Understanding Variance Estimation: Model-Based and Robust

Ashley I Naimi

Fall 2024

Contents

1 Introduction

Statistics is primarily focused on capturing or quantifying the uncertainty that results from random variation. This variation is often attributed to several sources, including non-response (individuals who should be in the study's sampling frame are not included), missing data (individual's recruited into the study are missing information on certain variables), coverage frame errors (e.g., inclusion and/or exclusion criterion fail to cover population of interest), measurement, and processing errors.

Note that every one of these issues can result in one of two problems: systematic error, including measurement error, selection bias, and other biases; and non-systematic error.

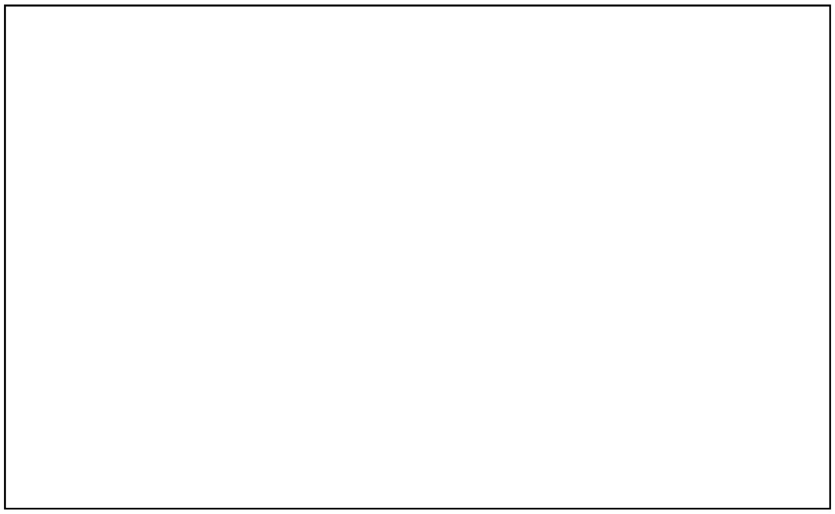
Typical variance estimators that we'll discuss are second order estimators, in the sense that they depend on a first order of estimation. Recall, the formula for variance for a random variable Y:

$$V(Y) = \frac{1}{n-1}\sum_{i=1}^{N}(y_i - \bar{y})^2$$

where the standard deviation is the square root of this variance. This estimator is "second order" in the sense that it's validity depends on $\bar{x}$. If the mean estimator is invalid (e.g., biased or inconsistent), the variance will also not be valid (and other statistics derived therefrom, such as confidence intervals or p values, will also be invalid).

Typical formulations of the variance of an estimator are often motivated from the super population perspective.[1]

[1] Other formulations exist, which lead to distinctions between sample versus population average treatment effects. See Balzer et al (2016) Stat Med. 35(21): 3717–3732 for relevant references and descriptions.



Let's use the above box to demonstrate the super-population.[2] We can draw from this super-population to construct samples. We can use regression in each sample to estimate a (say) risk difference $\hat{\theta}$ for an exposure-outcome association, and then compute the standard error (variance) of this estimator for the risk difference.

[2] It's important to understand that the "super population" is a mathematical abstraction. This sometimes comes up when an organization / institute conducts a census of a population, where everyone in the population is counted. The takeaway from this is, even when we have a full census of the population, it's important to still compute variance, standard errors, confidence intervals, p values, and other inferential statistics.

The standard error is meant to estimate the standard deviation of the risk differences $\hat{\theta}$ that we'd obtain if we took random samples from the super population and estimated the risk difference in each sample.

