Practice Problems in R

Peter Sun

September 30-31, 2021

Contents

1 Practice 1: Generalized Boosted Regression and Propensity Score Weighting

1.1 Problem 1: Generalized Boosted Regression

1.1.1 Load Packages

```
library(tidyverse)
library(haven)
library(sjlabelled)
library(lmtest)
library(gbm)
library(modelr)
library(broom)
library(sandwich)
library(cobalt)
library(WeightIt)
library(Matching)
library(kableExtra)
select <- dplyr::select
```

1.1.2 Load and Randomly Shuffle Data

```
d <- read_dta("data/ldw_exper.dta") %>%
  haven::zap_formats() %>%
  sjlabelled::remove_all_labels() %>%
  as_tibble()
set.seed(1000)
d2 <- d %>%
  add_column(runif = runif(nrow(.))) %>%
  arrange(runif)
```

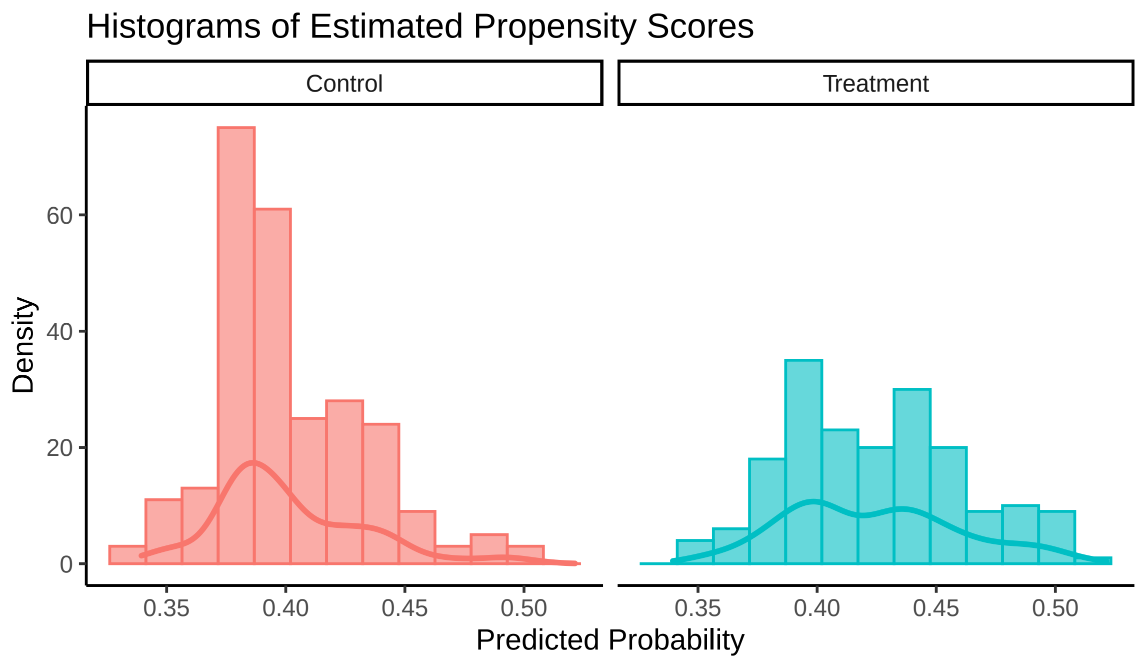
1.1.3 Generate Propensity Scores

```
set.seed(1000)
m1 <- gbm::gbm(formula = t ~ age + educ + black + hisp + married + re74 +
                         re75 + u74 + u75,
               data = d2,
               distribution = "bernoulli",
               n.trees = 1000,
               train.fraction = 0.8,
               interaction.depth = 4,
               shrinkage = 0.0005)

# Estimate Propensity Scores and Obtain Summary Statistics
d3 <- d2 %>%
  modelr::add_predictions(m1, var = "psb", type = "response")
summary(d3$psb)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.3394  0.3855  0.4006  0.4108  0.4346  0.5214
```

