

Variance Estimation

Ashley I Naimi

October 2026

Contents

1	Introduction	2
2	Model-Based Standard Errors	5
3	Robust (Sandwich) Standard Errors	10
3.1	Robust Standard Errors for a GLM	16
3.2	Cluster-Robust Standard Errors and the Link to GEE	18
4	The Bootstrap	21
4.1	The Bootstrap Principle	22
4.2	Example Data	24
4.3	Example Analysis	24
4.4	Normal Interval Bootstrap	25
4.5	Percentile Bootstrap	26
4.6	Bias-Corrected and Accelerated Intervals	27
4.7	Bootstrap Confidence Intervals in the NHEFS Data	28
5	Conclusion	34

1 Introduction

This week we focus on variance (namely, standard error) estimation, and about the three ways of obtaining it that you will encounter: the model-based estimator that comes with the likelihood, the robust (or sandwich) estimator, and the bootstrap.

Statistics is primarily focused on capturing or quantifying the uncertainty that results from random variation. Error in a study is often attributed to several sources, including non-response (individuals who should be in the study's sampling frame are not included), missing data (individuals recruited into the study are missing information on certain variables), coverage frame errors (e.g., inclusion and/or exclusion criteria fail to cover the 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 (random) error. Standard errors, confidence intervals, and p values speak only to the second of these. Nothing in this lecture addresses the first.

The variance estimation is, in some sense, a second order estimation problem, in the sense that it depends on a first order of estimation. Recall the formula for the sample variance of 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 is built around a first order estimate, the sample mean $\bar{y}$. The same holds for the standard error of a regression coefficient: it is computed around the coefficient estimate, describing the spread of that estimator. If the estimator is biased for the quantity we care about, no standard error will fix that, and the confidence intervals and p values derived from it will be centered on the wrong number.

Typical formulations of the variance of an estimator are often motivated from the super population perspective.¹ Picture the super-population as a very large box of individuals.² We can draw from this super-population to construct samples. We can use regression in each sample to estimate (say) a log odds ratio $\hat{\theta}$ for an exposure-outcome association, and then compute the standard

What to read: Mansournia et al. Reflection on modern methods: demystifying robust standard errors for epidemiologists. *Int J Epidemiol* 2021;50:346-351

¹ Other formulations exist, which lead to distinctions between sample versus population average treatment effects. See [Balzer et al. \(2016\)](#) for relevant references and descriptions.

² The “super population” is a mathematical abstraction. This sometimes comes up when an organization or institute conducts a census of a population, where everyone in the population is counted. The takeaway from this is that, under the super-population view, even when we have a full census of the population, it is still appropriate to compute variances, standard errors, confidence intervals, p values, and other inferential statistics.

error of this estimator.

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

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

```
# 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 = 1e3
  p = 10

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

  ## EXPOSURE
  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, the log odds ratio). The figure below shows the distribution of the 20,000 estimates, with two normal curves laid over it: one whose spread is the standard deviation of the estimates, and one whose spread is the average of the standard errors. The two curves are nearly indistinguishable.

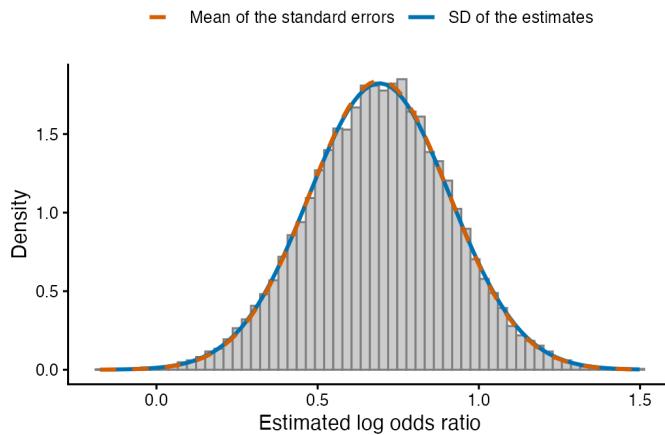


Figure 1: Distribution of the log odds ratio estimated in 20,000 samples of 1,000 drawn from the same super-population. The solid curve is a normal distribution with the standard deviation of the 20,000 estimates; the dashed curve is a normal distribution with the average of the 20,000 model-based standard errors.

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](#)). There are two patients per practice, and twenty patients in all. The same twenty observations appear as Table 1 of this week's assigned reading ([Mansournia et al., 2021](#)), which works through the model-based and cluster-robust standard errors that we compute by hand below.

For the moment, we'll ignore the clustering (we return to it at the end of the next section), and consider the standard ordinary least squares regression model, with Y being BMI and X being the treatment indicator:

$$E(Y | X) = \beta_0 + \beta_1 X$$

The point estimates for this model can be obtained using the standard OLS estimator we saw in Weeks 4 and 6:

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

With the cluster trial data, we get this:

```
X <- cbind(1,bmi_data$treatment) ## design matrix

y <- bmi_data$BMI ## outcome

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)

```
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, the next step is to estimate their standard errors. The matrix equation to obtain the variance of the parameter estimates of our OLS estimator:

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

This equation has two important pieces:

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

Because the second piece is “sandwiched” between two copies of the first, this layout is called the sandwich form, with the outer pieces often called the “bread” and the inner piece the “meat”. The model-based variance is the special case of this form in which Σ is filled in from the model’s own variance assumption. The robust estimators of the next section keep the same “sandwich” structure, but change only what goes into Σ .

The first piece is the inverse of the cross-product 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 outcome around its model mean on the diagonal:

```
## residual sum of squares over degrees of freedom
s <- sum((y - X%*%beta_hat)^2)/(nrow(bmi_data) - 2)

s
```

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

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

Basically, Σ (Sigma) is a 20×20 matrix (recall, there are 20 observations in the data) with 18.13 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. 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 an estimate of the variance of the outcome implied by the model.

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.³

Putting each piece together, we have:

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

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

³ 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.

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
```

Because Σ is a constant times the identity matrix here, the sandwich collapses: $(X^\top X)^{-1} X^\top (s^2 I) X (X^\top X)^{-1} = s^2 (X^\top X)^{-1}$, which is the textbook variance of the least squares estimator. Writing it in sandwich form is what lets us see, in the next section, exactly which piece the robust estimator changes.



