Replicating the stata results

Daniel Heimgartner

August 20, 2024

This is an exploratory exercise to replicate the regression results as reported in [?]. The key goal is to model response rates as a function of the response burden score (essentially a univariate model). However, there are two complicating factors at play: First, the response rate is bounded between 0 and 100 and second, we would like to use clustered standard errors (since distinct survey waves for the same overall study are distinct observations but of course not independent).

1 Loading the data

```
> rr <- response_rates
> ## similar sample as in paper
> rr <- rr[1:67, ]
> rr <-
+   rr %>%
+   select(year, authors, survey_id,
+          response_rate,
+          response_burden_score,
+          sample_size,
+          matches("^flag")) %>%
+   filter(sample_size >= 10)
>
> # head(rr)
```

2 First replication attempt

In the paper they talk about a logistic regression model:

$$\log\left(\frac{R_i}{100 - R_i}\right) = \beta_0 + \beta_1 \frac{B_i}{1000} + \varepsilon_i \tag{1}$$

However, I rather think it is a linear regression model with a logistic transformation of the response. Logistic regression in my understanding refers to a model with a logistic link function mapping a latent continuous variable on $[0, 1]$.

Transforming the data and calling `lm`:

```
> ## no weights
> X <-
+   rr %>%
+   mutate(y = log(response_rate / (100 - response_rate)),
+          x = response_burden_score / 1000) %>%
+   mutate(across(matches("^flag"), function(x) as.numeric(x)))
> fit <- lm(y ~ 0 + x + flag_no_no + flag_yes_no + flag_yes_yes, data = X)
> summary(fit)

Call:
lm(formula = y ~ 0 + x + flag_no_no + flag_yes_no + flag_yes_yes,
    data = X)

Residuals:
    Min      1Q  Median      3Q     Max
-1.5077 -0.3851  0.0550  0.2958  2.0471

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
x             -0.3853     0.1437  -2.681   0.0094 **
flag_no_no    -1.1149     0.1444  -7.722 1.20e-10 ***
flag_yes_no    0.8266     0.1669   4.953 5.93e-06 ***
flag_yes_yes   1.2437     0.2320   5.362 1.29e-06 ***
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.6869 on 62 degrees of freedom
Multiple R-squared:  0.7204,        Adjusted R-squared:  0.7023
F-statistic: 39.93 on 4 and 62 DF,  p-value: < 2.2e-16
```

3 Weighted attempt

Further, the observations are "weighted". From the stata file I get that the weights are the square root of the sample size (number of respondents):

```
> ## with weights (sqrt(sample_size))
> fit <- lm(y ~ 0 + x + flag_no_no + flag_yes_no + flag_yes_yes,
+           weights = sqrt(sample_size),
+           data = X)
> summary(fit)

Call:
lm(formula = y ~ 0 + x + flag_no_no + flag_yes_no + flag_yes_yes,
    data = X, weights = sqrt(sample_size))

Weighted Residuals:
     Min       1Q   Median       3Q      Max
-11.7157  -1.7051   0.3219   1.7132   8.4562
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
x             -0.5765     0.1972  -2.923  0.00483 **
flag_no_no    -1.1082     0.1359  -8.152 2.16e-11 ***
flag_yes_no    0.8371     0.1629   5.139 2.98e-06 ***
flag_yes_yes   1.5298     0.2691   5.684 3.78e-07 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.474 on 62 degrees of freedom
Multiple R-squared:  0.7891,	Adjusted R-squared:  0.7754
F-statistic: 57.98 on 4 and 62 DF,  p-value: < 2.2e-16
```

The estimates are very close to the ones reported by [?]!

Some further computations and comparing to the intercept-only model:

```
> logLik(fit)

'log Lik.' -69.99197 (df=5)

> AIC(fit)

[1] 149.9839

> ## intercept only model
> fit0 <- lm(y ~ 1, data = X, weights = sqrt(sample_size))
> summary(fit0)

Call:
lm(formula = y ~ 1, data = X, weights = sqrt(sample_size))

Weighted Residuals:
    Min      1Q  Median      3Q     Max
-20.039  -4.136   1.770   5.052  14.972

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.4369     0.1552  -2.816  0.00643 **
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.975 on 65 degrees of freedom

> McFadden <- 1 - (logLik(fit) / logLik(fit0))
> as.numeric(McFadden)

[1] 0.4045634
```

4 Clustered standard errors

The least square estimates are still ok, but we can't rely on the standard errors if we expect omega to have a block structure (similar errors for different survey waves of the same study):

```
> ## clustered se at the survey level (survey_id)
> ## https://www.r-bloggers.com/2021/05/clustered-standard-errors-with-r/
> clustered_se <- lmtest::coeftest(fit, vcov = sandwich::vcovCL, cluster= ~survey_id)
> clustered_se

t test of coefficients:

             Estimate Std. Error t value  Pr(>|t|)
x            -0.57652    0.16562 -3.4810 0.0009207 ***
flag_no_no   -1.10820    0.19030 -5.8233 2.210e-07 ***
flag_yes_no   0.83709    0.14623  5.7244 3.237e-07 ***
flag_yes_yes  1.52978    0.20505  7.4606 3.418e-10 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

> ## alternatively, first compute block omega
> vc_mat <- sandwich::vcovCL(fit, cluster = ~survey_id)
> clustered_se_alt <- lmtest::coeftest(fit, vcov = vc_mat)
> clustered_se_alt  # same as clustered_se

t test of coefficients:

             Estimate Std. Error t value  Pr(>|t|)
x            -0.57652    0.16562 -3.4810 0.0009207 ***
flag_no_no   -1.10820    0.19030 -5.8233 2.210e-07 ***
flag_yes_no   0.83709    0.14623  5.7244 3.237e-07 ***
flag_yes_yes  1.52978    0.20505  7.4606 3.418e-10 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

5 Backtransform the response

As we have estimated the model on the log-odds, but are interested in the expected response rates, we have to backtransform the response. Be aware that `predict(fit, interval = "confidence")` does use the regular standard errors. But that's ok to start with:

```
> ## backtransform (no clustered se!)
> pred <- as.data.frame(predict(fit, interval = "confidence"))
> backtransform <- function(y) {
+   y_star <- exp(y) / (1 + exp(y)) * 100
+   y_star
+ }
```

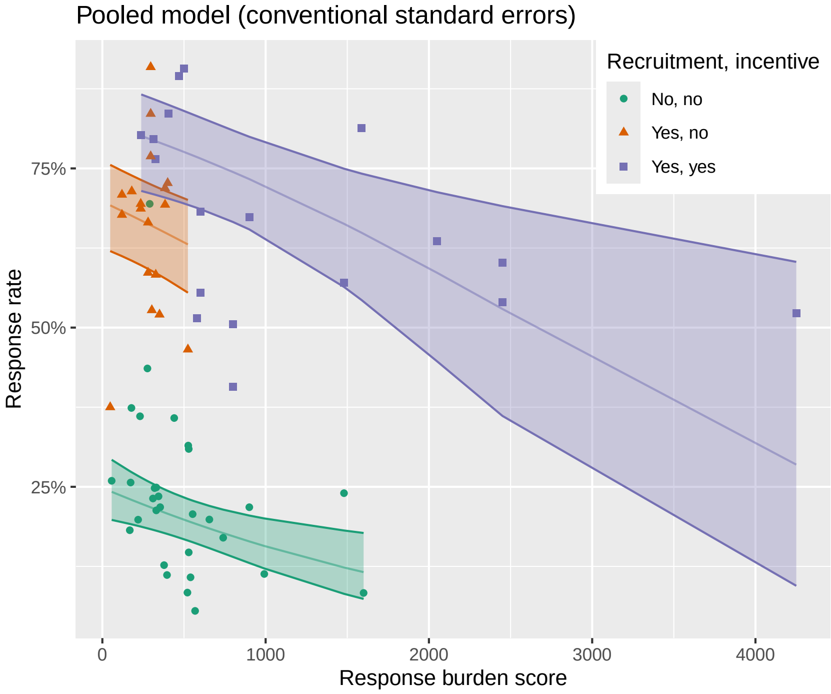
```
> pred_backtrans <-
+   pred %>%
+   mutate(pred_rr = backtransform(fit),
+          ci_lower = backtransform(lwr),
+          ci_upper = backtransform(upr))
```

Helper for plotting:

```
> plot_fun <- function(data, title) {
+   df <-
+     data %>%
+     select(pred_rr, ci_lower, ci_upper) %>%
+     bind_cols(select(X, response_rate, response_burden_score, matches("flag"))) %>%
+     pivot_longer(matches("flag")) %>%
+     filter(value == 1) %>%
+     mutate(key = factor(name)) %>%
+     select(-name, -value)
+
+   df %>%
+     mutate(key = case_when(key == "flag_no_no" ~ "No, no",
+                            key == "flag_yes_no" ~ "Yes, no",
+                            key == "flag_yes_yes" ~ "Yes, yes")) %>%
+     ggplot(aes(x = response_burden_score, group = key, shape = key, col = key)) +
+     geom_point(aes(y = response_rate)) +
+     geom_line(aes(y = pred_rr), alpha = 0.5, show.legend = FALSE) +
+     geom_ribbon(aes(ymin = ci_lower, ymax = ci_upper, fill = key), alpha = 0.3, show.legend =
+     scale_y_continuous(labels = scales::label_percent(scale = 1)) +
+     scale_color_brewer(type = "qual", palette = 2) +
+     scale_fill_brewer(type = "qual", palette = 2) +
+     labs(x = "Response burden score", y = "Response rate",
+          col = "Recruitment, incentive", shape = "Recruitment, incentive",
+          title = title) +
+     theme(legend.position = c(1, 1),
+           legend.justification = c("right", "top"))
+ }
```

Visualize:

```
> plot_fun(pred_backtrans, "Pooled model (conventional standard errors)")
```



6 Backtransform with clustered standard errors

Now let's account for the clustered standard errors even in the confidence intervals:

```
> ## use clustered standard errors in prediction
> pred_cluster_se <- function(fit, vcov) {
+   X <- model.matrix(fit)
+   se <- sqrt(rowSums((X %*% vcov) * X))
+   se
+ }
> cl_se <- pred_cluster_se(fit, vc_mat)
> cl_pred <-
+   data.frame(fit = predict(fit)) %>%
+   mutate(cl_se = cl_se,
+          ci_lower = fit - 1.96 * cl_se,
+          ci_upper = fit + 1.96 * cl_se)
> cl_pred_backtrans <-
+   cl_pred %>%
+   mutate(pred_rr = backtransform(fit),
+          ci_lower = backtransform(ci_lower),
+          ci_upper = backtransform(ci_upper))
> plot_fun(cl_pred_backtrans, "Pooled model (clustered standard errors)")
```

The plots are very similar...

7 Outlook

- The paper estimates different models for the categories (recruitment x incentive).
- Include the most recent surveys. However, the TimeUse+ point that we should use heteroscedastic errors as the error variance for recruitment is yes and incentive is yes increases with the response burden... Also TimeUse+ is a tracking study. How did Caro differentiate between drop-outs because of the response burden vs. tracking?

References