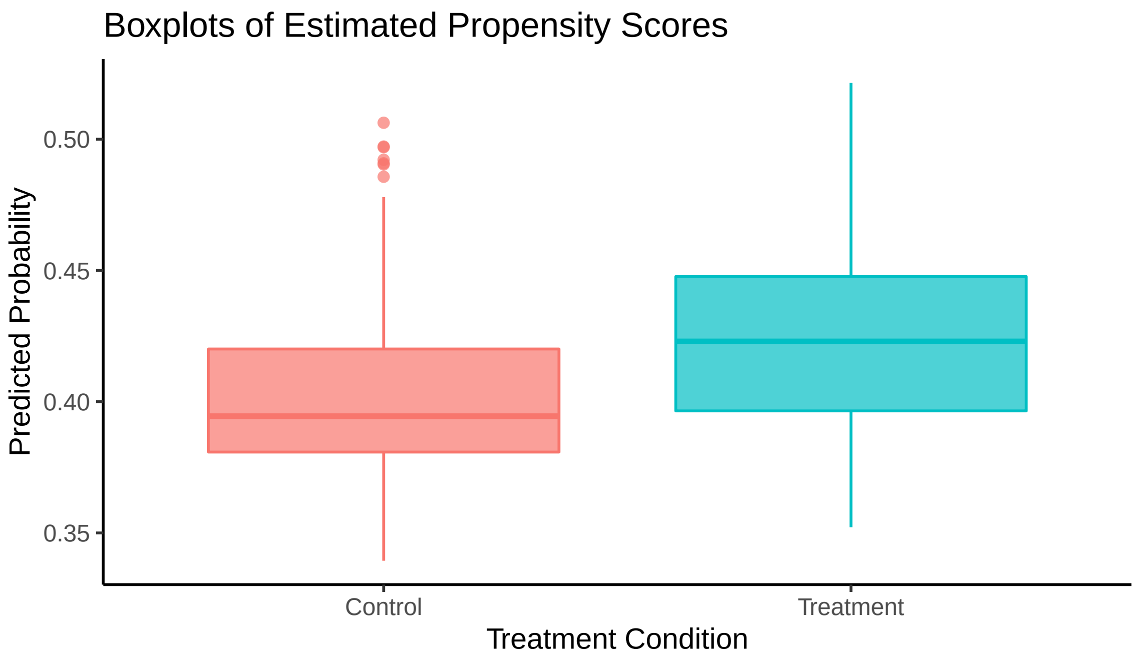
1.1.4 Histograms of Propensity Scores

```r
d3 %>%
  mutate(t = factor(t, labels = c("Control", "Treatment"))) %>%
  ggplot(aes(x = psb, color = t)) +
  theme_classic() +
  geom_histogram(aes(fill = t), alpha = 0.6, bins = 13) +
  geom_density(size = 1) +
  labs(x = "Predicted Probability", y = "Density",
       title = "Histograms of Estimated Propensity Scores") +
  theme(legend.position = "none") +
  facet_wrap(~ t)
```



1.1.5 Boxplots of Propensity Scores

```
d3 %>%
  mutate(t = factor(t, labels = c("Control", "Treatment"))) %>%
  ggplot(aes(x = t, y = psb, color = t, fill = t)) +
  theme_classic() +
  geom_boxplot(alpha = 0.7) +
  labs(x = "Treatment Condition",
       y = "Predicted Probability",
       title = "Boxplots of Estimated Propensity Scores") +
  theme(legend.position = "none")
```



1.2 Problem 2: Propensity Score Weighting

1.2.1 Estimate ATE and ATT Weights

```
d4 <- d3 %>%
  mutate(ate_w = ifelse(t == 0, 1/(1-psb), 1/psb),
         att_w = ifelse(t == 0, psb/(1-psb), 1))

# Import Stata-generated weights to replicate Stata results
stata_weights <- read_dta("data/ldw1.dta") %>%
  haven::zap_formats() %>%
  sjlabelled::remove_all_labels() %>%
  as_tibble() %>%
  mutate(stata_ate_w = ifelse(t == 0, 1/(1-psb), 1/psb),
         stata_att_w = ifelse(t == 0, psb/(1-psb), 1)) %>%
  select(id, stata_ate_w, stata_att_w)
d5 <- d4 %>%
  arrange(id) %>%
  left_join(stata_weights, by = "id")
```

