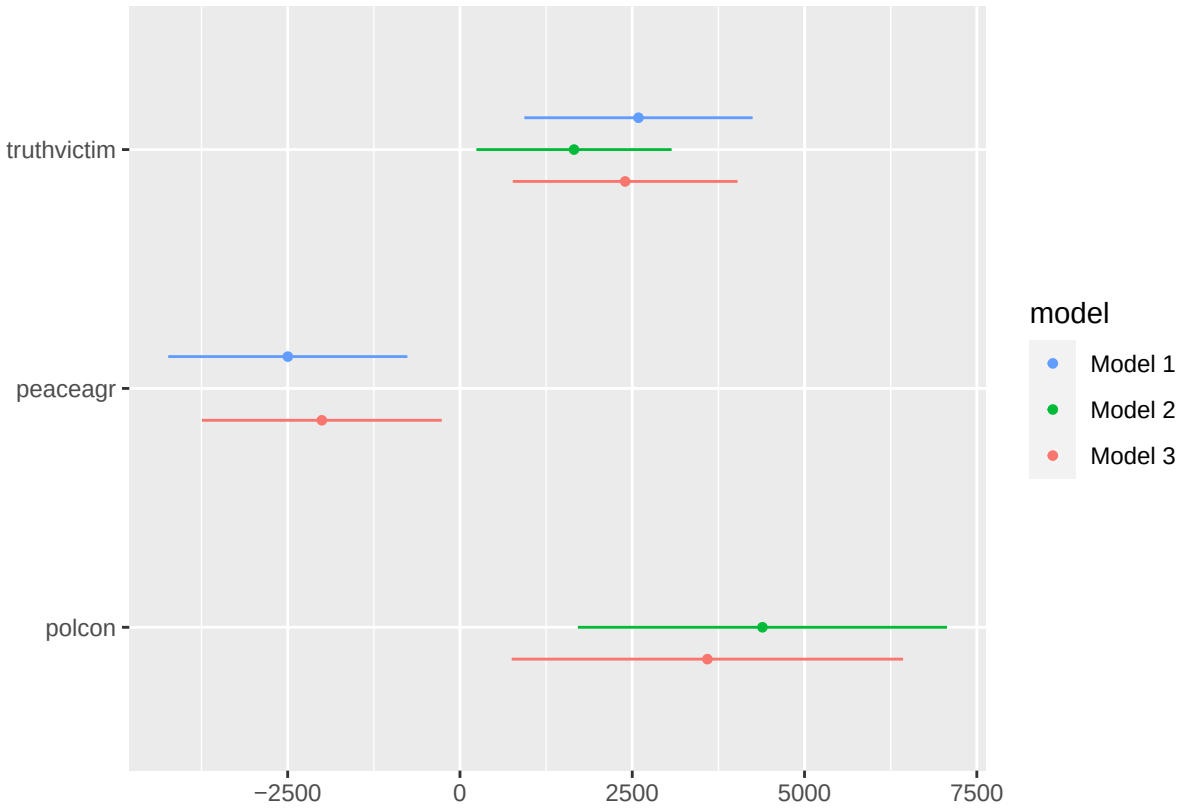
```
m1df <- tidy(m1) %>%
    filter(term == "truthvictim" | term == "peaceagr") %>%
    mutate(model = "Model 1")
m2df <- tidy(m2) %>%
    filter(term == "truthvictim" | term == "polcon") %>%
    mutate(model = "Model 2")
m3df <- tidy(m3) %>%
    filter(term == "truthvictim" | term == "peaceagr" |
        term == "polcon") %>%
    mutate(model = "Model 3")
```

Use the `rbind()` function to merge tidy dataframes together into one dataframe.

```
models <- rbind(m1df, m2df, m3df)  #merge dataframes
```

Generate a dot-and-whisker plot for all regression models, displaying key variables of interest. Remember we have selected variables already.

```
dwplot(models)
```

