Code for QSS tidyverse Chapter 7: Uncertainty

Kosuke Imai and Nora Webb Williams

First Printing

Uncertainty

Estimation

Unbiasedness and Consistency

```
## simulation parameters
n <- 100 # sample size
mu0 <- 0 # mean of Y_i(0) [not treated]
sd0 <- 1 # standard deviation of Y_i(0)
mu1 <- 1 # mean of Y_i(1) [treated]
sd1 <- 1 # standard deviation of Y_i(1)

## generate a sample as a tibble
smpl <- tibble(id = seq_len(n),
               # Y if not treated
               Y0 = rnorm(n, mean = mu0, sd = sd0),
               # Y if treated
               Y1 = rnorm(n, mean = mu1, sd = sd1),
               # individual treatment effect
               tau = Y1 - Y0)

## true value of the sample average treatment effect
SATE <- smpl %>% select(tau) %>% summarize(SATE = mean(tau)) %>% pull()
SATE
```

```
## [1] 1.198005
```

```
sim_treat <- function(smpl) {
  n <- nrow(smpl)
  # indexes of obs receiving treatment (randomly assign to half)
  idx <- sample(seq_len(n), floor(nrow(smpl) / 2), replace = FALSE)
  # "treat" variable is 1 for those receiving treatment, else 0
  smpl[["treat"]] <- as.integer(seq_len(nrow(smpl)) %in% idx)
  smpl %>%
    # what outcome we observe for unit based on treatment assignment
    mutate(Y_obs = if_else(treat == 1, Y1, Y0)) %>%
    group_by(treat) %>%
    summarize(Y_obs = mean(Y_obs)) %>%
```

```r
    pivot_wider(names_from = treat,
                values_from = Y_obs) %>%
    rename(Y1_mean = `1`, Y0_mean = `0`) %>%
    mutate(diff_mean = Y1_mean - Y0_mean,
           est_error = diff_mean - SATE)
}
## show the results of the function on the data
## values will differ each time it is run
sim_treat(smpl)
```

```
## # A tibble: 1 x 4
##   Y0_mean Y1_mean diff_mean est_error
##     <dbl>   <dbl>     <dbl>     <dbl>
## 1 -0.0235    1.09      1.11   -0.0865
```

```r
## number of simulations
sims <- 500
## run the created function sims times
sate_sims <- map_df(seq_len(sims), ~ sim_treat(smpl))
## what is the distribution of the error?
summary(sate_sims$est_error)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -0.36538 -0.11269 -0.01706 -0.01248  0.09023  0.44552
```

```r
PATE <- mu1 - mu0
PATE
```

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

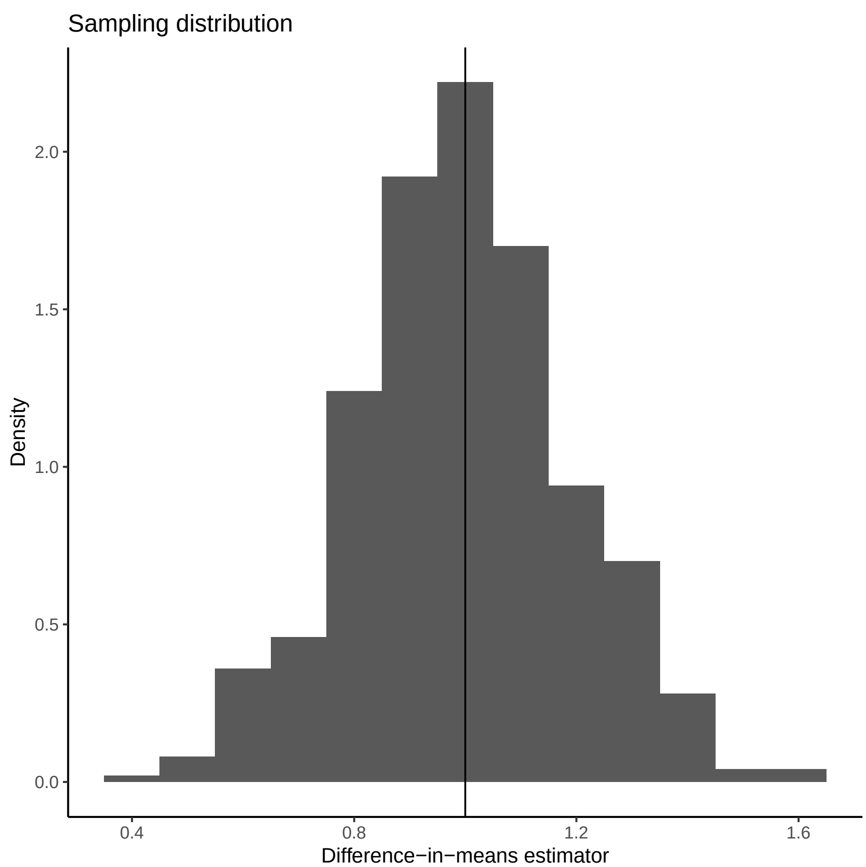
```r
## Update the function for PATE instead of SATE
sim_pate <- function(n, mu0, mu1, sd0, sd1) {
  smpl <- tibble(Y0 = rnorm(n, mean = mu0, sd = sd0),
                 Y1 = rnorm(n, mean = mu1, sd = sd1),
                 tau = Y1 - Y0)
  # indexes of obs receiving treatment
  idx <- sample(seq_len(n), floor(nrow(smpl) / 2), replace = FALSE)
  # treat variable are those receiving treatment, else 0
  smpl[["treat"]] <- as.integer(seq_len(nrow(smpl)) %in% idx)
  smpl %>%
    mutate(Y_obs = if_else(treat == 1L, Y1, Y0)) %>%
    group_by(treat) %>%
    summarize(Y_obs = mean(Y_obs)) %>%
    pivot_wider(names_from = treat,
                values_from = Y_obs) %>%
    rename(Y1_mean = `1`, Y0_mean = `0`) %>%
    mutate(diff_mean = Y1_mean - Y0_mean,
           est_error = diff_mean - PATE)
}
## number of simulations
sims <- 500
```

```r
## run the created function sims times
## input values are defined above
pate_sims <-  map_df(seq_len(sims), ~ sim_pate(n, mu0, mu1, sd0, sd1))
## what is the distribution of the error?
summary(pate_sims$est_error)
```

```
##       Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -0.576352 -0.127658 -0.006791 -0.003441  0.120382  0.618467
```