Note: The Model-Based Variance of a GLM

The same layout holds for every generalized linear model, with two changes. The bread becomes $(X^\top W X)^{-1}$, where W is the diagonal matrix of IRLS working weights from Week 6, and the diagonal of Σ holds, for each person, the dispersion times the variance function evaluated at their fitted mean, $\phi V(\hat{\mu}_i)$. For a logistic model that is $\hat{\mu}_i(1 - \hat{\mu}_i)$; for a Poisson model it is $\hat{\mu}_i$; for the Gaussian model above it is the constant s^2 . With a canonical link the working weights are the variance function itself, so $X^\top \Sigma X = \phi X^\top W X$ and the sandwich collapses to

$$V(\hat{\beta}) = \phi (X^\top W X)^{-1},$$

which is what `vcov()` returns for a `glm` object. The dispersion ϕ is fixed at one for the binomial and Poisson families. When it is estimated (Gaussian, gamma, and the quasi families), R uses the Pearson statistic divided by the residual degrees of freedom. The point to take away is that the model-based Σ is never estimated from the data person by person. It is a formula, $\phi V(\hat{\mu}_i)$, that the family dictates.

Importantly, for this **model-based SE**, the variance on the diagonal of Σ is the same for each person in the data: 18.13. 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 linear model to analyze binary outcome data, the constant diagonal is wrong by construction, because the variance of a binary outcome is $\mu(1 - \mu)$ and changes with the fitted mean. The same mismatch arose in Week 6 when a Poisson family was used to estimate a risk ratio for a binary outcome. 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 in the BMI data:

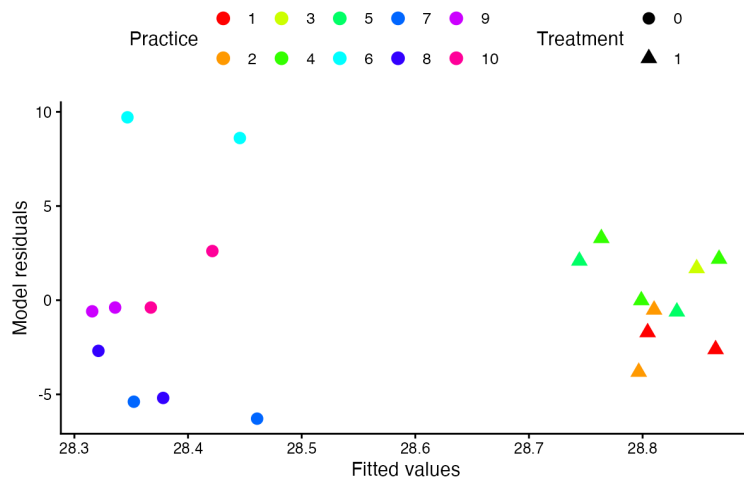


Figure 2: Residuals of the linear model for BMI against its fitted values in the cluster trial data. Fitted values are jittered because the model has only one binary covariate and therefore only two distinct fitted values. Colors are practices, shapes are treatment arms (circles untreated, triangles treated).

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 makes 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 18.13 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 one reason why robust variance estimators were developed ([Huber, 1967](#); [Eicker, 1967](#); [White, 1980](#)). Several exist. We will start with the HC1 estimator.⁴

The HC1 estimator essentially uses the squared residuals in 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 18.13 for all 20 observations in the sample, the HC1 estimator will simply use each individual’s squared residual, without summing over them:

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

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
```

⁴ Note “HC” stands for heteroscedasticity consistent. The numbered variants HC0 through HC3 are from [MacKinnon and White \(1985\)](#).

```
## [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 its 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 diagonal of 18.13 for each person with these individual-level squared residuals, multiplied by a degrees of freedom correction (in this case, $N/(N - p)$). We can then use the same equation for the variance, but this time with the new Σ_{HC1} :

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

var_beta_hc1 <- solve(t(X)%*%X) %*%
  (t(X)%*%Sigma_HC1%*%X) %*%
  solve(t(X)%*%X)

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 (Zeileis, 2006):

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

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

Look closely at the two matrices. The variance of the treatment coefficient is 3.63 under the model-based estimator and 3.63 under HC1, and only the intercept's variance changed, from 1.81 to 3.1. This is not a rounding coincidence but a feature of these data. With a single binary covariate, the intercept is the mean of the untreated arm and the slope is the difference between the two arm means. The robust variance of the intercept is the untreated arm's own residual variance divided by its size, and because the untreated are the more variable arm, it is larger than the pooled value the model-based estimator used. The variance of the difference in means adds the two arms' contributions, and with ten people per arm, the sum of the two arm-specific variances equals the pooled variance times $(1/10 + 1/10)$ exactly. Heteroscedasticity that lines up perfectly with a balanced binary covariate changes the uncertainty of the group means but not of their difference. In an unbalanced design, or with continuous covariates, the slope's variance would change as well.

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 squared residuals for each observation with:

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

where $\hat{u}_i = y_i - x_i^T \hat{\beta}$ are the estimated residuals from the model (the vector computed as `y - X%*%beta_hat` above), and where h_i are the leverage values for each observation obtained from the model.

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

$$H = X(X^T X)^{-1} X^T.$$

The hat matrix is what turns the observed outcomes into the fitted values ($\hat{y} = Hy$), and the leverage h_i measures how much observation i 's own outcome pulls on its own fitted value. 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
## [13] 0.1 0.1 0.1 0.1 0.1 0.1 0.1 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
## 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
## 14 15 16 17 18 19 20
## 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)