To see this principle in action, we can simulate data from a super population, and compute the standard deviation of the risk differences we estimate, and compare them to the average standard error we obtain from each sample. These two numbers should be the same:

```r
packages <- c("data.table", "tidyverse", "skimr", "here", "mvtnorm")

for (package in packages) {
    if (!require(package, character.only = T, quietly = T)) {
        install.packages(package, repos = "http://lib.stat.cmu.edu/R/CRAN")
    }
}

for (package in packages) {
    library(package, character.only = T)
}
```

```r
# CREATE EXPIT AND LOGIT FUNCTIONS
expit <- function(x) {
    exp(x)/(1 + exp(x))
}
logit <- function(x) {
    log(x/(1 - x))
}

res <- NULL
for (i in 1:20000) {

    set.seed(i)
    n = 1000
    p = 10

    ## CONFOUNDERS
    sigma <- matrix(0, nrow = p, ncol = p)
    diag(sigma) <- 1
    c <- rmvnorm(n, mean = rep(0, p), sigma = sigma)

    ## EXPSOURE
    pi_x <- expit(-2 + log(2) * c[, 1] + log(2) * c[, 2])
    x <- rbinom(n, 1, pi_x)

    ## OUTCOME
    pi_m <- expit(-2 + log(2) * x + log(2) * c[, 1] + log(2) *
        c[, 2])
    y <- rbinom(n, 1, pi_m)

    a <- data.frame(y, x, c)

    names(a) <- c("y", "x", paste0("c", 1:10))

    model_mu <- glm(y ~ x + c1 + c2, data = a, family = binomial("logit"))
    estimate <- summary(model_mu)$coefficients[2, 1]

    std.err <- summary(model_mu)$coefficients[2, 2]

    res <- rbind(res, cbind(estimate, std.err))

}

head(res)
```

```
##       estimate   std.err
## [1,] 0.6876913 0.2213973
## [2,] 0.2705164 0.2207663
## [3,] 0.9554663 0.2206441
## [4,] 0.6242044 0.2176443
## [5,] 0.7147260 0.2293350
## [6,] 0.6572079 0.2191512
```

```
res <- data.frame(res)

round(sd(res$estimate), 3)
```

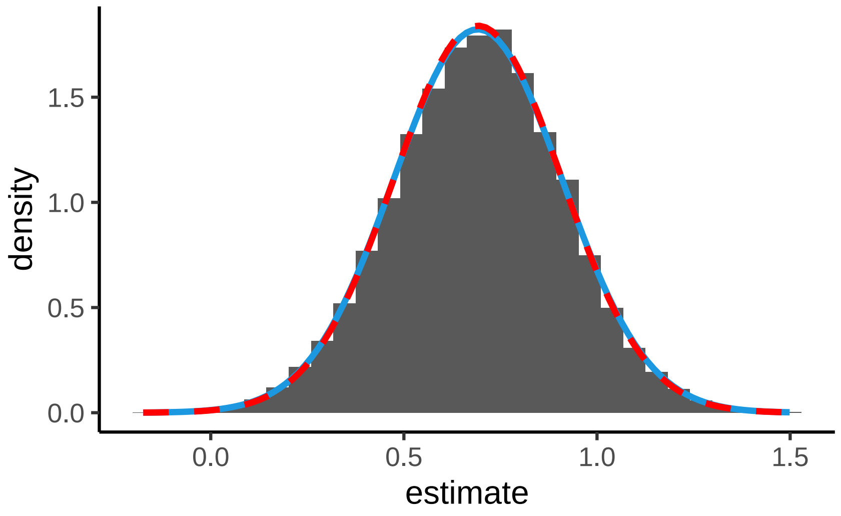
```
## [1] 0.219
```

```
round(mean(res$std.err), 3)
```

```
## [1] 0.217
```

From this, we can see that, on average, the standard errors capture the standard deviation of the effect estimate (in this case, log-OR).

```
ggplot(res) + geom_histogram(aes(x = estimate, after_stat(density))) +
    geom_histogram(aes(x = estimate, after_stat(density))) +
    stat_function(fun = dnorm, args = list(mean = mean(res$estimate),
        sd = sd(res$estimate)), col = "#1b98e0", size = 1) +
    stat_function(fun = dnorm, args = list(mean = mean(res$estimate),
        sd = mean(res$std.err)), col = "red", size = 1, linetype = "dashed") +
    theme_classic()
```



2 Model-Based Standard Errors

Let's dive a bit deeper in how the variance of a parameter estimate is obtained from a regression model. To keep concepts relatively simple, we'll work with linear models in a simple data setting: Continuous outcome with a binary treatment.

```
bmi_data <- read_csv(here("data", "cluster_trial_data_bmi.csv"))

bmi_data %>%
    print(n = 5)
```

```
## # A tibble: 20 x 4
##       ID   BMI treatment practice
##    <dbl> <dbl>     <dbl>    <dbl>
## 1      1  26.2         1        1
## 2      2  27.1         1        1
## 3      3  25           1        2
## 4      4  28.3         1         2
## 5      5  30.5         1         3
## # i 15 more rows
```

These data were obtained from a cluster randomized trial example in which 10 clinical practices were randomly assigned to two treatment groups, and BMI at year 1 was assessed as the outcome (Campbell, 2006).