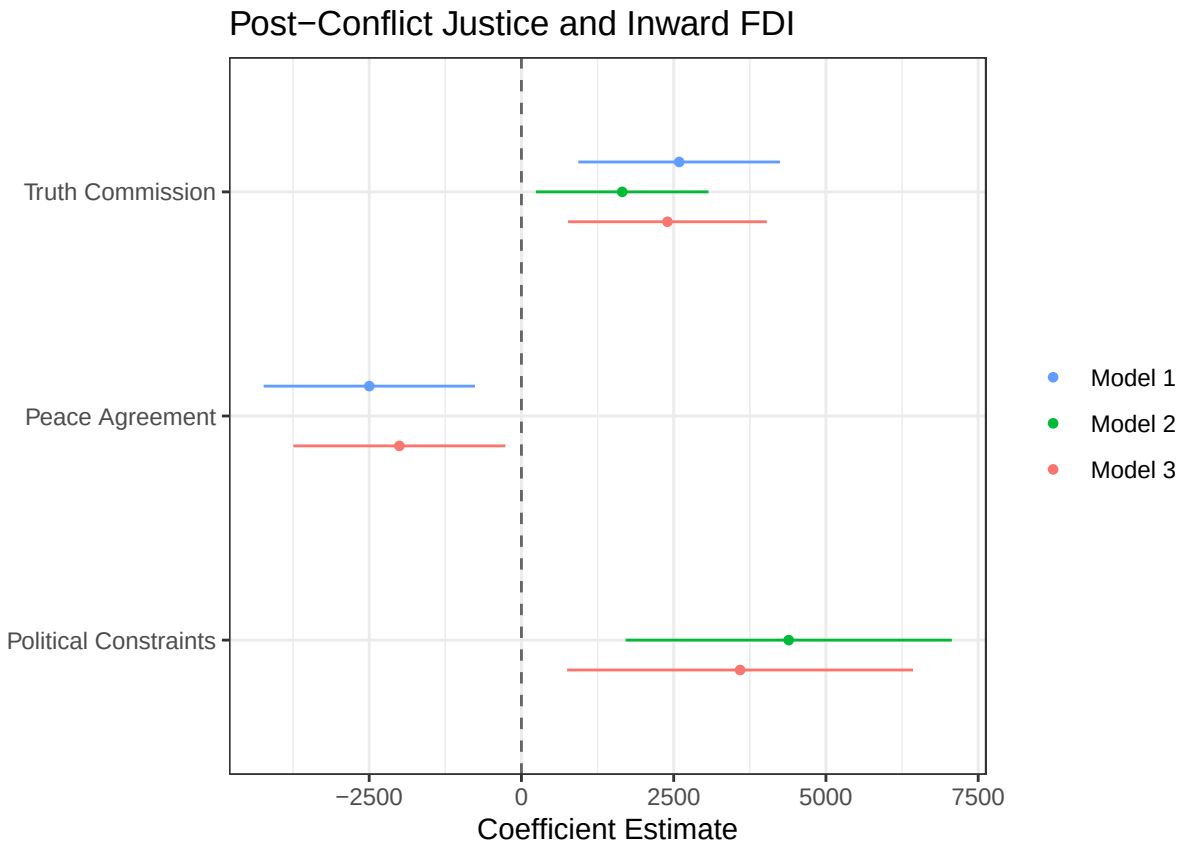


In-Class Exercise: Customizing a dot-and-whisker plot

Let's customize the plot we just generated with our final tidy dataframe. Try to replicate the plot you saved above (`dwplot1.png`), first starting with `plot 2 <-` to store the plot as an object. Save the plot as `dwplot2.png`. (Bonus: Add a dotted vertical line where x-intercept is 0.)

```
plot2 <- dwplot(models,
                model_order=c("Model 1","Model 2","Model 3"),
                vars_order=c("truthvictim","peaceagr","polcon")) %>%
  relabel_predictors(
    c(truthvictim="Truth Commission",
      peaceagr="Peace Agreement",
      polcon="Political Constraints")) +
  theme_bw()+
  xlab("Coefficient Estimate") +
  ylab("") +
  geom_vline(xintercept = 0,     #add a vertical line across 0
             colour = "grey40",
             linetype = 2) +
  ggtitle("Post-Conflict Justice and Inward FDI")+
  theme(legend.title=element_blank())

plot2
```




```
ggsave("dwplot2.png", plot2, width=6.5, height=4.5, dpi=400)
```

Adding a vertical line across zero is useful, especially when we have coefficient estimates of opposite signs (i.e. positive and negative). The plot clearly shows our finding that implementing a post-conflict justice institution is associated with higher levels of FDI inflows, or that the effect of implementing a post-conflict justice institution is positive on attracting inward FDI.