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
```

The table below collects the four estimators that are most often encountered. The only difference between them is what each person contributes to the diagonal of Σ .

Estimator	Diagonal of Σ	Treatment SE
Model-based	s^2 (one value for everyone)	1.904
HC0	$\hat{u}_i^2$	1.807
HC1	$\hat{u}_i^2 N/(N-p)$	1.904
HC2	$\hat{u}_i^2/(1-h_i)$	1.904
HC3	$\hat{u}_i^2/(1-h_i)^2$	2.007

The heteroscedasticity consistent family. $\hat{u}_i$ is the residual and h_i the leverage of observation i ; N is the sample size and p the number of coefficients. The last column is the standard error of the treatment coefficient in the cluster trial data under each estimator. HC1 and HC2 coincide here because every leverage is $p/N = 0.1$, so $1/(1-h_i)$ and $N/(N-p)$ are the same number.

HC0 is the original estimator of [White \(1980\)](#). HC1 rescales it by $N/(N-p)$, HC2 divides each squared residual by one minus its leverage, and HC3 divides by the square of that quantity. In large samples, where leverages shrink toward

zero and $N/(N-p)$ toward one, all four converge to the same answer. In small samples HC0 is known to be too small, and HC3 is the one recommended for general use (Long and Ervin, 2000).



Note: Robust Defaults Differ Across Software

Because the variants agree in large samples, software packages have made different choices about their defaults, and the choice matters in small samples. In R, `sandwich::vcovHC()` defaults to HC3 for both `lm` and `glm` objects, while `sandwich::sandwich()` (the one-line call used in Week 6) returns HC0. Stata's `vce(robust)` option is HC1. SAS offers HC0 through HC3 through the `HCCMETHOD` option in `PROC REG`, and the empirical ("sandwich") variance from the `REPEATED` statement in `PROC GENMOD` is of the HC0 type. When two analysts report different "robust" standard errors from the same data, this is usually the reason.

3.1 Robust Standard Errors for a GLM

The sandwich layout above is written for a linear model, but as the note box in the previous section explained, the bread of a GLM is $(X^T W X)^{-1}$ and its model-based Σ holds $\phi V(\hat{\mu}_i)$. The robust version keeps the bread and fills Σ with each person's squared residual, $(y_i - \hat{\mu}_i)^2$, exactly as HC0 did above. Here is the Week 6 example in which this matters most: the adjusted NHEFS model for above-median weight gain fit with a Poisson family and a log link, a common device for estimating a risk ratio from a binary outcome (Zou, 2004):

```
nhefs <- read_csv(here("data", "nhefs.csv")) %>%
  filter(!is.na(wt82_71)) %>%
  mutate(wt_delta = as.numeric(wt82_71 > median(wt82_71)),
         marital = ifelse(marital == 6, 5, marital),
         marital = ifelse(marital == 8, 2, marital))

f_adj <- wt_delta ~ qsmk + age + sex + race +
  factor(exercise) + factor(marital) + smokeintensity

mod_pois <- glm(f_adj, data = nehs,
               family = poisson("log"))

Xp <- model.matrix(mod_pois)
```

```

mup <- fitted(mod_pois)
yp <- nhefs$wt_delta

# bread:  $(X'WX)^{-1}$  with the Poisson working weights,
#  $W = \mu$ 
bread_p <- solve(t(Xp) %*% (mup * Xp))

# model-based meat:  $\Sigma = \text{diag}(\mu)$ 
# robust meat:  $\Sigma = \text{diag}((y - \mu)^2)$ 
meat_model <- t(Xp) %*% (mup * Xp)
meat_robust <- t(Xp) %*% ((yp - mup)^2 * Xp)

V_model <- bread_p %*% meat_model %*% bread_p
V_robust <- bread_p %*% meat_robust %*% bread_p

se_model <- sqrt(V_model["qsmk", "qsmk"])
se_robust <- sqrt(V_robust["qsmk", "qsmk"])

# what the software returns
se_vcov <- sqrt(vcov(mod_pois)["qsmk", "qsmk"])
se_vcovHC <- sqrt(vcovHC(mod_pois,
                        type = "HCO")["qsmk", "qsmk"])

# the Week 6 log-binomial model, for comparison
mod_logbin <- glm(f_adj, data = nhefs,
                 family = binomial("log"),
                 start = c(log(mean(nhefs$wt_delta)),
                           rep(0, 10)))
se_logbin <- summary(mod_logbin)$coefficients["qsmk", 2]

round(c(by_hand_model = se_model,
        vcov_model = se_vcov,
        by_hand_robust = se_robust,
        vcovHC_HCO = se_vcovHC,

```

```
logbin_model = se_logbin), 4)
```

```
## by_hand_model      vcov_model by_hand_robust
##      0.0795          0.0795      0.0520
##      vcovHC_HC0    logbin_model
##      0.0520          0.0501
```

The by-hand numbers match `vcov()` and `vcovHC()` to four decimals. The model-based standard error for the quitting coefficient is 0.0795, which is about sixty percent larger than the 0.0501 from the log-binomial model of Week 6, because the Poisson variance function μ is larger than the binary outcome's true variance $\mu(1 - \mu)$ at every fitted value. The robust standard error, 0.052, ignores the variance function altogether and uses the observed residuals instead, which brings it back to within four percent of the binomial's.

Note that the coefficients from this model are untouched by any of this. The robust variance estimator changes what we say about a coefficient's precision, not what its value.

3.2 Cluster-Robust Standard Errors and the Link to GEE

We have so far ignored the design of the BMI trial: twenty people in ten practices, two per practice, with treatment assigned by practice. People in the same practice are likely to be more alike than people in different practices, so their residuals are correlated, and the independence that both the model-based and the HC estimators take for granted here fails. The solution to this is to keep the same sandwich wstructure, but make one more change to the “meat”.

Instead of a single squared residual per person on the diagonal of Σ , the residuals are **summed within each cluster**, so that the meat becomes a sum over clusters rather than over people:

$$X^\top \Sigma X = \sum_{g=1}^G \left(X_g^\top \hat{u}_g \right) \left(X_g^\top \hat{u}_g \right)^\top$$

where X_g and $\hat{u}_g$ are the rows of the design matrix and the residuals belonging to cluster g . Equivalently, Σ gains off-diagonal entries for every pair of people in the same practice, the products of their residuals, and stays zero between practices. This is the cluster-robust sandwich of [Liang and Zeger \(1986\)](#), which

we can compute by hand and compare to the `sandwich` package's `vcovCL()` (Zeileis et al., 2020):

```
u <- as.vector(y - X %*% beta_hat)
G <- length(unique(bmi_data$practice))

# one score vector per practice, then the sum of their
# outer products
rows_by_practice <- split(1:nrow(X), bmi_data$practice)

meat_cl <- Reduce(`+`, lapply(rows_by_practice,
  function(idx) {
    s_g <- t(X[idx, , drop = FALSE]) %*% u[idx]
    s_g %*% t(s_g)
  }))

bread <- solve(t(X)%*%X)

var_beta_cl <- bread %*% meat_cl %*% bread

var_beta_cl
```

```
##           [,1]      [,2]
## [1,]  5.40028 -5.400280
## [2,] -5.40028  6.123202
```