Standard Error

```r
ggplot(pate_sims, aes(x = diff_mean, y = ..density..)) +
  geom_histogram(binwidth = 0.1) +
  geom_vline(xintercept = PATE, color = "black", size = 0.5) +
  ggtitle("Sampling distribution") +
  labs(x = "Difference-in-means estimator", y = "Density")
```




```r
## the standard deviation of the difference in means
pate_sims %>%
  select(diff_mean) %>%
  summarize(sd = sd(diff_mean))
```

```
## # A tibble: 1 x 1
##      sd
##   <dbl>
## 1 0.193
```

```
RMSE <- pate_sims %>%
  summarize(rmse = sqrt(mean(est_error)^2))
RMSE
```

```
## # A tibble: 1 x 1
##      rmse
##     <dbl>
## 1 0.00344
```

```
## PATE simulation with standard error
sim_pate_se <- function(n, mu0, mu1, sd0, sd1) {
  # PATE - difference in means
  PATE <- mu1 - mu0
  # sample
  smpl <- tibble(Y0 = rnorm(n, mean = mu0, sd = sd0),
                 Y1 = rnorm(n, mean = mu1, sd = sd1),
                 tau = Y1 - Y0)
  # indexes of obs receiving treatment
  idx <- sample(seq_len(n), floor(nrow(smpl) / 2), replace = FALSE)
  # treat variable are those receiving treatment, else 0
  smpl[["treat"]] <- as.integer(seq_len(nrow(smpl)) %in% idx)
  # sample
  smpl %>%
    mutate(Y_obs = if_else(treat == 1, Y1, Y0)) %>%
    group_by(treat) %>%
    summarize(mean = mean(Y_obs),
              var = var(Y_obs),
              nobs = n()) %>%
    summarize(diff_mean = diff(mean),
              se = sqrt(sum(var / nobs)),
              est_error = diff_mean - PATE)
}
## test a single simulation
sim_pate_se(n, mu0, mu1, sd0, sd1)
```

```
## # A tibble: 1 x 3
##   diff_mean    se est_error
##       <dbl> <dbl>     <dbl>
## 1      1.08 0.206    0.0797
```

```
## run 500 times
sims <- 500
pate_sims_se <- map_df(seq_len(sims), ~ sim_pate_se(n, mu0, mu1, sd0, sd1))

## standard deviation of difference-in-means
## and mean of standard errors
sd_se <- pate_sims_se %>%
```

```r
  summarize(sd = sd(diff_mean),
            mean_se = mean(se))
sd_se
```

```
## # A tibble: 1 x 2
##      sd mean_se
##   <dbl>   <dbl>
## 1 0.199   0.200
```

Confidence Interval

```r
## set the sample size
n <- 1000
## set the point estimate
x_bar <- 0.6
## calculate the standard error
se <- sqrt(x_bar * (1 - x_bar) / n)
## set the desired Confidence levels
levels <- c(0.99, 0.95, 0.90)
## build a tibble to calculate the ci at each level
tibble(level = levels) %>%
  mutate(
    ci_lower = x_bar - qnorm(1 - (1 - level) / 2) * se,
    ci_upper = x_bar + qnorm(1 - (1 - level) / 2) * se
  )
```

```
## # A tibble: 3 x 3
##   level ci_lower ci_upper
##   <dbl>    <dbl>    <dbl>
## 1  0.99    0.560    0.640
## 2  0.95    0.570    0.630
## 3  0.9     0.575    0.625
```

```r
## initial confidence level
level <- 0.95
## CI at that level for the PATE simulations with standard errors
pate_sims_ci <- pate_sims_se %>%
  mutate(ci_lower = diff_mean - qnorm(1 - (1 - level) / 2) * se,
         ci_upper = diff_mean + qnorm(1 - (1 - level) / 2) * se,
         includes_pate = PATE > ci_lower & PATE < ci_upper)
## view a subset of the CIs
glimpse(pate_sims_ci)
```

```
## Rows: 500
## Columns: 6
## $ diff_mean     <dbl> 1.0541359, 1.5085375, 0.9039328, 0.8~
## $ se            <dbl> 0.1994236, 0.1953555, 0.1868624, 0.2~
## $ est_error     <dbl> 0.054135883, 0.508537479, -0.0960671~
## $ ci_lower      <dbl> 0.6632727, 1.1256478, 0.5376893, 0.4~
## $ ci_upper      <dbl> 1.444999, 1.891427, 1.270176, 1.2034~
## $ includes_pate <lgl> TRUE, FALSE, TRUE, TRUE, TRUE, TRUE,~
```

```r
## compute the rate of PATE coverage
pate_sims_ci %>%
  summarize(coverage = mean(includes_pate))
```

```
## # A tibble: 1 x 1
##   coverage
##      <dbl>
## 1    0.952
```

```r
pate_sims_coverage <- function(.data, level = 0.95) {
  mutate(.data,
         ci_lower = diff_mean - qnorm(1 - (1 - level) / 2) * se,
         ci_upper = diff_mean + qnorm(1 - (1 - level) / 2) * se,
         includes_pate = PATE > ci_lower & PATE < ci_upper) %>%
    summarize(coverage = mean(includes_pate))
}

pate_sims_coverage(pate_sims_se, level = 0.95)
```

```
## # A tibble: 1 x 1
##   coverage
##      <dbl>
## 1    0.952
```

```r
pate_sims_coverage(pate_sims_se, level = 0.99)
```

```
## # A tibble: 1 x 1
##   coverage
##      <dbl>
## 1    0.984
```

```r
pate_sims_coverage(pate_sims_se, level = 0.90)
```

```
## # A tibble: 1 x 1
##   coverage
##      <dbl>
## 1    0.914
```

```r
## Function to test if CI contains true parameter value
binom_ci_contains <- function(n, p, alpha = 0.05) {
  x <- rbinom(n, size = 1, prob = p)
  x_bar <- mean(x)
  se <- sqrt(x_bar * (1 - x_bar) / n)
  ci_lower <- x_bar - qnorm(1 - alpha / 2) * se
  ci_upper <- x_bar + qnorm(1 - alpha / 2) * se
  (ci_lower <= p) & (p <= ci_upper)
}
## Demonstrate the function
p <- 0.6 # true parameter value
n <- 10
binom_ci_contains(n = n, p = p, alpha = 0.05)
```

```
## [1] TRUE
```

```r
## Show coverage by taking the average of the logical result for each sim
mean(map_lgl(seq_len(sims), ~ binom_ci_contains(n, p)))
```

```
## [1] 0.908
```

```r
## Function to calculate CI coverage while varying number of simulations
binom_ci_coverage <- function(n, p, sims) {
  mean(map_lgl(seq_len(sims), ~ binom_ci_contains(n, p)))
}
## Apply the function to a range of simulations values
tibble(n = c(10, 100, 1000)) %>%
  mutate(coverage = map_dbl(n, binom_ci_coverage,
                            p = p,
                            sims = sims))
```

```
## # A tibble: 3 x 2
##       n coverage
##   <dbl>    <dbl>
## 1    10    0.896
## 2   100    0.952
## 3  1000    0.954
```