1.2.2 Outcome Analysis with ATE and ATT Weights

```
# Define outcome formula
f = as.formula(re78 ~ t + age + educ + black + hisp + married + re74 + re75 +
           u74 + u75)

# Weighted OLS with R-Generated Propensity Scores
m2 <- lm(f, data = d5, weights = ate_w)
tidy(lmtest::coeftest(m2, vcov. = vcovHC(m2, "HC1"))) %>% filter(term == "t") # ATE
```

```
## # A tibble: 1 x 5
##   term  estimate std.error statistic p.value
##   <chr>    <dbl>     <dbl>     <dbl>   <dbl>
## 1 t         1.65     0.655      2.52  0.0122
```

```
m3 <- lm(f, data = d5, weights = att_w)
tidy(lmtest::coeftest(m3, vcov. = vcovHC(m3, "HC1"))) %>% filter(term == "t") # ATT
```

```
## # A tibble: 1 x 5
##   term  estimate std.error statistic p.value
##   <chr>    <dbl>     <dbl>     <dbl>   <dbl>
## 1 t         1.72     0.663      2.60 0.00965
```

```
# Weighted OLS with Stata-Generated Propensity Scores (Identical Results)
m2.stata <- lm(f, data = d5, weights = stata_ate_w)
tidy(lmtest::coeftest(m2.stata, vcov. = vcovHC(m2.stata, "HC1"))) %>%
  filter(term == "t") # ATE
```

```
## # A tibble: 1 x 5
##   term  estimate std.error statistic p.value
##   <chr>    <dbl>     <dbl>     <dbl>   <dbl>
## 1 t         1.63     0.656      2.48  0.0135
```

```
m3.stata <- lm(f, data = d5, weights = stata_att_w)
tidy(lmtest::coeftest(m3.stata, vcov. = vcovHC(m3.stata, "HC1"))) %>%
  filter(term == "t") # ATT
```

```
## # A tibble: 1 x 5
##   term  estimate std.error statistic p.value
##   <chr>    <dbl>     <dbl>     <dbl>   <dbl>
## 1 t         1.70     0.665      2.55  0.0111
```

1.2.3 Check Imbalance

See the Appendix for the custom function `robustse()` that is used to replicate the robust standard errors in Stata.

```r
# Function to Check Imbalance
check_bal <- function(var, weight, type) {
  if(type == "categorical") {
    m <- glm(as.formula(paste0(var, "~t")),
      family = quasibinomial,
      data = d5,
      weights = weight
    )
    m %>%
      tidy() %>%
      mutate(odds.ratio = exp(estimate), variable = var) %>%
      mutate(or.se = robustse(m, coef = "odd.ratio")[,2]) %>%
      mutate(statistic = robustse(m, coef = "odd.ratio")[,3]) %>%
      mutate(p.value = robustse(m, coef = "odd.ratio")[,4]) %>%
      select(variable, term, odds.ratio, or.se, statistic, p.value)
  } else if(type == "continuous") {
    m <- lm(as.formula(paste0(var, "~t")),
            data = d5,
            weights = weight)
    lmtest::coeftest(m, vcov. = vcovHC(m, "HC1")) %>%
      tidy() %>%
      add_column(var, .before = "term")
  }
}
format_bal <- function(df) {
  df %>%
    filter(term != "(Intercept)") %>%
    kbl(booktabs = T, digits = 7) %>%
    kable_styling(position = "center") %>%
    kable_styling(latex_options = c("striped", "HOLD_position"))
}
```

```
# Categorical Variables
cat_vars <- c("black", "hisp", "married", "u74", "u75")
format_bal(map_dfr(cat_vars, check_bal, d5$stata_ate_w, "categorical"))
```

variable	term	odds.ratio	or.se	statistic	p.value
black	t	1.0997119	0.2872935	0.3638291	0.7159856
hisp	t	0.5680117	0.2106900	-1.5248703	0.1272914
married	t	1.2388627	0.3163250	0.8388736	0.4015402
u74	t	0.8907275	0.1929677	-0.5341418	0.5932434
u75	t	0.7896914	0.1590241	-1.1725044	0.2409946

```
format_bal(map_dfr(cat_vars, check_bal, d5$stata_att_w, "categorical"))
```

variable	term	odds.ratio	or.se	statistic	p.value
black	t	1.0785012	0.2815619	0.2894740	0.7722186
hisp	t	0.5688118	0.2106633	-1.5234109	0.1276559
married	t	1.2497739	0.3193590	0.8725380	0.3829149
u74	t	0.8616888	0.1868079	-0.6866516	0.4923023
u75	t	0.7639094	0.1539378	-1.3364198	0.1814121

```
# Continuous Variables
cont_vars <- c("age", "educ", "re74", "re75")
format_bal(map_dfr(cont_vars, check_bal, d5$stata_ate_w, "continuous"))
```

var	term	estimate	std.error	statistic	p.value
age	t	0.5085618	0.6801859	0.7476806	0.4550495
educ	t	0.1629619	0.1762747	0.9244774	0.3557411
re74	t	-0.1742469	0.4893238	-0.3560973	0.7219372
re75	t	0.1288075	0.2977874	0.4325485	0.6655533

```
format_bal(map_dfr(cont_vars, check_bal, d5$stata_att_w, "continuous"))
```

var	term	estimate	std.error	statistic	p.value
age	t	0.5532783	0.6945403	0.7966107	0.4261038
educ	t	0.1874483	0.1813631	1.0335526	0.3019093
re74	t	-0.0963895	0.5091306	-0.1893218	0.8499273
re75	t	0.1832357	0.3088324	0.5933178	0.5532713