```
# vcovCL multiplies by G/(G - 1) by default;
# HC1 adds (N - 1)/(N - p) on top
var_beta_cl * G/(G - 1)
```

```
##           [,1]      [,2]
## [1,]  6.000311 -6.000311
## [2,] -6.000311  6.803558
```

```
vcovCL(mod, cluster = bmi_data$practice, type = "HCO")
```

```
##              (Intercept) treatment
## (Intercept)    6.000311 -6.000311
## treatment      -6.000311  6.803558
```

```
vcovCL(mod, cluster = bmi_data$practice, type = "HC1")
```

```
##              (Intercept) treatment
## (Intercept)    6.333662 -6.333662
## treatment      -6.333662  7.181533
```

The by-hand cluster sandwich matches `vcovCL()` once its default finite-cluster correction, $G/(G - 1)$, is applied, and the HC1 version adds the $(N - 1)/(N - p)$ factor on top. The standard error of the treatment coefficient grows from 1.9 under the model-based estimator to 2.68 under the cluster-robust HC1 estimator, because two patients from the same practice carry less independent information than two patients from different practices. These are the two standard errors reported in the article by [Mansournia et al. \(2021\)](#), whose small-sample adjusted cluster-robust standard error is this HC1 version. One caution is that the cluster-robust estimator relies on the number of clusters being large, not the number of people. With ten practices it is a rough guide at best ([Cameron and Miller, 2015](#)).

This is also the point of contact with generalized estimating equations. A GEE with an “independence” working correlation solves exactly the score equations of the GLM from Week 6, so it returns the same coefficients, and its “robust” or “empirical” variance is the cluster sandwich above, without the finite-cluster correction:

```
gee_ind <- geeglm(BMI ~ treatment, id = practice,
                  data = bmi_data, family = gaussian,
                  corstr = "independence")