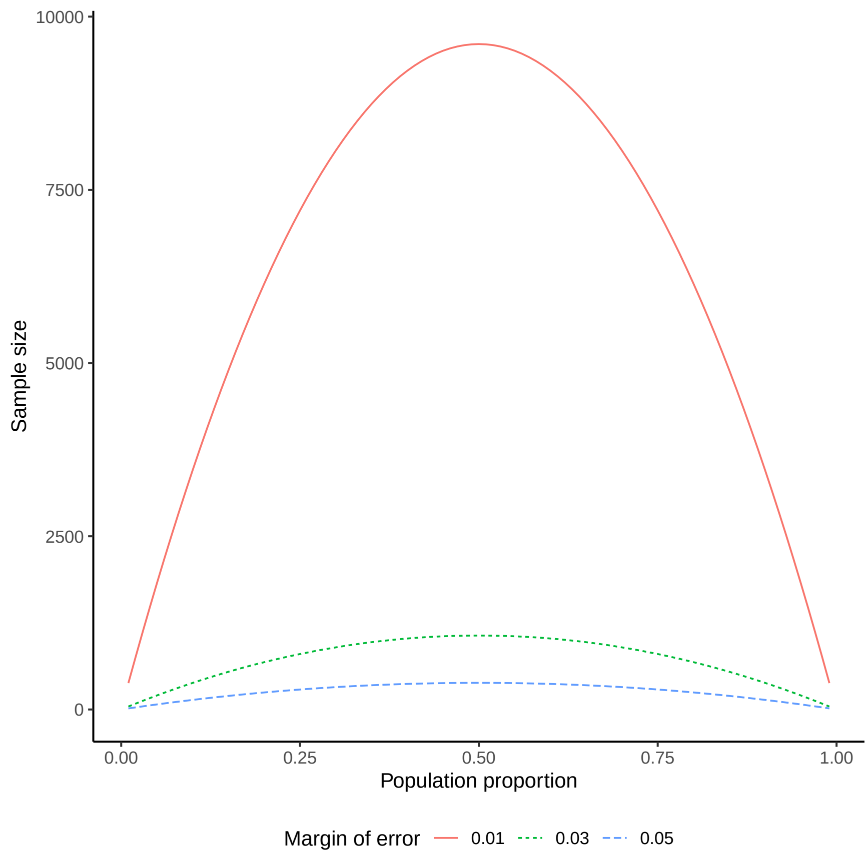
Margin of Error and Sample Size Calculation in Polls

```r
## First, write a function to find population proportion give MoE
moe_pop_prop <- function(MoE) {
  tibble(p = seq(from = 0.01, to = 0.99, by = 0.01),
         n = 1.96 ^ 2 * p * (1 - p) / MoE ^ 2,
         MoE = MoE)
}
glimpse(moe_pop_prop(0.01))
```

```
## Rows: 99
## Columns: 3
## $ p   <dbl> 0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08~
## $ n   <dbl> 380.3184, 752.9536, 1117.9056, 1475.1744, 1824~
## $ MoE <dbl> 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01, 0.01~
```

```r
## Then use map_df to call the function for a range of MoEs
MoE <- c(0.01, 0.03, 0.05)
props <- map_df(MoE, moe_pop_prop)

## plot the results
ggplot(props, aes(x = p, y = n, color = factor(MoE))) +
  geom_line(aes(linetype = factor(MoE))) +
  labs(color = "Margin of error",
       linetype = "Margin of error",
       x = "Population proportion",
       y = "Sample size") +
  theme(legend.position = "bottom")
```

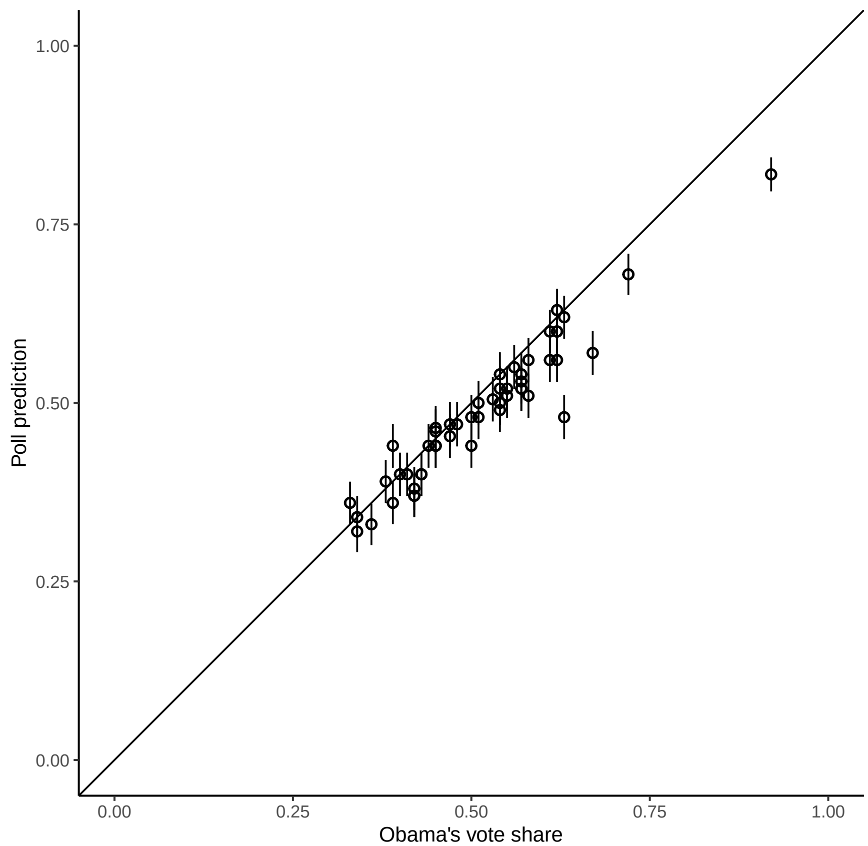



```r
## load required library
library(lubridate)
## load final vote shares
data("pres08", package = "qss")
## load polling data
data("polls08", package = "qss")
## set the election date
ELECTION_DATE <- ymd(20081104)
## Add days to election variable
polls08 <- polls08 %>%
  mutate(DaysToElection = as.integer(ELECTION_DATE - middate))
## Calculate mean of latest polls by state
poll_pred <- polls08 %>%
  group_by(state) %>%
  # latest polls in the state
  filter(DaysToElection == min(DaysToElection)) %>%
  # take mean of latest polls and convert from 0-100 to 0-1
  summarize(Obama = mean(Obama) / 100)
## Add confidence intervals
## sample size (assumed)
sample_size <- 1000
# confidence level
alpha <- 0.05
## Add the CIs and se
poll_pred <- poll_pred %>%
```

```r
  mutate(se = sqrt(Obama * (1 - Obama) / sample_size),
         ci_lwr = Obama + qnorm(alpha / 2) * se,
         ci_upr = Obama + qnorm(1 - alpha / 2) * se)
## Add actual outcome
## And check if coverage includes the actual result
poll_pred <-left_join(poll_pred,
            select(pres08, state, actual = Obama),
            by = "state") %>%
  mutate(actual = actual / 100,
         covers = (ci_lwr <= actual) & (actual <= ci_upr))
## Check the results
glimpse(poll_pred)
```

```
## Rows: 51
## Columns: 7
## $ state  <chr> "AK", "AL", "AR", "AZ", "CA", "CO", "CT", "~
## $ Obama  <dbl> 0.390, 0.360, 0.440, 0.465, 0.600, 0.520, 0~
## $ se     <dbl> 0.01542401, 0.01517893, 0.01569713, 0.01577~
## $ ci_lwr <dbl> 0.3597695, 0.3302498, 0.4092342, 0.4340863,~
## $ ci_upr <dbl> 0.4202305, 0.3897502, 0.4707658, 0.4959137,~
## $ actual <dbl> 0.38, 0.39, 0.39, 0.45, 0.61, 0.54, 0.61, 0~
## $ covers <lgl> TRUE, FALSE, FALSE, TRUE, TRUE, TRUE, FALSE~
```

```r
## Plot the results
ggplot(poll_pred, aes(x = actual, y = Obama)) +
  geom_abline(intercept = 0, slope = 1, color = "black", size = 0.5) +
  geom_pointrange(aes(ymin = ci_lwr, ymax = ci_upr),
                  shape = 1) +
  scale_y_continuous("Poll prediction", limits = c(0, 1)) +
  scale_x_continuous("Obama's vote share", limits = c(0, 1)) +
  scale_color_discrete("CI includes result?") +
  coord_fixed()
```




```
poll_pred %>%
  summarize(mean(covers))
```

```
## # A tibble: 1 x 1
##   `mean(covers)`
##            <dbl>
## 1          0.588
```