Similar results can be obtained using the R-generated propensity score weights:

```
# With R-generated weights
format_bal(map_dfr(cat_vars, check_bal, d5$ate_w, "categorical"))
format_bal(map_dfr(cat_vars, check_bal, d5$att_w, "categorical"))
format_bal(map_dfr(cont_vars, check_bal, d5$ate_w, "continuous"))
format_bal(map_dfr(cont_vars, check_bal, d5$att_w, "continuous"))
```

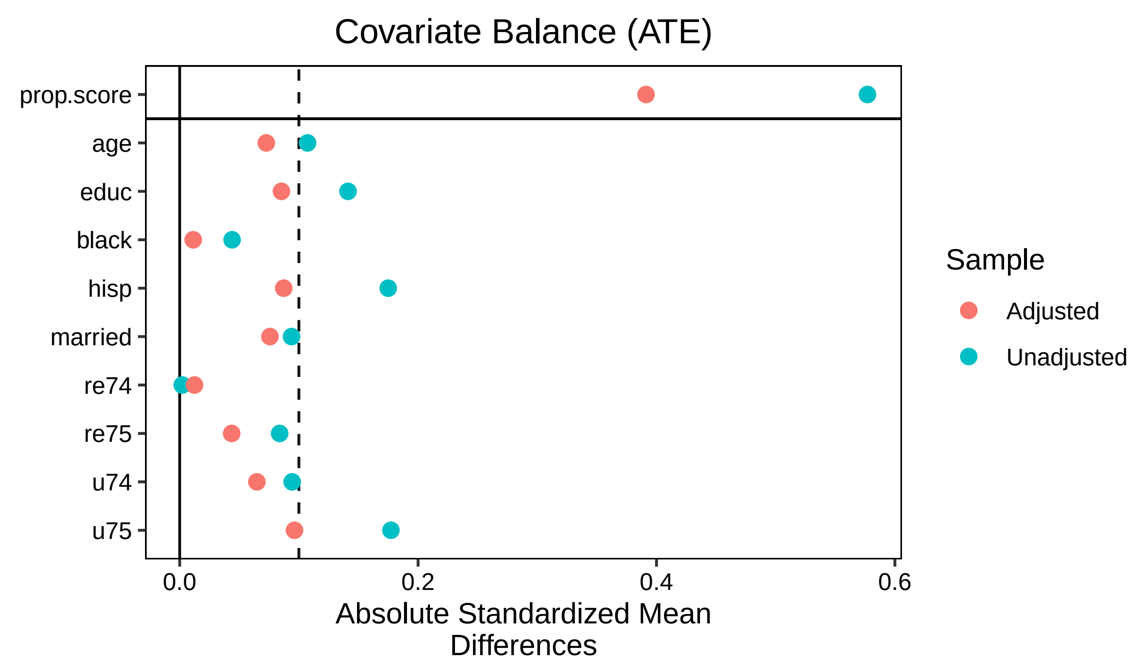
1.2.4 Alternative Solution with the WeightIt Package

Use GBM to estimate ATE and ATT:

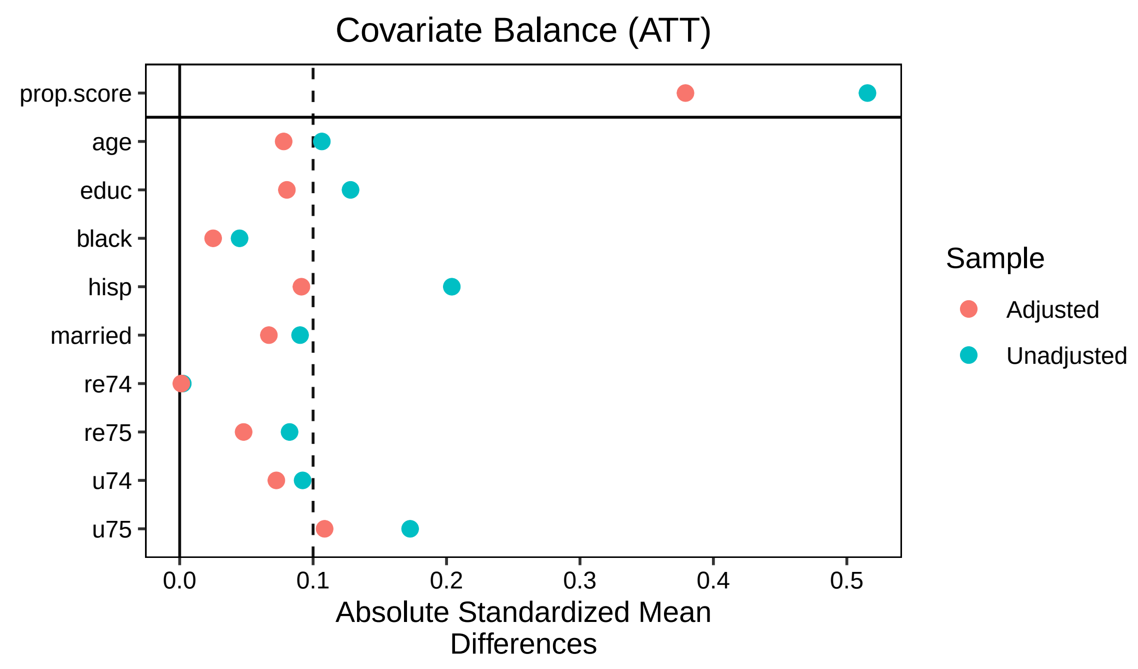
```
set.seed(1000)
w1.out <- WeightIt::weightit(
  formula = t ~ age + educ + black + hisp + married + re74 +
                 re75 + u74 + u75,
  data = d2,
  method = "gbm",
  distribution = "bernoulli",
  stop.method = "es.mean",
  n.trees = 1000,
  nTrain = 0.8 * nrow(d2),
  interaction.depth = 4,
  shrinkage = 0.0005,
  estimand = "ATE")

set.seed(1000)
w2.out <- WeightIt::weightit(
  formula = t ~ age + educ + black + hisp + married + re74 +
                 re75 + u74 + u75,
  data = d2,
  method = "gbm",
  distribution = "bernoulli",
  stop.method = "es.mean",
  n.trees = 1000,
  nTrain = 0.8 * nrow(d2),
  interaction.depth = 4,
  shrinkage = 0.0005,
  estimand = "ATT")
```

Assess balance with the cobalt package:

```
cobalt::love.plot(w1.out, thresholds = c(m = .1), binary = "std", abs = T) +
  labs(title = "Covariate Balance (ATE)")
```



```r
cobalt::love.plot(w2.out, thresholds = c(m = .1), binary = "std", abs = T) +
  labs(title = "Covariate Balance (ATT)")
```



For the outcome analysis, the ATE and ATT weights can be obtained with `w1.out$weights` (ATE) and `w2.out$weights` (ATT):

```r
m2.weightit <- lm(f, data = d2, weights = w1.out$weights)
tidy(lmtest::coeftest(m2.weightit, vcov. = vcovHC(m2.weightit, "HC1"))) %>%
  filter(term == "t")
```

```
## # A tibble: 1 x 5
##   term  estimate std.error statistic p.value
##   <chr>    <dbl>     <dbl>     <dbl>   <dbl>
## 1 t         1.57     0.646      2.43  0.0155
```

```r
m3.weightit <- lm(f, data = d2, weights = w2.out$weightit)
tidy(lmtest::coeftest(m3.weightit, vcov. = vcovHC(m3.weightit, "HC1"))) %>%
  filter(term == "t")
```

```
## # A tibble: 1 x 5
##   term  estimate std.error statistic p.value
##   <chr>    <dbl>     <dbl>     <dbl>   <dbl>
## 1 t         1.67     0.662      2.53  0.0119
```

2 Practice 2: Matching Estimators

2.1 Load Data

```
p2.d <- read_dta("data/prac2.dta") %>%
  haven::zap_formats() %>%
  sjlabelled::remove_all_labels() %>%
  as_tibble()
```