summary(gee_ind)$coefficients[, 1:2]
```

```
##              Estimate Std.err
## (Intercept)    28.39  2.32385
## treatment      0.42  2.47451
```

The coefficient is the OLS coefficient and the standard error is the square root of the unadjusted by-hand cluster variance printed first above. Other working correlation structures (exchangeable, autoregressive) change the weights inside the score equations, and therefore the coefficients slightly, but the variance reported is still a sandwich with the cluster as the unit. GEE, in other words, is the GLM machinery of Week 6 with the cluster-robust sandwich attached.

4 The Bootstrap

Epidemiologists rely almost exclusively on statistical models to estimate exposure effects. After obtaining a point estimate, standard practice is to quantify its statistical uncertainty. This quantification is usually accomplished via standard errors and/or confidence interval estimates. A confidence interval is meant to provide a range of parameter values compatible with the data, in the sense that intervals constructed the same way would contain the true value a stated proportion of the time under repeated random sampling and/or exposure allocation (i.e., due only to random variation, and not systematic bias). There are several ways to estimate confidence intervals of an estimate. The most common technique is to use maximum likelihood theory, which provides an estimate of the standard error of the parameter of interest directly from the likelihood function (known as the observed Fisher information) (Cole et al., 2013; Pawitan, 2001; Therneau and Grambsch, 2000), which can be used to obtain confidence intervals. In a linear modeling context, this “observed Fisher information” approach is equivalent to the model-based variance equation we observed in the first section:

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

with $\Sigma = s^2 I$. Ninety-five percent confidence intervals are computed by assuming the estimate follows a normal distribution, under which 95% of the mass of the estimate’s distribution falls within $1.96 \times SE(\hat{\beta})$ units of the point estimate (Wackerly et al., 2002). This approach is referred to as the Wald

equation or normal-interval confidence intervals.

This approach is based on asymptotic approximations that assume the sample size is sufficiently large.

There are generally three situations in which an alternative is preferred:

- 1) when an analytic expression for the standard error or confidence interval of an estimator is unknown;
- 2) when such an expression exists, but is too complex to implement manually with standard software;
- 3) or when the sample size is too small to allow large-sample (asymptotic) approximations to be reliably invoked.

A well-known alternative to variance estimation in such situations is the bootstrap ([Efron, 1979](#)).

Several bootstrap estimators exist ([Efron and Tibshirani, 1993](#)), however epidemiologists often rely on a few simple versions, namely the normal interval (Wald) bootstrap, and the percentile bootstrap.

Here, we'll illustrate how to use bootstrap confidence interval estimators. We'll first provide a brief conceptual overview of the bootstrap, focusing on three particular bootstrap estimators. We then show how the bootstrap can be used to obtain measures of uncertainty for the association between quitting smoking and weight gain in the NHEFS data of Weeks 5 and 6.

4.1 The Bootstrap Principle

Let's revisit the super-population construct we introduced at the start of this lecture. This time, we'll show the basic idea behind the bootstrap. This concept is often depicted using the following figure, from [Efron and Tibshirani \(1993\)](#):

The bootstrap is a technique that employs the Monte Carlo method and the substitution (or plug-in) principle to obtain standard error or confidence interval estimates ([Efron, 2003](#)). The Monte Carlo method is a general approach for solving a broad class of problems using computation to generate random numbers ([Metropolis and Ulam, 1949](#)).

This method is used for both **parametric** and **non-parametric** bootstrap estimators.

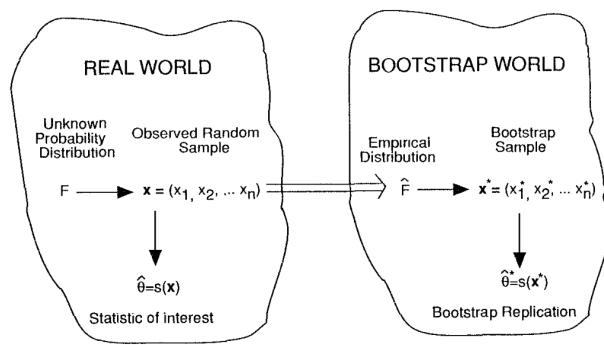


Figure 3: Illustration of the core principle behind the bootstrap. In the real world, an unknown distribution F produces the observed sample, from which the statistic of interest is computed. In the bootstrap world, the empirical distribution of the observed sample plays the role of F : resampling from it produces bootstrap samples, from which bootstrap replicates of the statistic are computed.

For the parametric bootstrap, the Monte Carlo method is used to generate residuals from a statistical model defining the relation between the exposure, confounders, and outcome of interest. The estimated parameters from this model are used, along with the exposure and confounder data, and the residuals generated from the Monte Carlo process, to generate a bootstrapped outcome. This process is repeated B times, giving B datasets with a bootstrapped outcome, and the original exposure and confounder data. One then fits a separate model to each of these B datasets to obtain a distribution of bootstrapped parameter estimates that can be used to quantify the uncertainty around the original estimates. In this setting, the bootstrapping process is “parametric” in that a parametric (e.g., logistic regression) model is used to generate the bootstrapped outcome data. Violations of the model’s assumptions can lead to biased estimates of statistical uncertainty for the original parameter estimates (Carpenter and Bithell, 2000).

The non-parametric bootstrap uses the random numbers generated from the Monte Carlo process to select random samples with replacement from the original data. The number of bootstrap samples chosen by the researcher is limited only by the available computing power. For each bootstrap sample, one can obtain a “bootstrap replicate” of the parameter of interest. With a large enough number of bootstrap replicates, one can use the distribution of bootstrap estimates to obtain information on the degree of uncertainty associated with the point estimate of interest. In effect, one can substitute (or plug-in) the empirical distribution of the estimates from each bootstrap resample for the unknown distribution of the point estimate. This empirical distribution can be used to estimate any feature (such as the standard error or

percentiles) of the unknown distribution of the parameter estimate.

It is often stated that this version of the bootstrap is non-parametric in that the data are not assumed to follow a specified parametric model (Carpenter and Bithell, 2000). While true, this does not imply that the non-parametric bootstrap will work equally well when the model used to generate the parameter estimate of interest is itself nonparametric. Indeed, this is the result of the fact that both the parametric and nonparametric bootstrap estimators require that the underlying estimation model meets certain regularity conditions, which are not guaranteed to hold when nonparametric methods are used.

4.2 Example Data

To demonstrate implementation, we return to the NHEFS data of Weeks 5 and 6 (Hernán and Robins, 2020), which were loaded in the robust standard error section above. Briefly, the NHEFS enrolled adults aged 25 to 74 who smoked cigarettes at a baseline visit in 1971, and measured them again in 1982. As in Weeks 5 and 6, the outcome is an indicator of weight change above the median between the two visits, the exposure is quitting smoking between them, and the covariates are age, sex, race, exercise, marital status (with the separated pooled with the divorced, and the one person of unknown status joining the married), and smoking intensity. The analytic sample holds the 1,566 people with a weight measurement at both visits, 403 of whom quit.

To estimate the association between quitting smoking and above-median weight gain, we fit the adjusted logistic regression model of Week 6, regressing the weight gain indicator against the quitting indicator and the covariates just listed.

4.3 Example Analysis

For clarity, we refer to these data collectively as $\mathcal{X}$. This set consists of the indicator of above-median weight gain (denoted Y), the quitting indicator (denoted T), and the vector of covariates (collectively denoted C). Our regression model can thus be defined as:

$$\text{logit } P(Y = 1 \mid T, C) = \beta_0 + \theta T + \beta_C C \quad (1)$$

Here $\hat{\theta}$ is the adjusted log odds ratio for quitting, the same coefficient Week

6 estimated. On the descriptive roadmap of Week 4 it is a conditional association; reading it as a causal effect would require the identification assumptions discussed there, and nothing about how we compute its confidence interval changes that.

Confidence intervals (CIs) for θ may be obtained using the inverse of the Fisher information, which would entail the standard practice of computing the values that result from ± 1.96 times the model-based standard error. However, here we proceed with obtaining CIs via the bootstrap.