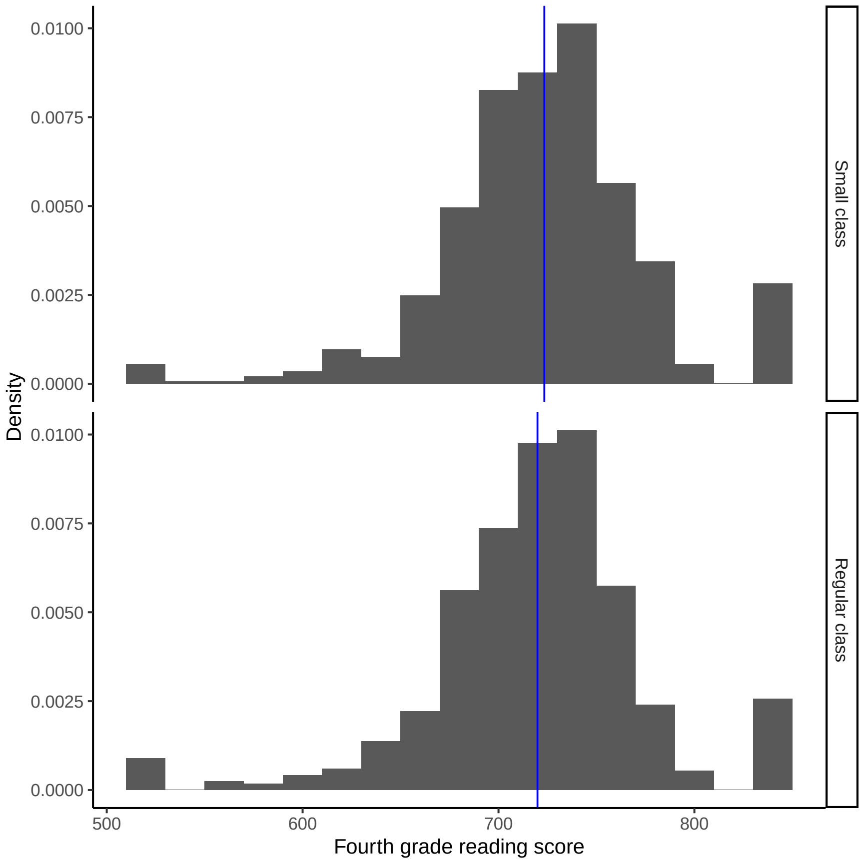
```
poll_pred <- poll_pred %>%
  # calculate the bias
  mutate(bias = Obama - actual) %>%
  # bias corrected prediction, se, and CI
  mutate(Obama_bc = Obama - mean(bias),
         se_bc = sqrt(Obama_bc * (1 - Obama_bc) / sample_size),
         ci_lwr_bc = Obama_bc + qnorm(alpha / 2) * se_bc,
         ci_upr_bc = Obama_bc + qnorm(1 - alpha / 2) * se_bc,
         covers_bc = (ci_lwr_bc <= actual) & (actual <= ci_upr_bc))
## Updated coverage rate
poll_pred %>%
  summarize(mean(covers_bc))
```

```
## # A tibble: 1 x 1
##   `mean(covers_bc)`
##               <dbl>
## 1             0.765
```

Analysis of Randomized Controlled Trials

```r
## load the data
data("STAR", package = "qss")
## Add meaningful labels to the classtype variable:
STAR <- STAR %>%
  mutate(classtype = factor(classtype,
                            labels = c("Small class", "Regular class",
                                       "Regular class with aide")))
## Summarize scores by classroom type:
classtype_means <- STAR %>%
  group_by(classtype) %>%
  summarize(g4reading = mean(g4reading, na.rm = TRUE))

## Plot the distribution of scores by classroom type for two classroom types
classtypes_used <- c("Small class", "Regular class")
ggplot(filter(STAR,
              classtype %in% classtypes_used,
              !is.na(g4reading)),
       aes(x = g4reading, y = ..density..)) +
  geom_histogram(binwidth = 20) +
  geom_vline(data = filter(classtype_means, classtype %in% classtypes_used),
             mapping = aes(xintercept = g4reading),
             color = "blue", size = 0.5) +
  facet_grid(classtype ~ .) +
  labs(x = "Fourth grade reading score", y = "Density")
```




```
## alpha for 95% confidence
alpha <- 0.05
## calculate the mean, se, and CIs
star_estimates <- STAR %>%
  filter(!is.na(g4reading),
         classtype %in% classtypes_used) %>%
  group_by(classtype) %>%
  summarize(n = n(),
            est = mean(g4reading),
            se = sd(g4reading) / sqrt(n)) %>%
  mutate(lwr = est + qnorm(alpha / 2) * se,
         upr = est + qnorm(1 - alpha / 2) * se)

star_estimates
```

```
## # A tibble: 2 x 6
##   classtype         n   est    se   lwr   upr
##   <fct>         <int> <dbl> <dbl> <dbl> <dbl>
## 1 Small class     726  723.  1.91  720.  727.
## 2 Regular class   836  720.  1.84  716.  723.
```

```
## difference-in-means estimator
star_ate <- star_estimates %>%
  # ensure that it is ordered small then regular
  arrange(desc(classtype)) %>%
```

```
  summarize(
    se = sqrt(sum(se ^ 2)),
    est = diff(est)
  ) %>%
  mutate(ci_lwr = est + qnorm(alpha / 2) * se,
         ci_up = est + qnorm(1 - alpha / 2) * se)

star_ate
```

```
## # A tibble: 1 x 4
##      se   est ci_lwr ci_up
##   <dbl> <dbl>  <dbl> <dbl>
## 1  2.65  3.50  -1.70  8.70
```