2.2 Breusch-Pagan Test for Heteroskedasticity

The homoscedasticity assumption is not valid (e.g., p-value of the test for `age97` is $< .05$), indicating that the conditional variance of the outcome variable was not constant across levels of child's age, therefore a robust estimation of variance is warranted.

```
p2.m0 <- lm(lwss97 ~ kuse + male + black + age97 + pcged97 + mratio96 + pcg_adc,
            data = p2.d)
get_bptest <- function(data, lm.model, var) {
  b <- lmtest::bptest(lm.model, as.formula(paste0("~", var)),
                      data = data, studentize = F)
  return(tibble(variable = var, statistic = b$statistic,
                df = b$parameter, p.value = b$p.value))
}
map_dfr(c("kuse", "male", "black", "age97", "pcged97", "mratio96", "pcg_adc"),
        get_bptest, data = p2.d, lm.model = p2.m0) %>%
  kbl(booktabs = T, digits = 2, linesep = "",
      caption = "Results of Breusch-Pagan Tests for Heteroskedasticity") %>%
  kable_styling(position = "center") %>%
  kable_styling(latex_options = c("striped", "HOLD_position"))
```

Table 1: Results of Breusch-Pagan Tests for Heteroskedasticity

variable	statistic	df	p.value
kuse	1.78	1	0.18
male	0.86	1	0.35
black	1.15	1	0.28
age97	8.55	1	0.00
pcged97	4.43	1	0.04
mratio96	6.85	1	0.01
pcg_adc	0.60	1	0.44

2.3 Matching Estimators

2.3.1 Define Outcome (Y), Treatment Index (Tr), and Variables to Match On (X)

```r
Y <- p2.d$lwss97
Tr <- p2.d$kuse
X <- select(p2.d, male, black, age97, pcged97, mratio96, pcg_adc)
```

2.3.2 Define Function for Matching

```r
get_match <- function(estimand, sample) {
  m <- Matching::Match(Y = Y, Tr = Tr, X = X, M = 4, BiasAdjust = T, Var.calc = 4,
              estimand = estimand, sample = sample)
  return(list(
    est = m$est[,1],
    se = m$se,
    t.stat = m$est[,1]/m$se,
    p = (1 - pnorm(abs(m$est[,1]/m$se))) * 2
  ))
}
```

2.3.3 Get All Estimators

```r
tribble(
  ~estimator, ~estimand, ~sample,
  "SATE", "ATE", T,
  "PATE", "ATE", F,
  "SATT", "ATT", T,
  "PATT", "ATT", F,
  "SATC", "ATC", T,
  "PATC", "ATC", F
) %>%
  rowwise() %>%
  mutate(match = list(get_match(estimand, sample))) %>%
  tidyr::unnest_wider(match) %>%
  select(-estimand, -sample) %>%
  kbl(booktabs = T, linesep = "") %>%
  kable_styling(position = "center") %>%
  kable_styling(latex_options = c("striped", "hold_position"))
```

estimator	est	se	t.stat	p
SATE	-5.448863	1.646936	-3.3084850	0.0009380
PATE	-5.448863	1.652232	-3.2978811	0.0009742
SATT	-1.277287	1.683284	-0.7588067	0.4479682
PATT	-1.277287	1.695820	-0.7531973	0.4513314
SATC	-7.016781	1.965677	-3.5696503	0.0003575
PATC	-7.016781	1.969424	-3.5628594	0.0003668

3 Appendix: Replicating Stata's Robust Standard Errors

Custom function by Jorge Cimentada that is used to replicate the robust standard errors in Stata:[1]

```r
robustse <- function(x, coef = c("logit", "odd.ratio", "probs")) {
  suppressMessages(suppressWarnings(library(lmtest)))
  suppressMessages(suppressWarnings(library(sandwich)))

  sandwich1 <- function(object, ...) {
    sandwich(object) *
      nobs(object) / (nobs(object) - 1)
  }
  # Function calculates SE's
  mod1 <- coeftest(x, vcov = sandwich1)
  # apply the function over the variance-covariance matrix

  if (coef == "logit") {
    return(mod1) # return logit with robust SE's
  } else if (coef == "odd.ratio") {
    mod1[, 1] <- exp(mod1[, 1]) # return odd ratios with robust SE's
    mod1[, 2] <- mod1[, 1] * mod1[, 2]
    return(mod1)
  } else {
    mod1[, 1] <- (mod1[, 1] / 4) # return probabilites with robust SE's
    mod1[, 2] <- mod1[, 2] / 4
    return(mod1)
  }
}
```

[1]https://cimentadaj.github.io/blog/2016-09-19-obtaining-robust-standard-errors-and-odds-ratios/obtaining-robust-standard-errors-and-odds-ratios-for-logistic-regression-in-r/