Drawing a simple random sample (with replacement) of $N = 1,566$ individuals from $\mathcal{X}$, we obtain a bootstrap sample of our observed data that we denote $\mathcal{X}^*$. This bootstrap re-sample contains the same number of individuals as in the original sample ($N = 1,566$), except that some individuals are selected more than once, while others are not selected. We then re-fit the above logistic model to $\mathcal{X}^*$ to obtain a bootstrap replicate $\hat{\theta}^*$. Repeating the sampling procedure and re-fitting the model B times, we obtain a sample of bootstrap replicates $\{\hat{\theta}_1^*, \hat{\theta}_2^*, \dots, \hat{\theta}_B^*\}$. This sample of replicates allows us to estimate any feature of the distribution of our point estimate, such as its standard error or percentiles of its distribution.

While many different types of nonparametric bootstrap confidence interval estimators exist, the above features comprise the central characteristics of the approach. Different estimators are distinguished by what information is extracted (and how) from the sample of bootstrap replicates $\{\hat{\theta}_1^*, \hat{\theta}_2^*, \dots, \hat{\theta}_B^*\}$. Three bootstrap confidence interval estimators are arguably best suited to epidemiologic research due to their relative ease of implementation, established theoretical properties, and robustness to violations of a varying range of assumptions (Greenland, 2004). These are the Wald (or normal-interval estimator), percentile, and Bias-Corrected and Accelerated (BC_a) bootstrap estimators.

4.4 Normal Interval Bootstrap

The Wald, or normal interval, bootstrap estimator is one of the simplest of bootstrap confidence interval estimators. The approach requires the assumption that the parameter estimator $\hat{\theta}$ is normally distributed.

After obtaining a sample of bootstrap estimates as outlined above, one can implement the method by simply estimating the standard deviation of this

sample to obtain an estimate of the standard error of the point estimate. By assuming that $\hat{\theta}$ follows a normal distribution, one can then add and subtract $1.96 \times SE(\hat{\theta})$ from the point estimate to obtain upper and lower limits, respectively. For the example illustrated with Model 1, 95% Wald-type bootstrap confidence intervals for the estimated odds ratio $\exp(\hat{\theta})$ can be obtained as $\exp\{\hat{\theta} \pm 1.96 \times SD(\hat{\theta}^*)\}$, where $SD(\hat{\theta}^*)$ is the standard deviation of the bootstrap replicates. The standard deviation of the distribution of bootstrap replicates $SD(\hat{\theta}^*)$ is an estimate of the standard error of the point estimate $SE(\hat{\theta})$ (Altman and Bland, 2005).

Note that this normal-interval bootstrap is NOT transformation respecting. This means that if we are trying to construct confidence intervals for a (e.g.) risk ratio, we cannot transform the log risk ratio to the risk ratio scale, then take the standard deviation, then construct CIs. Instead, we must construct CIs on the log-scale, and then transform all the results (UCL, LCL, estimate). The normal-interval bootstrap is also NOT range respecting. This means that it is possible to estimate upper or lower bounds that are beyond the possible range of the estimator (e.g., negative risk ratios).

The assumption that $\hat{\theta}$ follows a normal distribution is only valid in reasonably large samples and under other conditions. As a consequence, Wald-type bootstrap confidence intervals can have a less than nominal coverage probability, particularly in small samples (Efron and Tibshirani, 1993; DiCiccio and Efron, 1996). Finally, Efron suggests that between 50 and 200 replicates is sufficient to obtain a good estimate of $SE(\hat{\theta})$ (Efron and Tibshirani, 1993). However, with increases in computing power over the last 30 years, a minimum of 500 replicates is usually expected in the scientific literature.

4.5 Percentile Bootstrap

The percentile bootstrap estimator is as simple to implement as the Wald estimator, but **does not require the assumption that $\hat{\theta}$ follows a normal distribution**. To obtain two-sided percentile bootstrap confidence intervals, one simply selects the bootstrap replicate (i.e., the estimate based on bootstrap resample) corresponding to the $100 \times \alpha/2$ and $100 \times (1 - \alpha/2)$ percentile of the distribution of bootstrap replicates. For example, with 2,000 bootstrap replicates sorted from smallest to largest, $\{\hat{\theta}_{(1)}^*, \hat{\theta}_{(2)}^*, \dots, \hat{\theta}_{(2,000)}^*\}$, and a nominal coverage of 95%, the 2.5th and 97.5th percentile points representing

the lower and upper confidence limits would correspond to $\hat{\theta}_{(50)}^*$ and $\hat{\theta}_{(1,950)}^*$, respectively.

The percentile method is both *transformation* and *range respecting*: for example, percentile confidence interval estimates can be obtained on either the log-scale and transformed to the exponential scale or vice versa (Efron and Tibshirani, 1993, p. 175). Moreover, they respect the boundedness of the estimator in that they will not provide confidence interval estimates that fall outside of the allowable range of the parameter estimate. Percentile intervals (when obtained using the non-parametric bootstrap) are completely non-parametric. However, because they take the bootstrap distribution at face value, with whatever bias and skewness it carries, their coverage probability is usually less than nominal, that is, anti-conservative (DiCiccio and Efron, 1996; Greenland, 2004; Efron and Tibshirani, 1993; Carpenter and Bithell, 2000). Additionally, because percentile-based methods require estimates of the tails of the distribution of bootstrap replicates, more bootstrap resamples are required. Although the specific number may depend on the scenario, 1,000 to 2,000 re-samples is often seen in practice, and more is usually better.

4.6 Bias-Corrected and Accelerated Intervals

Early recognition of the poor coverage probabilities of the Wald-type and percentile bootstrap confidence intervals led to two modifications of the percentile method (DiCiccio and Efron, 1996). The resulting “bias-corrected and accelerated” confidence intervals are meant to improve the performance of this percentile confidence interval estimator. The BC_a confidence interval estimator is a percentile-based estimator in that the confidence interval end-points selected are percentiles of the distribution of bootstrap replicates. This interval estimator is also transformation and range respecting. It differs from the standard percentile method in that the percentiles corresponding to the upper and lower interval estimates are chosen as a function of a bias correction factor and an acceleration factor that are determined by the data.

The bias-correction factor is meant to account for the discrepancy between the median of the sample of bootstrap replicates and the point estimate. This factor can be calculated as a function of the proportion of bootstrap replicates less than the original point estimate:

$$\hat{z}_0 = \Phi^{-1} \left(\frac{1}{B} \sum_{b=1}^B I(\hat{\theta}_b^* < \hat{\theta}) \right)$$

Note that the $\Phi^{-1}(\bullet)$ function is the inverse of the normal CDF, and returns a z-statistic for a given quantile from the normal distribution (e.g., `qnorm()` in R), and $I(\bullet)$ is an indicator function that equals one when its argument is true. Note that if exactly 50% of the bootstrap estimates $\hat{\theta}_b^*$ are less than the estimate obtained in the original data ($\hat{\theta}$), then the bootstrap estimator is said to be median unbiased, and the $\hat{z}_0$ value will be zero.

The acceleration factor is meant to compensate for possible heterogeneity in the standard error of the estimator as a function of the true parameter value, and can be calculated using the jackknife procedure (Tukey, 1958):

$$\hat{a} = \frac{\sum_i (\bar{\theta}^{(-i)} - \hat{\theta}^{(-i)})^3}{6 \left[\sum_i (\bar{\theta}^{(-i)} - \hat{\theta}^{(-i)})^2 \right]^{3/2}}$$

where $\hat{\theta}^{(-i)}$ is the point estimate in the original data with the i^{th} individual removed, and where $\bar{\theta}^{(-i)}$ denotes the average of these N leave-one-out estimates.

The bias correction and acceleration factors are then used to select altered percentiles of the distribution of bootstrap replicates. The BC_a bounds are defined as $[\hat{\theta}_{(\alpha_1)}^*, \hat{\theta}_{(\alpha_2)}^*]$, where:

$$\alpha_1 = \Phi \left(\hat{z}_0 + \frac{\hat{z}_0 + z_{(\alpha/2)}}{1 - \hat{a}[\hat{z}_0 + z_{(\alpha/2)}]} \right)$$

and

$$\alpha_2 = \Phi \left(\hat{z}_0 + \frac{\hat{z}_0 + z_{(1-\alpha/2)}}{1 - \hat{a}[\hat{z}_0 + z_{(1-\alpha/2)}]} \right)$$

When $\hat{z}_0 = 0$ and $\hat{a} = 0$, these reduce to $\Phi(z_{(\alpha/2)}) = \alpha/2$ and $\Phi(z_{(1-\alpha/2)}) = 1 - \alpha/2$, and the BC_a interval is the ordinary percentile interval.

4.7 Bootstrap Confidence Intervals in the NHEFS Data

We compared the normal interval, percentile, and bias corrected and accelerated bootstrap CI estimators in the NHEFS data to obtain uncertainty estimates for the association defined in Model 1. In the R programming language, these CI estimators can be easily obtained using the `boot` package (Davison and

Hinkley, 1997).

First, we fit the model. The data and the formula `f_adj` are the ones loaded in the robust standard error section above:

```
dim(nhefs)

## [1] 1566    65

mod_mu <- glm(f_adj, data = nehs, family = binomial("logit"))

round(summary(mod_mu)$coefficients, 4)

##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)      1.5071     0.2636  5.7170  0.0000
## qsmk              0.6743     0.1240  5.4399  0.0000
## age             -0.0367     0.0048 -7.6945  0.0000
## sex              0.0496     0.1114  0.4451  0.6562
## race            -0.1918     0.1625 -1.1804  0.2378
## factor(exercise)1 -0.0850     0.1456 -0.5836  0.5595
## factor(exercise)2 -0.2315     0.1511 -1.5327  0.1254
## factor(marital)3   0.2606     0.2393  1.0891  0.2761
## factor(marital)4  -0.0114     0.2240 -0.0509  0.9594
## factor(marital)5  -0.0602     0.1835 -0.3281  0.7428
## smokeintensity    0.0018     0.0047  0.3831  0.7016

# log-odds ratio for quitting
summary(mod_mu)$coefficients[2,1]

## [1] 0.6743473
```

Now we obtain bootstrap confidence intervals for this log-odds ratio of interest. The `boot()` function needs a function of two arguments, the data and a vector of row indices, that returns the statistic computed on the resampled rows. We start with the normal-interval bootstrap:

```

boot_func <- function(data, index){

  boot_dat <- data[index,]

  mod_mu <- glm(f_adj, data = boot_dat,
               family = binomial("logit"))

  res <- summary(mod_mu)$coefficients[2,1]

  return(res)

}

set.seed(123)
boot_res <- boot(data = nhefs, statistic = boot_func,
                R = 500)

res_ci_norm <- boot.ci(boot_res, type = "norm")

res_ci_norm

```

```

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 500 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot_res, type = "norm")
##
## Intervals :
## Level      Normal
## 95%      ( 0.4388,  0.9108 )
## Calculations and Intervals on Original Scale

```

```

round(c(model_based_SE = summary(mod_mu)$coefficients[2,2],
      bootstrap_SD    = sd(boot_res$t)), 4)

```

```
## model_based_SE    bootstrap_SD
##           0.1240          0.1204
```

The standard deviation of the 500 replicates is close to the model-based standard error, as it should be: the logistic model's variance function is the right one for a binary outcome, so the two routes to the standard error agree. One detail of the printout is worth knowing. The normal interval that `boot.ci()` reports is not centered on the point estimate. It first subtracts the bootstrap estimate of bias, the mean of the replicates minus the original estimate, and then adds and subtracts $1.96 \times SD(\hat{\theta}^*)$ (Davison and Hinkley, 1997). Here the bias is small, so the interval is close to the one computed by hand:

```
theta_hat <- boot_res$t0
se_boot   <- sd(boot_res$t)
bias_boot <- mean(boot_res$t) - theta_hat

# textbook normal interval, centered on the estimate
round(theta_hat + c(-1, 1) * 1.96 * se_boot, 4)
```

```
## [1] 0.4384 0.9103
```

```
# boot.ci(): shift by the bias, then add the margin
round(theta_hat - bias_boot + c(-1, 1) * 1.96 * se_boot, 4)
```

```
## [1] 0.4388 0.9108
```

Next, we obtain percentile intervals. Note the larger R value:

```
set.seed(123)
boot_res <- boot(data = nhefs, statistic = boot_func,
                 R = 1000)

res_ci_perc <- boot.ci(boot_res, type = "perc")

res_ci_perc
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot_res, type = "perc")
##
## Intervals :
## Level      Percentile
## 95%      ( 0.4417,  0.9307 )
## Calculations and Intervals on Original Scale
```

Next, we obtain bias-corrected and accelerated intervals, with the same R value:

```
res_ci_bca <- boot.ci(boot_res, type = "bca")

res_ci_bca
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot_res, type = "bca")
##
## Intervals :
## Level      BCa
## 95%      ( 0.4440,  0.9311 )
## Calculations and Intervals on Original Scale
```

One implementation detail is worth knowing about. To compute the acceleration factor, `boot.ci()` needs the influence of each observation on the estimate. When the number of resamples R is at least as large as the sample size N , the package estimates these by regressing the bootstrap replicates on how many times each observation appeared in each resample. When $R < N$, as above, it switches to the jackknife, which refits the model N times (here, 1,566 times), which is why the BC_a call takes noticeably longer than the percentile call. An earlier version of these notes showed this call failing when $R < N$

with the message “estimated adjustment ‘a’ is NA”. With the version of `boot` used to compile these notes (1.3.32), it runs. If you do meet that error, the remedy is to set $R \geq N$, or to pass jackknife influence values to `boot.ci()` through its `L` argument. Here is the same interval with $R = 2000$:

```
set.seed(123)
boot_res <- boot(data = nhefs, statistic = boot_func,
                R = 2000)

res_ci_bca <- boot.ci(boot_res, type = "bca")