Analysis based on Student's t Distribution

```
## alpha for 95% confidence
alpha <- 0.05
## calculate the mean, se, and CIs
star_estimates_t <- STAR %>%
  filter(!is.na(g4reading),
         classtype %in% classtypes_used) %>%
  group_by(classtype) %>%
  summarize(n = n(),
            est = mean(g4reading),
            se = sd(g4reading) / sqrt(n)) %>%
  mutate(lwr = est + qt(alpha / 2, df = n - 1) * se,
         upr = est + qt(1 - alpha / 2, df = n - 1) * se)

star_estimates_t
```

```
## # A tibble: 2 x 6
##   classtype         n   est    se   lwr   upr
##   <fct>         <int> <dbl> <dbl> <dbl> <dbl>
## 1 Small class     726  723.  1.91  720.  727.
## 2 Regular class   836  720.  1.84  716.  723.
```

```
## compare to original
star_estimates
```

```
## # A tibble: 2 x 6
##   classtype         n   est    se   lwr   upr
##   <fct>         <int> <dbl> <dbl> <dbl> <dbl>
## 1 Small class     726  723.  1.91  720.  727.
## 2 Regular class   836  720.  1.84  716.  723.
```

```
## compare reading scores between small and regular classes
reading_small <-  filter(STAR, classtype == "Small class")$g4reading
reading_reg <- filter(STAR, classtype == "Regular class")$g4reading
```

```r
t_ci <- t.test(reading_small, reading_reg)

t_ci
```

```
##
##  Welch Two Sample t-test
##
## data:  reading_small and reading_reg
## t = 1.3195, df = 1541.2, p-value = 0.1872
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.703591  8.706055
## sample estimates:
## mean of x mean of y
##  723.3912  719.8900
```

Hypothesis Testing

Lady Tasting Tea Experiment

```r
## Number of cups of tea
cups <- 4
## Number guessed correctly
k <- c(0, seq_len(cups))
## Calculate probability correct
true <-  tibble(correct = k * 2,
                n = choose(cups, k) * choose(cups, cups - k)) %>%
  mutate(prob = n / sum(n))
true
```

```
## # A tibble: 5 x 3
##   correct     n   prob
##     <dbl> <dbl>  <dbl>
## 1       0     1 0.0143
## 2       2    16 0.229
## 3       4    36 0.514
## 4       6    16 0.229
## 5       8     1 0.0143
```

```r
## Number of simulations
sims <- 1000
## The lady's guess (fixed); M for milk first; T for tea first
guess <- tibble(guess = c("M", "T", "T", "M", "M", "T", "T", "M"))

## A function to randomize the tea and calculate correct guesses
randomize_tea <- function(df) {
  # randomize the order of teas
  assignment <- sample_frac(df, 1) %>%
    rename(actual = guess)
  bind_cols(df, assignment) %>%
    summarize(correct = sum(guess == actual))
```

```r
}

## Run the function 1000 times
approx <-
  map_df(seq_len(sims), ~ randomize_tea(guess)) %>%
  count(correct) %>%
  mutate(prob = n / sum(n))

## Then merge with the analytical solution
results <- approx %>%
  select(correct, prob_sim = prob) %>%
  left_join(select(true, correct, prob_exact = prob),
          by = "correct") %>%
  mutate(diff = prob_sim - prob_exact)

results
```

```
## # A tibble: 5 x 4
##   correct prob_sim prob_exact      diff
##     <dbl>    <dbl>      <dbl>     <dbl>
## 1       0    0.017     0.0143  0.00271
## 2       2    0.233     0.229   0.00443
## 3       4    0.508     0.514  -0.00629
## 4       6    0.228     0.229  -0.000571
## 5       8    0.014     0.0143 -0.000286
```

The General Framework

```r
## all correct
x <- matrix(c(4, 0, 0, 4), byrow = TRUE, ncol = 2, nrow = 2)
## six correct
y <- matrix(c(3, 1, 1, 3), byrow = TRUE, ncol = 2, nrow = 2)
## `M' milk first, `T' tea first
rownames(x) <- colnames(x) <- rownames(y) <- colnames(y) <- c("M", "T")
x
```

```
##   M T
## M 4 0
## T 0 4
```

```r
y
```

```
##   M T
## M 3 1
## T 1 3
```

```r
## one-sided test for 8 correct guesses
fisher.test(x, alternative = "greater")
```

```
##
```

```
##  Fisher's Exact Test for Count Data
##
## data:  x
## p-value = 0.01429
## alternative hypothesis: true odds ratio is greater than 1
## 95 percent confidence interval:
##  2.003768      Inf
## sample estimates:
## odds ratio
##        Inf
```

```r
## two-sided test for 6 correct guesses
fisher.test(y)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  y
## p-value = 0.4857
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##    0.2117329 621.9337505
## sample estimates:
## odds ratio
##   6.408309
```