For the moment, we'll ignore the clustering, and consider the standard ordinary least squares regression model, with y being BMI and X being the treatment indicator:

$$E(y \mid X) = \beta_0 + \beta_1 X$$

The point estimates for this model can be obtained using the standard OLS estimator:

$$\hat{\beta} = (X^{\intercal}X)^{-1}X^{\intercal}y$$

With the cluster_trial data, we get this:

```r
X <- cbind(1, bmi_data$treatment)

y <- bmi_data$BMI

beta_hat <- solve(t(X) %*% X) %*% t(X) %*% y

beta_hat
```

```
##        [,1]
## [1,] 28.39
## [2,]  0.42
```

(note how they exactly match the estimates from the `lm()` output)

```r
mod <- lm(BMI ~ treatment, data = bmi_data)

mod
```

```
##
## Call:
## lm(formula = BMI ~ treatment, data = bmi_data)
##
## Coefficients:
## (Intercept)    treatment
##       28.39         0.42
```

Once the parameter estimates are obtained, next step is to estimate their standard errors. The equation to obtain standard errors for the parameter estimates is:

$$V(\hat{\beta}) = (X^\intercal X)^{-1} X^\intercal \Sigma X (X^\intercal X)^{-1}$$

This equation has two important pieces:

1. $(X^\intercal X)^{-1}$
2. $X^\intercal \Sigma X$

Because the second piece is "sandwiched" between the first, this formula is often called the sandwich variance estimator.

The first piece is the inverse of the matrix multiplication of the design matrix. In the cluster trial data, the design matrix is a column of 1s (for the intercept), and a column representing the treatment variable.

The second piece has two components. The first is the variance matrix Σ, which is a square matrix with the variance of the residuals on the diagonal:

```
s <- sum((y - X %*% beta_hat)^2)/(nrow(bmi_data) - 2)  ## note: deviance generalizes residual sum
## of squares for GLM
s
```

```
## [1] 18.13322
```

```
Sigma <- s * diag(nrow(bmi_data))  ## this is the key ingredient in the 'meat'
```

Basically, Σ (`Sigma`) is a 20×20 matrix (recall, there are 20 observations in the data) with 18.1332222 on the diagonal and zeros everywhere else.

This diagonal value is obtained as the sum of the squared model residuals divided by the degrees of freedom, and represents the "variation" in the model.[3] Hopefully, the diagonal of this Σ matrix looks familiar, but if not, note that the `X%*%beta_hat` term represents the model's predicted mean. So the diagonal represents as estimate of the variance of the outcome implied by the model.

[3] More generally (i.e., when a GLM is used with a nonlinear link function), the diagonal value can be obtained using the deviance statistic.

The second piece is there to take this model variation, and translate it into the parameter space. In other words, the second piece is taking the model variation and "transforming" it so that it is represented as uncertainty in the parameter estimates.[4]

[4] More technically (linear algebra!), Σ undergoes a basis change from $\mathbb{R}^N$ to $\mathbb{R}^p$, where N is the sample size and p is the number of parameters.

Putting each piece together, we have:

```
var_beta <- solve(t(X) %*% X) %*% (t(X) %*% Sigma %*% X) %*%
    solve(t(X) %*% X)  # recall: for a square matrix t(X)%*%X, solve() returns the inverse


var_beta
```

```
##            [,1]      [,2]
## [1,]  1.813322 -1.813322
## [2,] -1.813322  3.626644
```

The diagonal of `var_beta` is the variance of the parameter estimates (its square root thus the standard error), while the off-diagonal represents the covariance between the parameter estimates. Note, again, how this variance covariance matrix `var_beta` is identical to the model output:

```
vcov(mod)
```

```
##             (Intercept) treatment
## (Intercept)    1.813322 -1.813322
## treatment     -1.813322  3.626644
```

Importantly, for this **model-based SE**, the variance on the diagonal of Σ is the same for each person in the data: 18.1332222. This is reflective of one of the central assumptions underlying the model: in the OLS case, homoscedasticity. Alternatively, if we were to use a generalized linear model with a Gaussian distribution and identity link (a mathematically equivalent approach), this assumption is captured as "constant variance", which is the σ^2 we would use to parameterize the Gaussian distribution used in the model.

This homoscedasticity or constant variance assumption may not be correct or ideal. Furthermore, this variance equation is technically derived from a Gaussian distribution. If we use this model to analyze binary outcome data, this variance equation can be way off. This is precisely what the robust variance estimator is meant to address.