res_ci_bca
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 2000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = boot_res, type = "bca")
##
## Intervals :
## Level      BCa
## 95%      ( 0.4430, 0.9495 )
## Calculations and Intervals on Original Scale
```

The choice of a given bootstrap estimator will often depend on context. For a computationally intensive estimator, the only practical choice may be the normal interval estimator, due to the need for fewer resamples than the percentile or BC_a methods. However, when computationally feasible, these latter estimators should be considered. Between the two, the percentile method is simpler to implement in that it only requires selecting percentile values of the bootstrap distribution of estimates. However, Greenland emphasizes the point that the naive percentile method is conceptually backwards: uncorrected bootstrap estimates incorporate a small sample bias and skewness of the original estimators and add bias and skewness in the errors (Efron and Tibshirani, 1993; Greenland, 2004; Carpenter and Bithell, 2000). The BC_a bootstrap estimator was constructed specifically to overcome these problems.

5 Conclusion

Estimator	What fills the meat	Protects against	Where it breaks
Model-based	the family's variance function at each fitted mean	sampling error only	Poisson for a binary outcome, clustering
Robust (HC)	each person's squared residual	wrong variance function, heteroscedasticity	few observations (use HC3), clustering
Cluster-robust	residual cross-products summed within cluster	correlated residuals within clusters	few clusters
Bootstrap	nothing: the sampling is redone by resampling	no formula, or a formula too hard to trust	nonregular estimators, dependent data

Four routes to a standard error. Every one of the first three is the same sandwich, $(X^T W X)^{-1} X^T \Sigma X (X^T W X)^{-1}$, with a different rule for what goes into Σ . The bootstrap replaces the formula with repetition. For dependent data, the resampling unit must be the cluster rather than the person.

This lecture took the standard error apart in the same way the previous three took the point estimate apart. Every model-based standard error is a sandwich whose meat is a formula handed down by the family. The robust estimator keeps the bread and fills the meat with what was actually observed, the squared residuals, which is why it repairs a wrong variance function (the Poisson model of a binary outcome in Week 6, the constant variance of a linear model fit to heteroscedastic data here) without touching the coefficient. The cluster-robust estimator sums those residuals within cluster first, which is also what generalized estimating equations report. The bootstrap drops the formula altogether and re-runs the sampling.

One caution closes the loop with Week 4. A robust standard error is robust to the variance being wrong, not to the mean model being wrong. If the conditional mean is misspecified, the coefficient converges to the best approximation the model can offer, the projection we discussed in Week 4, and the robust standard error is the honest standard error for that projection, which is not the same thing as the honest standard error for the quantity we had in mind (Freedman, 2006; King and Roberts, 2015). A large gap between the model-based and robust standard errors is therefore worth reading as a diagnostic of the model, and not only as a correction to its uncertainty.

References

- Douglas G Altman and J Martin Bland. Standard deviations and standard errors. *BMJ*, 331(7521):903, 2005.
- Laura B Balzer, Maya L Petersen, and Mark J van der Laan. Targeted estimation and inference for the sample average treatment effect in trials with and without pair-matching. *Stat Med*, 35(21):3717–3732, 2016.
- A Colin Cameron and Douglas L Miller. A practitioner’s guide to cluster-robust inference. *J Hum Resour*, 50(2):317–372, 2015.
- Michael J Campbell. *Statistics at square two: understanding modern statistical applications in medicine*. Blackwell, 2006.
- James Carpenter and John Bithell. Bootstrap confidence intervals: when, which, what? a practical guide for medical statisticians. *Stat Med*, 19(9): 1141–1164, 2000.
- Stephen R Cole, Haitao Chu, and Sander Greenland. Maximum likelihood, profile likelihood, and penalized likelihood: A primer. *Am J Epidemiol*, 179(2): 252–260, 2013.
- AC Davison and DV Hinkley. *Bootstrap methods and their application*. Cambridge University Press, Cambridge, NY, 1997.
- T.J. DiCiccio and B. Efron. Bootstrap confidence intervals. *Stat Sci*, 11(3): 189–212, 1996.
- B. Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979.
- Bradley Efron. Second thoughts on the bootstrap. *Stat Sci*, 18(2):135–140, 2003.
- Bradley Efron and Robert Tibshirani. *Introduction to the Bootstrap*. Chapman & Hall/CRC, Boca Raton, FL, 1993.
- Friedhelm Eicker. Limit theorems for regressions with unequal and dependent errors. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 59–82. University of California Press, 1967.

- David A Freedman. On the so-called “Huber sandwich estimator” and “robust standard errors”. *Am Stat*, 60(4):299–302, 2006.
- Sander Greenland. Interval estimation by simulation as an alternative to and extension of confidence intervals. *Int J Epidemiol*, 33(6):1389–1397, 2004.
- M. A. Hernán and JM Robins. *Causal Inference*. Chapman & Hall/CRC, Boca Raton, FL, 2020.
- Peter J. Huber. The behavior of maximum likelihood estimates under non-standard conditions. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 221–233, Berkeley, CA, 1967. University of California Press.
- Gary King and Margaret E Roberts. How robust standard errors expose methodological problems they do not fix, and what to do about it. *Polit Anal*, 23(2): 159–179, 2015.
- Kung-Yee Liang Liang and Scott L. Zeger. Longitudinal data analysis using generalized linear models. *Biometrika*, 73(1):13–22, 1986.
- J Scott Long and Laurie H Ervin. Using heteroscedasticity consistent standard errors in the linear regression model. *Am Stat*, 54(3):217–224, 2000.
- James G MacKinnon and Halbert White. Some heteroskedasticity-consistent covariance matrix estimators with improved finite sample properties. *J Econom*, 29(3):305–325, 1985.
- Mohammad Ali Mansournia, Maryam Nazemipour, Ashley I Naimi, Gary S Collins, and Michael J Campbell. Reflection on modern methods: demystifying robust standard errors for epidemiologists. *Int J Epidemiol*, 50(1): 346–351, 2021.
- N Metropolis and S Ulam. The Monte Carlo method. *J Am Stat Assoc*, 44(247): 335–341, 1949.
- Yudi. Pawitan. *In all likelihood : statistical modelling and inference using likelihood*. Clarendon Press ; Oxford University Press, Oxford; New York, 2001.
- Terry M. Therneau and Patricia M. Grambsch. *Modeling survival data : extending the Cox model*. Springer, New York, 2000.

J.W. Tukey. Bias and confidence in not quite large samples. *Annals of Mathematical Statistics*, 29(2):614–, 1958.

Dennis D. Wackerly, William. Mendenhall, and Richard L. Scheaffer. *Mathematical statistics with applications*. Duxbury, Pacific Grove, CA, 2002.

Halbert White. A heteroskedasticity-consistent covariance matrix estimator and a direct test for heteroskedasticity. *Econometrica*, 48(4):817–838, 1980.

Achim Zeileis. Object-oriented computation of sandwich estimators. *J Stat Softw*, 16(9):1–16, 2006.

Achim Zeileis, Susanne Köll, and Nathaniel Graham. Various versatile variances: An object-oriented implementation of clustered covariances in R. *J Stat Softw*, 95(1):1–36, 2020.

Guangyong Zou. A modified poisson regression approach to prospective studies with binary data. *Am J Epidemiol*, 159(7):702–706, Apr 2004.