One-sample Tests

```r
n <- 1018
x.bar <- 550 / n
se <- sqrt(0.5 * 0.5 / n) # standard deviation of sampling distribution
## upper red area in the figure
upper <- pnorm(x.bar, mean = 0.5, sd = se, lower.tail = FALSE)
## lower red area in the figure; identical to the upper area
lower <- pnorm(0.5 - (x.bar - 0.5), mean = 0.5, sd = se)
## two-side p-value
upper + lower
```

```
## [1] 0.01016866
```

```r
2 * upper
```

```
## [1] 0.01016866
```

```r
## one-sided p-value
upper
```

```
## [1] 0.005084332
```

```r
z.score <- (x.bar - 0.5) / se
z.score
```

```
## [1] 2.57004
```

```r
pnorm(z.score, lower.tail = FALSE) # one-sided p-value
```

```
## [1] 0.005084332
```

```r
2 * pnorm(z.score, lower.tail = FALSE) # two-sided p-value
```

```
## [1] 0.01016866
```

```r
## 99% confidence interval contains 0.5
c(x.bar - qnorm(0.995) * se, x.bar + qnorm(0.995) * se)
```

```
## [1] 0.4999093 0.5806408
```

```r
## 95% confidence interval does not contain 0.5
c(x.bar - qnorm(0.975) * se, x.bar + qnorm(0.975) * se)
```

```
## [1] 0.5095605 0.5709896
```

```r
## no continuity correction to get the same p-value as above
prop.test(550, n = n, p = 0.5, correct = FALSE)
```

```
##
##  1-sample proportions test without continuity
##  correction
##
## data:  550 out of n, null probability 0.5
## X-squared = 6.6051, df = 1, p-value = 0.01017
## alternative hypothesis: true p is not equal to 0.5
## 95 percent confidence interval:
##  0.5095661 0.5706812
## sample estimates:
##        p
## 0.540275
```

```r
## with continuity correction
prop.test(550, n = n, p = 0.5)
```

```
##
##  1-sample proportions test with continuity correction
##
## data:  550 out of n, null probability 0.5
## X-squared = 6.445, df = 1, p-value = 0.01113
## alternative hypothesis: true p is not equal to 0.5
## 95 percent confidence interval:
```

```
##  0.5090744 0.5711680
## sample estimates:
##        p
## 0.540275
```

```
prop.test(550, n = n, p = 0.5, conf.level = 0.99)
```

```
##
##  1-sample proportions test with continuity correction
##
## data:  550 out of n, null probability 0.5
## X-squared = 6.445, df = 1, p-value = 0.01113
## alternative hypothesis: true p is not equal to 0.5
## 99 percent confidence interval:
##  0.4994182 0.5806040
## sample estimates:
##        p
## 0.540275
```

```
# two-sided one-sample t-test
t.test(STAR$g4reading, mu = 710)
```

```
##
##  One Sample t-test
##
## data:  STAR$g4reading
## t = 10.407, df = 2352, p-value < 2.2e-16
## alternative hypothesis: true mean is not equal to 710
## 95 percent confidence interval:
##  719.1284 723.3671
## sample estimates:
## mean of x
##  721.2478
```

Two-sample Tests

```
star_ate %>%
  mutate(p_value_1sided = pnorm(-abs(est),
                                mean = 0, sd = se),
         p_value_2sided = 2 * pnorm(-abs(est), mean = 0,
                                    sd = se))
```

```
## # A tibble: 1 x 6
##      se   est ci_lwr ci_up p_value_1sided p_value_2sided
##   <dbl> <dbl>  <dbl> <dbl>          <dbl>          <dbl>
## 1  2.65  3.50  -1.70  8.70         0.0935          0.187
```

```
## testing the null of zero average treatment effect
reading_small <- filter(STAR, classtype == "Small class")$g4reading
reading_reg <- filter(STAR, classtype == "Regular class")$g4reading
```

```r
## t-test
t.test(reading_small,
       reading_reg)
```

```
##
##  Welch Two Sample t-test
##
## data:  reading_small and reading_reg
## t = 1.3195, df = 1541.2, p-value = 0.1872
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.703591  8.706055
## sample estimates:
## mean of x mean of y
##  723.3912  719.8900
```

```r
## load the data
data("resume", package = "qss")
## reshape the data
x <- resume %>%
  count(race, call) %>%
  pivot_wider(names_from = call, values_from = n) %>%
  ungroup()
x
```

```
## # A tibble: 2 x 3
##   race    `0`   `1`
##   <chr> <int> <int>
## 1 black  2278   157
## 2 white  2200   235
```

```r
## run the test on the relevant columns
prop.test(as.matrix(select(x, -race)), alternative = "greater")
```

```
##
##  2-sample test for equality of proportions with
##  continuity correction
##
## data:  as.matrix(select(x, -race))
## X-squared = 16.449, df = 1, p-value = 2.499e-05
## alternative hypothesis: greater
## 95 percent confidence interval:
##  0.01881967 1.00000000
## sample estimates:
##    prop 1    prop 2
## 0.9355236 0.9034908
```

```r
## sample size
n0 <- sum(resume$race == "black")
n1 <- sum(resume$race == "white")

## sample proportions
```

```r
p <- mean(resume$call) # overall
p0 <- mean(filter(resume, race == "black")$call)
p1 <- mean(filter(resume, race == "white")$call)

## point estimate
est <- p1 - p0
est
```

```
## [1] 0.03203285
```

```r
## standard error
se <- sqrt(p * (1 - p) * (1 / n0 + 1 / n1))
se
```

```
## [1] 0.007796894
```

```r
## z-statistic
zstat <- est / se
zstat
```

```
## [1] 4.108412
```

```r
## one-sided p-value
pnorm(-abs(zstat))
```

```
## [1] 1.991943e-05
```

```r
prop.test(as.matrix(select(x, -race)), alternative = "greater", correct = FALSE)
```

```
##
##  2-sample test for equality of proportions without
##  continuity correction
##
## data:  as.matrix(select(x, -race))
## X-squared = 16.879, df = 1, p-value = 1.992e-05
## alternative hypothesis: greater
## 95 percent confidence interval:
##  0.01923035 1.00000000
## sample estimates:
##    prop 1    prop 2
## 0.9355236 0.9034908
```

Pitfalls of Hypothesis Testing

Power Analysis

```r
## set the parameters
n <- 250
p.star <- 0.48 # data generating process
p <- 0.5 # null value
alpha <- 0.05
## critical value
cr.value <- qnorm(1 - alpha / 2)
## standard errors under the hypothetical data generating process
se.star <- sqrt(p.star * (1 - p.star) / n)
## standard error under the null
se <- sqrt(p * (1 - p) / n)
## power
pnorm(p - cr.value * se, mean = p.star, sd = se.star) +
  pnorm(p + cr.value * se, mean = p.star, sd = se.star, lower.tail = FALSE)
```

```
## [1] 0.09673114
```

```r
## parameters
n1 <- 500
n0 <- 500
p1.star <- 0.05
p0.star <- 0.1
```

```r
## overall call back rate as a weighted average
p <- (n1 * p1.star + n0 * p0.star) / (n1 + n0)
## standard error under the null
se <- sqrt(p * (1 - p) * (1 / n1 + 1 / n0))
## standard error under the hypothetical data generating process
se.star <- sqrt(p1.star * (1 - p1.star) / n1
                + p0.star * (1 - p0.star) / n0)
```

```r
pnorm(-cr.value * se, mean = p1.star - p0.star, sd = se.star) +
  pnorm(cr.value * se, mean = p1.star - p0.star, sd = se.star,
        lower.tail = FALSE)
```

```
## [1] 0.85228
```

```r
power.prop.test(n = 500, p1 = 0.05, p2 = 0.1, sig.level = 0.05)
```

```
##
##      Two-sample comparison of proportions power calculation
##
##               n = 500
##              p1 = 0.05
##              p2 = 0.1
##       sig.level = 0.05
##           power = 0.8522797
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