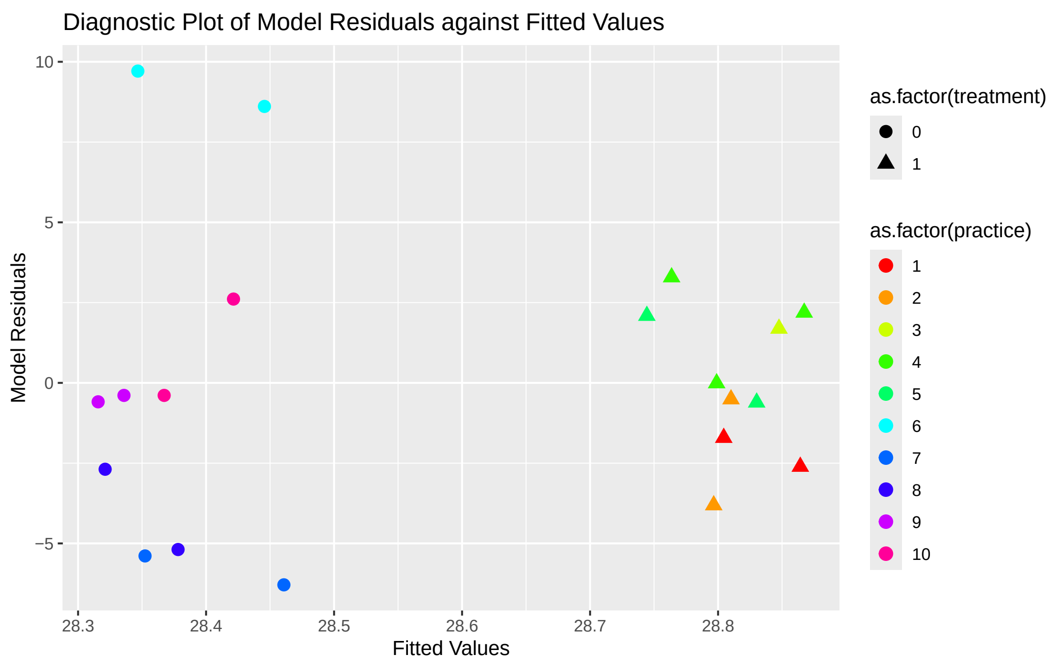
3 Robust (Sandwich) Standard Errors

One of the main problems with the above variance estimator is that it assumes the data for each observation is generated from a Gaussian distribution, **with the same variance for each** (constant variance). Let's explore this assumption.

```r
plot1 <- augment(mod) %>%
    mutate(practice = bmi_data$practice) %>%
    ggplot(.) + geom_point(aes(x = jitter(.fitted), y = .resid,
    color = as.factor(practice), shape = as.factor(treatment)),
    size = 3) + ylab("Model Residuals") + xlab("Fitted Values") +
    scale_color_manual(values = rainbow(10)) + ggtitle("Diagnostic Plot of Model Residuals against Fitte

ggsave("./diagnostic.pdf", width = 8, height = 5)
ggsave("./diagnostic.png", width = 8, height = 5)
```

This figure is a basic diagnostic plot, showing how the residuals of the model are spread as a function of the model's fitted values. In this case, we only have two fitted values (jittered for demonstration) because the model has only one binary treatment variable.



This plot shows a few important things. First, and most importantly, the spread of the residuals on the y-axis is far larger for lower fitted values than for higher fitted values. Second, This difference in spread is perfectly correlated with treatment (circles, or untreated, are spread out more than triangles, or treated), and also correlated with practice: practices 6, 7, 8, 9 and 10 have more spread than the first five. This make sense, since this was a cluster randomized trial and the first five practices were treated and the remaining were placebo.

These problems are a good example of heteroscedasticity, or a violation of the "constant variance" assumption. Untreated practices seem to have a higher variance than treated ones. Because of this, the homoscedastic (or constant variance) error assumption mentioned above may not be the best way to analyse these data. That is, the value of 18.1332222 on the diagonal of the Σ matrix may be too small for untreated practices, and too large for treated practices. This would generally lead to problems with inference−that is, standard errors that do not yield confidence intervals with nominal coverage, or inferential tests with incorrect error rates.

