The Problem

You have two predictors in your model. One seems to have a stronger coefficient than the other. But is it significant?

Example: when predicting a worker's salary, is the standardized coefficient of *number of extra hours* (`xtra_hours`) really larger than that of *number of compliments given the to boss* `n_comps`?

```
library(parameters)
library(effectsize)

data("hardlyworking", package = "effectsize")

hardlyworkingZ <- standardize(hardlyworking)

m <- lm(salary ~ xtra_hours + n_comps, data = hardlyworkingZ)

model_parameters(m)
#> Parameter   | Coefficient |   SE |        95% CI |    t(497) |
p
#> --------------------------------------------------------------
---------
#> (Intercept) |   -7.19e-17 | 0.02 | [-0.03, 0.03] | -4.14e-15 | >
.999
#> xtra_hours  |        0.81 | 0.02 | [ 0.78, 0.84] |     46.60 | <
.001
#> n_comps     |        0.41 | 0.02 | [ 0.37, 0.44] |     23.51 | <
.001
```

Here are 4 methods to test coefficient equality in `R`.

Notes
- If we were interested in the unstandardized coefficient, we would not need to first standardize the data.
- Note that if one parameter was positive, and the other was negative, one of the terms would need to be first reversed (-X) to make this work.

Method 1: As Model Comparisons

Based on this awesome tweet.

Since:

\(\hat{Y} = a \times \text{xtra_hours} + a \times \text{n_comps} \\ = a \times (\text{xtra_hours} + \text{n_comps})\)

We can essentially force a constraint on the coefficient to be equal by using a composite variable.

```
m0 <- lm(salary ~ I(xtra_hours + n_comps), data = hardlyworkingZ)
```

```
model_parameters(m0)
#> Parameter                | Coefficient |   SE |         95% CI |
t(498) |      p
#> ------------------------------------------------------------
------------------
#> (Intercept)              |   -2.05e-17 | 0.02 | [-0.04, 0.04] |
-9.57e-16 | > .999
#> xtra_hours + n_comps |         0.61 | 0.01 | [ 0.58, 0.64] |
41.09 | < .001
```

We can then compare how this model compares to our first model without this constraint:

```
anova(m0, m)
#> Analysis of Variance Table
#>
#> Model 1: salary ~ I(xtra_hours + n_comps)
#> Model 2: salary ~ xtra_hours + n_comps
#>   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
#> 1    498 113.67
#> 2    497  74.95  1    38.716 256.73 < 2.2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can conclude that the unconstrained model is significantly better than the constrained model - meaning that \(\beta_{\text{xtra_hours}} > \beta_{\text{n_comps}}\).

Method 2: Paternoster et al (1998)

According to Paternoster et al. (1998), we can compute a *t*-test to compare the coefficients:

```
bs <- coef(m)[-1]
V <- vcov(m)[-1, -1]

tibble::tibble(
  diff_estim = diff(bs),
  diff_SE = sqrt(V[1, 1] + V[2, 2] - 2 * V[1, 2]),
  t_stat = diff_estim / diff_SE,
  df = df.residual(m),
  p_value = 2 * pt(abs(t_stat), df = df, lower.tail = FALSE)
)
#> # A tibble: 1 x 5
#>   diff_estim diff_SE t_stat    df  p_value
#>
#> 1     -0.402  0.0251  -16.0   497 6.96e-47
```

This gives the exact same results as the first method! (\(t^2 = (-16)^2 = 256\)) is the same as the *F*-value from the `anova()` test.)

Method 3: `emmeans` <3

We can estimate the slopes from the model using `emmeans`, and then, with some trickery, compare them!

```
library(emmeans)
```

```
trends <- rbind(
  emtrends(m, ~1, "xtra_hours"),
  emtrends(m, ~1, "n_comps")
)

# clean up so it does not error later
trends@grid$`1` <- c("xtra_hours", "n_comps")
trends@misc$estName <- "trend"

trends
#>  1          trend     SE  df lower.CL upper.CL
#>  xtra_hours 0.811 0.0174 497    0.772    0.850
#>  n_comps    0.409 0.0174 497    0.370    0.448
#>
#> Confidence level used: 0.95
#> Conf-level adjustment: bonferroni method for 2 estimates
```

Compare them:

```
pairs(trends)
#>  contrast             estimate     SE  df t.ratio p.value
#>  xtra_hours - n_comps    0.402 0.0251 497 16.023  <.0001
```

Once again - exact same results!

Method 4: `lavaan`

The `lavaan` package for latent variable analysis and structural equation modeling allows for parameter constraining. So let's do just that:

```
library(lavaan)

m0 <- sem("salary ~ a * xtra_hours + a * n_comps", data =
hardlyworkingZ)
m <- sem("salary ~ xtra_hours + n_comps", data = hardlyworkingZ)

anova(m0, m)
#> Chi-Squared Difference Test
#>
#>    Df    AIC    BIC  Chisq Chisq diff Df diff Pr(>Chisq)
#> m   0 476.04 488.69   0.00
#> m0  1 682.26 690.69 208.22     208.22       1  < 2.2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This too yields similar results! (Only slightly different due to different estimation methods.)

Summary

That's it really - these 4 simple methods have wide applications to GL(M)M's, SEM, and more.

Enjoy!