```r
power.prop.test(p1 = 0.05, p2 = 0.1, sig.level = 0.05, power = 0.9)
```

```
##
##      Two-sample comparison of proportions power calculation
##
##               n = 581.0821
##              p1 = 0.05
##              p2 = 0.1
##       sig.level = 0.05
##           power = 0.9
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

```r
power.t.test(n = 100, delta = 0.25, sd = 1, type = "one.sample")
```

```
##
##      One-sample t test power calculation
##
##               n = 100
##           delta = 0.25
##              sd = 1
##       sig.level = 0.05
##           power = 0.6969757
##     alternative = two.sided
```

```r
power.t.test(power = 0.9, delta = 0.25, sd = 1, type = "one.sample")
```

```
##
##      One-sample t test power calculation
##
##               n = 170.0511
##           delta = 0.25
##              sd = 1
##       sig.level = 0.05
##           power = 0.9
##     alternative = two.sided
```

```r
power.t.test(delta = 0.25, sd = 1, type = "two.sample",
             alternative = "one.sided", power = 0.9)
```

```
##
##      Two-sample t test power calculation
##
##               n = 274.7222
##           delta = 0.25
##              sd = 1
##       sig.level = 0.05
##           power = 0.9
##     alternative = one.sided
##
## NOTE: n is number in *each* group
```

Linear Regression Model with Uncertainty

Linear Regression as a Generative Model

```r
## load the data
data("minwage", package = "qss")
## compute proportion of full employment before minimum wage increase
## same thing after minimum wage increase
## an indicator for NJ: 1 if it's located in NJ and 0 if in PA
minwage <-
  mutate(minwage,
         fullPropBefore = fullBefore / (fullBefore + partBefore),
         fullPropAfter = fullAfter / (fullAfter + partAfter),
         NJ = if_else(location == "PA", 0 , 1))
```

```r
fit_minwage <- lm(fullPropAfter ~ -1 + NJ + fullPropBefore +
                    wageBefore + chain, data = minwage)
## regression result
fit_minwage
```

```
##
## Call:
## lm(formula = fullPropAfter ~ -1 + NJ + fullPropBefore + wageBefore +
##     chain, data = minwage)
##
## Coefficients:
##              NJ   fullPropBefore       wageBefore
##         0.05422          0.16879          0.08133
## chainburgerking         chainkfc        chainroys
##        -0.11563         -0.15080         -0.20639
##     chainwendys
##        -0.22013
```

```r
## with intercept
fit_minwage1 <- lm(fullPropAfter ~ NJ + fullPropBefore +
                     wageBefore + chain, data = minwage)
fit_minwage1
```

```
##
## Call:
## lm(formula = fullPropAfter ~ NJ + fullPropBefore + wageBefore +
##     chain, data = minwage)
##
## Coefficients:
##    (Intercept)              NJ  fullPropBefore
##       -0.11563         0.05422         0.16879
##     wageBefore        chainkfc       chainroys
##        0.08133        -0.03517        -0.09076
##    chainwendys
##       -0.10451
```

```r
## load the required package
library(modelr)
## Generate prediction from the first model
pred_1 <-minwage %>%
  slice(1) %>%
  add_predictions(fit_minwage) %>%
  select(pred) %>%
  mutate(model = "fit_minwage")
## Generate prediction from the second model
## then add predictions from first to compare
pred_compare <-minwage %>%
  slice(1) %>%
  add_predictions(fit_minwage1) %>%
  select(pred) %>%
  mutate(model = "fit_minwage1") %>%
  bind_rows(pred_1)

pred_compare
```

```
##        pred        model
## 1 0.2709367 fit_minwage1
## 2 0.2709367  fit_minwage
```

Unbiasedness of Estimated Coefficients

Standard Errors of Estimated Coefficients

Inference about Coefficients

```r
library(broom)
## load the data
data("women", package = "qss")
## fit the model
fit_women <- lm(water ~ reserved, data = women)
## view the coefficients
summary(fit_women)
```

```
##
## Call:
## lm(formula = water ~ reserved, data = women)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -23.991 -14.738  -7.865   2.262 316.009
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   14.738      2.286   6.446 4.22e-10 ***
## reserved       9.252      3.948   2.344   0.0197 *
## ---
## Signif. codes:
```

```
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 33.45 on 320 degrees of freedom
## Multiple R-squared:  0.01688,    Adjusted R-squared:  0.0138
## F-statistic: 5.493 on 1 and 320 DF,  p-value: 0.0197
```

```
tidy(fit_women)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)    14.7       2.29      6.45 4.22e-10
## 2 reserved        9.25      3.95      2.34 1.97e- 2
```

```
## display confidence intervals
tidy(fit_women, conf.int = TRUE)
```

```
## # A tibble: 2 x 7
##   term        estimate std.error statistic  p.value conf.low
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>    <dbl>
## 1 (Intercept)    14.7       2.29      6.45 4.22e-10    10.2
## 2 reserved        9.25      3.95      2.34 1.97e- 2     1.49
## # ... with 1 more variable: conf.high <dbl>
```

```
summary(fit_minwage)
```

```
##
## Call:
## lm(formula = fullPropAfter ~ -1 + NJ + fullPropBefore + wageBefore +
##     chain, data = minwage)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.48617 -0.18135 -0.02809  0.15127  0.75091
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## NJ               0.05422    0.03321   1.633  0.10343
## fullPropBefore   0.16879    0.05662   2.981  0.00307 **
## wageBefore       0.08133    0.03892   2.090  0.03737 *
## chainburgerking -0.11563    0.17888  -0.646  0.51844
## chainkfc        -0.15080    0.18310  -0.824  0.41074
## chainroys       -0.20639    0.18671  -1.105  0.26974
## chainwendys     -0.22013    0.18840  -1.168  0.24343
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.2438 on 351 degrees of freedom
## Multiple R-squared:  0.6349, Adjusted R-squared:  0.6277
## F-statistic: 87.21 on 7 and 351 DF,  p-value: < 2.2e-16
```

```r
tidy(fit_minwage, conf.int = TRUE)
```

```
## # A tibble: 7 x 7
##   term          estimate std.error statistic p.value conf.low
##   <chr>            <dbl>     <dbl>     <dbl>   <dbl>    <dbl>
## 1 NJ              0.0542    0.0332     1.63  0.103   -0.0111
## 2 fullPropBef~    0.169     0.0566     2.98  0.00307  0.0574
## 3 wageBefore      0.0813    0.0389     2.09  0.0374   0.00478
## 4 chainburger~   -0.116     0.179     -0.646 0.518   -0.467
## 5 chainkfc       -0.151     0.183     -0.824 0.411   -0.511
## 6 chainroys      -0.206     0.187     -1.11  0.270   -0.574
## 7 chainwendys    -0.220     0.188     -1.17  0.243   -0.591
## # ... with 1 more variable: conf.high <dbl>
```