This is why robust variance estimators were developed. Several exist. We will start with the HC1 estimator[5]

[5] note "HC" stands for heteroscedastic consistent

The HC1 estimator essentially uses the squared residuals in $\mathbf{s}$ as above, but instead of summing over them, it leaves them at their original (i.e., unsummed) values. So, while the diagonal of the Σ matrix in the variance equation above was set to 18.1332222 for all 20 observations in the sample, the HC1 estimator will simply use each individuals' residual, without summing over them:

```
s_hc1 <- ((y - X %*% beta_hat)^2) * (nrow(bmi_data)/(nrow(bmi_data) -
    2))  ## note: no sum

round(s_hc1, 2)
```

```
##          [,1]
##  [1,]    7.57
##  [2,]    3.25
##  [3,]   16.13
##  [4,]    0.29
##  [5,]    3.17
##  [6,]    0.00
##  [7,]    5.33
##  [8,]   12.03
##  [9,]    0.41
## [10,]    4.85
## [11,]   82.37
## [12,]  104.76
## [13,]   43.96
## [14,]   32.28
## [15,]   29.93
## [16,]    8.04
## [17,]    0.39
## [18,]    0.17
## [19,]    0.17
## [20,]    7.57
```

The largest value in this list of individual-level variances is 104.76, which is much larger than it's lowest value, basically 0, again suggesting that constant variance is probably not a good assumption.

To implement the HC1 variance estimator, we simply replace the diagonals of 18.1332222 for each person with these diagonals of individual-level residuals multiplied by the degrees of freedom for the HC1 estimator (in this case, $N/(N-p)$). We can then use the same equation for the variance, but this time with the new Σ_{HC1}:

```r
Sigma_HC1 <- diag(nrow(bmi_data))
diag(Sigma_HC1) <- s_hc1

## meatHC(mod,type='HC1') = (t(X)%*%Sigma_HC1%*%X)/20

var_beta_hc1 <- solve(t(X) %*% X) %*% (t(X) %*% Sigma_HC1 %*%
    X) %*% solve(t(X) %*% X)  # recall: for a square matrix t(X)%*%X, solve() returns the inverse

var_beta_hc1
```

```
##           [,1]      [,2]
## [1,]  3.096322 -3.096322
## [2,] -3.096322  3.626644
```

The diagonal of `var_beta_hc1` is the HC1 version of the robust variance of the parameter estimates (its square root thus the standard error), while the off-diagonal represents the corresponding covariance between the parameter estimates. Note, again, how this variance covariance matrix `var_beta_hc1` is identical to the output we would get if we used the robust variance estimator in R from the sandwich package:

```r
library(sandwich)

vcovHC(mod, type = "HC1")
```

```
##             (Intercept) treatment
## (Intercept)    3.096322 -3.096322
## treatment     -3.096322  3.626644
```

What about different variations of the HC variance estimators? There are many types programmed in R, including “HC3” (the default), “const”, “HC”, “HC0”, “HC1”, “HC2”, “HC4”, “HC4m”, and “HC5”.

The default version replaces the HC1 variance obtained from the scaled residuals for each observation with:

$$\frac{\hat{u}_i^2}{(1-h_i)^2}$$

where $\hat{u}_i$ are the estimated residuals from the model, in our case `(y - X%*%beta_hat)^2`, and where h_i are the leverage points for each observation obtained from the model.

Leverage values are obtained as the diagonal entries of the hat matrix, defined as:

$$\mathbf{H} = X(X^\intercal X)^{-1}X^\intercal.$$

In R, these leverage values can be calculated easily with our design matrix as:

```
H <- X %*% solve(t(X) %*% X) %*% t(X)

h <- diag(H)

h
```

```
##  [1] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
## [20] 0.1
```

which match exactly the leverage values obtained from the fitted model:

```
hatvalues(mod)
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
## 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
```

With these, we can then compute the diagonal of Σ for the HC3 (default) estimator:

```
s_hc3 <- ((y - X %*% beta_hat)^2)/((1 - h)^2)
```

```
Sigma_HC3 <- diag(nrow(bmi_data))
diag(Sigma_HC3) <- s_hc3

var_beta_hc3 <- solve(t(X) %*% X) %*% (t(X) %*% Sigma_HC3 %*%
    X) %*% solve(t(X) %*% X)  # recall: for a square matrix t(X)%*%X, solve() returns the inverse

var_beta_hc3
```

```
##           [,1]      [,2]
## [1,]  3.440358 -3.440358
## [2,] -3.440358  4.029605
```

Once again, this variance covariance matrix `var_beta_hc3` is identical to the output for HC3 in R:

```
vcovHC(mod, type = "HC3")
```

```
##             (Intercept) treatment
## (Intercept)    3.440358 -3.440358
## treatment     -3.440358  4.029605
```

References

Michael J Campbell. *Statistics at square two: understanding modern statistical applications in medicine*. Blackwell, 2006.