Inference about Predictions

```r
## load the data and subset them into two parties
data("MPs", package = "qss")
MPs_labour <- filter(MPs, party == "labour")
MPs_tory <- filter(MPs, party == "tory")

## two regressions for labour: negative and positive margin
labour_fit1 <- lm(ln.net ~ margin, data = filter(MPs_labour, margin < 0))
labour_fit2 <- lm(ln.net ~ margin, data = filter(MPs_labour, margin > 0))

## two regressions for tory: negative and positive margin
tory_fit1 <- lm(ln.net ~ margin, data = filter(MPs_tory, margin < 0))
tory_fit2 <- lm(ln.net ~ margin, data = filter(MPs_tory, margin > 0))
```

```r
## tory party: prediction at the threshold
tory_y0 <- augment(tory_fit1, newdata = tibble(margin = 0),
          interval = "confidence",
          conf.level = 0.95)

tory_y0
```

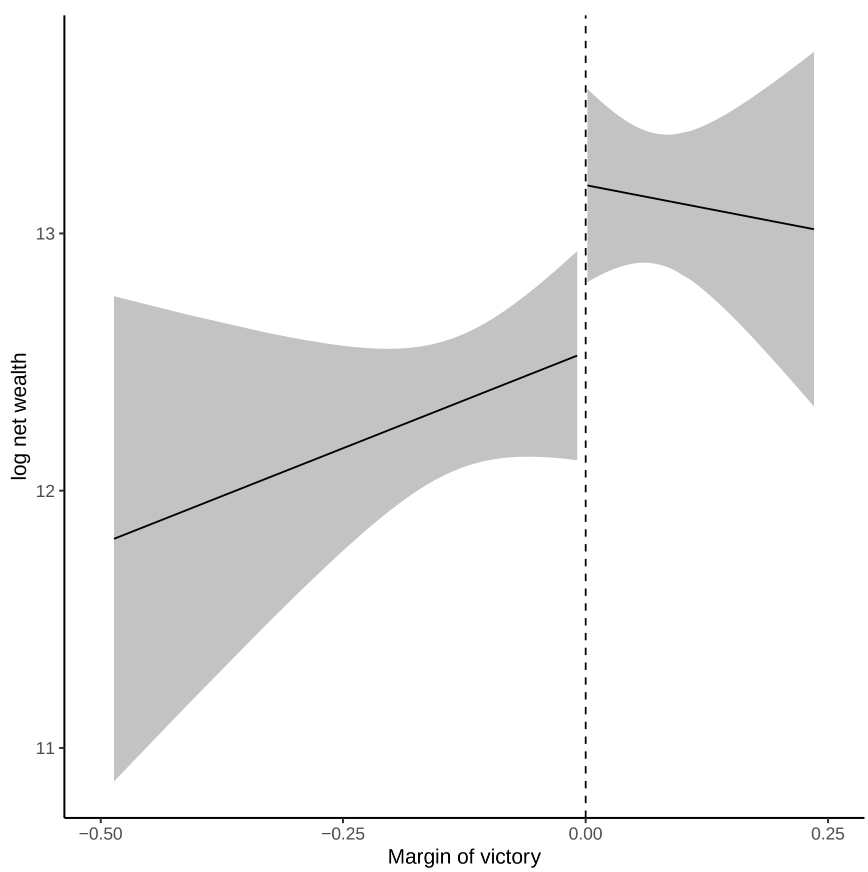
```
## # A tibble: 1 x 4
##   margin .fitted .lower .upper
##    <dbl>   <dbl>  <dbl>  <dbl>
## 1      0    12.5   12.1   13.0
```

```r
tory_y1 <- augment(tory_fit2, newdata = tibble(margin = 0),
                   interval = "confidence")
tory_y1
```

```
## # A tibble: 1 x 4
##   margin .fitted .lower .upper
##    <dbl>   <dbl>  <dbl>  <dbl>
## 1      0    13.2   12.8   13.6
```

```r
## create data with the ranges of "margin" less than zero
y1_range <- data_grid(filter(MPs_tory, margin <= 0), margin)
## add predictions and CIs
tory_y0 <- augment(tory_fit1, newdata = y1_range,
                   interval = "confidence")
## create the data with the ranges of "margin" greater than zero
y2_range <- data_grid(filter(MPs_tory, margin >= 0), margin)
## add predictions and CIs
tory_y1 <- augment(tory_fit2, newdata = y2_range,
                   interval = "confidence")
```

```r
ggplot() +
  geom_vline(xintercept = 0, linetype = "dashed") +
  # plot losers
  geom_ribbon(aes(x = margin, ymin = .lower, ymax = .upper),
              data = tory_y0, alpha = 0.3) +
  geom_line(aes(x = margin, y = .fitted), data = tory_y0) +
  # plot winners
  geom_ribbon(aes(x = margin, ymin = .lower, ymax = .upper),
              data = tory_y1, alpha = 0.3) +
  geom_line(aes(x = margin, y = .fitted), data = tory_y1) +
  xlim(-.5,.25) +
  labs(x = "Margin of victory", y = "log net wealth")
```



```r
## predictions at threshold with SEs
tory_y0 <- augment(tory_fit1, newdata = tibble(margin = 0),
          interval = "confidence",
          se_fit = TRUE)

tory_y0
```

```
## # A tibble: 1 x 5
##   margin .fitted .lower .upper .se.fit
##    <dbl>   <dbl>  <dbl>  <dbl>   <dbl>
## 1      0    12.5   12.1   13.0   0.214
```

```r
tory_y1 <- augment(tory_fit2, newdata = tibble(margin = 0),
                   interval = "confidence",
                   se_fit = TRUE)
tory_y1
```

```
## # A tibble: 1 x 5
##   margin .fitted .lower .upper .se.fit
##    <dbl>   <dbl>  <dbl>  <dbl>   <dbl>
## 1      0    13.2   12.8   13.6   0.192
```

```r
summary(tory_fit2)
```

```
##
## Call:
## lm(formula = ln.net ~ margin, data = filter(MPs_tory, margin >
##     0))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.8575 -0.8769  0.0007  0.8297  3.1257
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  13.1878     0.1920  68.693   <2e-16 ***
## margin       -0.7278     1.9824  -0.367    0.714
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.289 on 100 degrees of freedom
## Multiple R-squared:  0.001346,   Adjusted R-squared:  -0.008641
## F-statistic: 0.1348 on 1 and 100 DF,  p-value: 0.7143
```

```r
# standard error
se_diff <- sqrt(tory_y0$.se.fit ^ 2 + tory_y1$.se.fit ^ 2)
se_diff
```

```
## [1] 0.2876281
```

```r
# point estimate
diff_est <- tory_y1$.fitted - tory_y0$.fitted
diff_est
```

```
## [1] 0.6496861
```

```r
# confidence interval
CI <- c(diff_est - se_diff * qnorm(0.975),
        diff_est + se_diff * qnorm(0.975))
CI
```

```
## [1] 0.0859455 1.2134268
```

```r
# hypothesis test
z_score <- diff_est / se_diff
# two sided p value
p_value <- 2 * pnorm(abs(z_score), lower.tail = FALSE)
p_value
```

```
## [1] 0.02389759
```